THE DESIGN OF AN OPEN, SECURE AND SCALABLE BLOCKCHAIN-BASED ARCHITECTURE TO EXCHANGE TRADE DOCUMENTS IN TRADE LANES

LENNARD SEGERS

MASTER THESIS
FEBRUARY 2019

DELFT UNIVERSITY OF TECHNOLOGY
MASTER COMPLEX SYSTEMS ENGINEERING AND MANAGEMENT

TU Delft

THE DESIGN OF AN OPEN, SECURE AND SCALABLE BLOCKCHAIN-BASED ARCHITECTURE TO EXCHANGE TRADE DOCUMENTS IN TRADE LANES

Master thesis submitted to Delft University of Technology

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in Complex Systems Engineering and Management

Faculty of Technology, Policy and Management

by

Lennard Segers

To be defended in public on February 21st, 2019

Graduation committee

Preface

This thesis is the concluding part of my Master of Complex Systems Engineering and Management at the Delft University of Technology. Having an interest in the combination of IT innovation and supply chain management, researching the applicability of innovative blockchain technology within trade lanes was an interesting journey. The combination of a highly complex domain and novel technology made the research a challenging path to follow, but definitely satisfying in the end.

I could not have finished the thesis without the help of a group of people. Hereby, I would like to thank my graduation committee for their time and effort to guide me on this journey. First, I would like to thank Yao-Hua Tan. I enjoyed our discussions on the technicalities of blockchain technology and how the technology could fit within the trade lane domain. His knowledge on and passion for IT innovation in trade lanes is admirable and to a great extent helped me shaping the thesis as he provided me with critical and constructive comments. Next, I thank Jolien Ubacht for her tremendous help with structuring the thesis when I got a bit lost. The regular discussions we had resulted in a well-structured thesis. I thank Jan-Anne Annema for his quirky comments that helped me to sharpen my research and Boriana Rukanova for positioning the research within the trade lane domain. I also thank the interviewees for their time to discuss the findings of the research and provide me with valuable insights in the societal relevance of this thesis.

Finally, I would like to thank my family and friends and fellow students. Even though it took a bit longer than expected, they always supported me. This leaves me with answering the often-asked question: 'Is het nou al klaar?'. Ja.

Enjoy reading,

Lennard Segers
February 2019

Executive summary

To secure borders and collect customs duties, customs authorities perform risk assessment to determine which shipments should be inspected. The import declaration is the main source risk assessment is based on. For proper risk assessment, the declaration should be sufficiently detailed and correct. However, the import declaration is often not sufficiently detailed and correctness cannot be ensured. As a result, risk assessment is more difficult. This negatively affects security and increases tax fraud. Consequently, it leads to more inspections that in return cause delays and increasing administrative costs. To improve risk assessment, customs authorities need better exchange of trade data within trade lanes to improve visibility. The import declaration contains aggregated data from other documents. If customs authorities have access to these documents, they can cross-validate the import declaration to ensure correctness.

In recent years, a new technology to improve the exchange of trade data has gained attention: blockchain technology. Blockchain technology enables parties that do not (fully) trust each other to exchange data without the need for a central ledger that keeps track of exchange transactions or is managed by a trusted intermediary. The technology is based on cryptography and a peer-to-peer network of parties in which a blockchain ledger is distributed. The blockchain ledger consists of timestamped and digitally signed transactions that are stored in blocks that are cryptographically linked to other blocks to guarantee immutability. New transactions can only be added if the network reaches consensus on validity of a transaction. Because the exchange of trade documents takes place in distributed networks of many parties, blockchain technology can be of great added value to both businesses and customs authorities.

Whilst the potential of using blockchain technology is known, there is lack of guidance on how blockchain technology can be used to support the exchange of trade documents. In addition, because of the novelty of blockchain technology, many challenges are currently experienced. In context of the exchange of trade documents, multiple trade-offs between openness, security and scalability of a blockchain-based system to support exchange are found. This shows the need for guidance on how blockchain technology can support exchange of trade documents, especially given the challenges. The research addresses this objective by designing an open, secure and scalable blockchain-based architecture to support exchange of trade documents in trade lanes. The design process is structured by adopting a Design Science Research approach that combines insights from the problem domain and academic literature. To fulfill the objective, the following main research question is formulated:

Which design of a blockchain-based architecture can be developed to support exchange of trade documents in trade lanes that takes into account the requirements of openness, security and scalability?

The research approach is conducted in five steps: 1) explicate problem, 2) define requirements, 3) design and develop, 4) demonstrate and 5) evaluate. In the problem explication, a set of use cases is developed. Because exchange of trade documents is a broad concept, exchange is narrowed to three use cases that exchange different documents. These use cases are crucial because support via a blockchain-based architecture can in potential enable many more use cases.
In use case I, a pro forma invoice is exchanged by a consignor that customs authorities can piggyback for manual cross-validation of an import declaration. An extension of use case I is found in the automated generation of the import declaration by aggregating already exchanged pro forma invoice data. Because the declaration can only be generated using pro forma invoice data, the need for manual cross-validation is reduced. A third use case aims to support the most difficult trade document: the negotiable Bill of Lading. As the negotiable Bill of Lading can be transferred to another holder, uniqueness must be guaranteed. It can therefore not be treated like any other document that does not require a guarantee of uniqueness.

Based on the use cases, using a literature review on both the data pipeline concept and electronic negotiable Bill of Lading design requirements for the blockchain-based architecture to support the use cases are defined. The requirements relate to the requirements for openness, security (confidentiality and integrity) and scalability as formulated in the main research question.

To structure the blockchain-based architecture design, based on a literature review four core components of a blockchain-based architecture are defined: 1) network configuration, 2) data structure and storage, 3) consensus mechanism and 4) application. The *network configuration* concerns the peer-to-peer network of nodes in which a data structure (also known as blockchain ledger) is distributed and how nodes participate in the network. The *data structure and storage* component determines the way in which data within the network is structured and stored. The state of the blockchain ledger is updated via a set of rules and procedures to validate transactions according to the *consensus mechanism* component. The *application* component determines what is being stored on the blockchain ledger and how business logic is implemented. For each component a set of sub components and alternatives was identified that are used as design options.

Using the core components as basis, for each use case a detailed blockchain-based architecture design is developed based on the design requirements. A set of typical user activities that stem from the use cases demonstrate how the blockchain-based architecture design supports the exchange of trade documents. Mapping of the blockchain-based architecture design on the requirements for evaluation purposes finds that the design fulfills all requirements related to openness, security and scalability for each use case. Expert interviews to evaluate the relevance of the critical trade document use cases confirms that support can greatly benefit both businesses and customs authorities. Evaluation by expert opinion of the effectiveness and technical correctness of the blockchain-based architecture design finds that the architecture is well-embedded in the domain.

The research finds that a blockchain-based architecture design to exchange trade documents can be developed that takes into account openness, security and scalability. The design supports each use case and shows many similarities per case. For each use case, the architecture is governed by a consortium of parties to guarantee openness. Security is realized by using a permissioned network configuration and using data encryption. A data structure and storage is used to store transactions to exchange (references to) trade documents. Within the consensus mechanism, Practical Byzantine Fault Tolerance is used as consensus protocol to provide scalability. An application uses smart contracts to implement all business logic. However, subtle differences in the blockchain-based architecture design of each use case exist. This shows that generalization of the architecture to support exchange of any trade document is not straightforward.

In a wider perspective, there are not only differences between different domains, but also within a single domain. A blockchain-based architecture can thus not be seen as a black box. In-depth analysis of individual use cases is therefore needed to develop effective designs. Because the research shows that the three critical trade documents are supported by the blockchain-based architecture, to some extent generalization can be made for the exchange of other trade documents. If the crucial trade documents can be exchanged, then exchange of other documents will probably also be supported by a blockchain-based architecture. Therefore, the blockchain-based architecture design is a good first step in the implementation of an architecture to support actual exchange of trade documents that can benefit both customs authorities and businesses in trade lanes.

Future research should focus on development of a Proof-of-Concept based on the blockchain-based architecture design to determine if the architecture supports exchange of trade documents in real-world application. Whilst the design is based on requirements derived from academic literature such as the data pipeline concept which is highly relevant for the exchange of trade documents, there is a

potential gap between design and practical real-world application. This includes research on the scalability of the architecture using Practical Byzantine Fault Tolerance as it is yet only proved theoretically and by expert opinion. Future research should also focus on the need for standardization to support exchange and relevance of use cases from a business perspective.

Contents

Terminology

Term	Definition
Trade lane	Businesses and (Customs) authorities involved in a shipment of goods from consignor to consignee.
International shipment	Shipment of goods under the same terms from consignor to consignee which includes cross-border activities
Consignor	Business responsible for shipping the goods. Can also be referred to as: shipper or seller.
Consignee	Business goods are shipped to. Can also be referred to as: buyer.
Carrier	Business that operates a container vessel and is responsible for shipping goods by sea from port of loading to port of discharge.
Freight Forwarder	Business operating on behalf of consignor and/or consignee that is responsible for arranging transport including preparation and processing of documentation pertaining to the shipment.
Customs authority	Public authority responsible for secure cross-border flow of goods and government tax revenue.
Pro forma invoice document	Preliminary commercial invoice document prepared by the consignor and exchanged with the consignee as commercial invoice is often only exchanged after delivery. Acts as a form of transport document on which other documents are based.
Import declaration	Document prepared by a declarant (consignor or Freight Forwarder on behalf of consignor) and exchanged with customs authority to declare a shipment of goods for import.
Negotiable Bill of Lading	Document issued by a carrier to the consignor as contract of carriage, receipt for goods loaded and title to the goods.
Blockchain ledger	Database of cryptographically linked blocks consisting of multiple ordered transactions in which each block contains the cryptographic has of the previous block.
Transaction	Exchange of any type of digital asset between two or more parties.
Consensus mechanism	Mechanism that enables parties in the network to reach agreement on the validity of transactions without the need for a trusted intermediary.
Smart contract	Agreement between two or more parties that is deployed as code on the blockchain ledger.

List of figures

List of tables

1. Introduction

In international (container) shipping, customs authorities are responsible for secure cross-border activities and collection of customs duties. The sheer volume of shipments[1] makes it impossible to physically inspect each shipment. For example, the Port of Rotterdam handles over ten million container shipments each year (Francisconi, 2017). Therefore, customs authorities rely on data-based risk assessment to determine which shipments are subject to physical inspection.

Customs authorities mostly use the customs declaration (e.g., the import declaration) to base risk assessment on. For proper risk assessment and collection of customs duties, customs authorities require the customs declaration to contain sufficiently detailed data[2]. This data includes information on the consignor and consignee of a shipment, a detailed description of the goods and the country of origin (Hesketh, 2010). However, the data provided on the declaration is often vague or inaccurate (Hesketh, 2010; WCO, 2005). One reason is that the customs declaration contains aggregated information from other documents like the Bill of Lading used by a carrier. In return, the bill of lading contains aggregated data from other documents.

Another reason is that international trade lanes[3] for the shipping of goods are complex. In a typical trade lane many parties, both businesses like consignors, consignees, Freight Forwarders and carriers (e.g., Maersk Line) and public authorities like the customs authority are involved. A case study by Jensen, Vatrapu & Bjørn-Andersen (2018) on the international shipment of a container of avocados shows forty trade lane parties that together exchange around two hundred trade data items can be involved in an international shipment. As a result of the complexity, Freight Forwarders often manage required paperwork. For example, they act as declarant to lodge an import declaration. This results in import declarations that contain only limited required details and often label the Freight Forwarder as both consignor and consignee (Hesketh, 2010).

The import declaration does therefore not provide sufficient details or ensure correctness (Hesketh, 2010). For example, as the Freight Forwarder is labeled as consignor and consignee, the customs authority lacks detailed information on the original consignor and consignee needed for proper risk assessment. This makes risk assessment more difficult and thus requires more physical inspections. As a consequence, this leads to delay in cross-border activities and administrative costs. Currently, the administrative cost of a shipment account for twenty per cent of the total costs (Thomas & Tan, 2015). Also, due to vague or inaccurate information, cross-border tax and duty fraud via falsified documentation is a growing problem in international shipping (Triepels, Daniels, & Feelders, 2018). There is thus a need to improve exchange of trade data to improve visibility of trade data within

---

[1] In this research the scope of analysis is a shipment. A shipment consists of a number of goods that are shipped under the same terms (BusinessDictionary.com, n.d.). Trade data (e.g., trade documents) is typically linked to a single shipment.

[2] Trade data consists of two types of data: 1) shipping events and 2) trade documents. Examples of shipping events are 'packed container loaded' if a container carrier has loaded the container on board or 'container selected for inspection' if a customs administration wants to notify a party that the container is subject to physical inspection. Trade documents examples are the 'Bill of Lading' that a container or air carrier provides to the shipper as a contract of carriage upon loading the container on board or 'import declaration' that is used by a declarant to declare the importation of goods. As many of the issues of exchanging trade data between trade lane parties relate to trade documents, the focus in this research is on this type of trade data.

[3] In research on international shipping the terms supply chain and (international) trade lane are often used intertwined. An international trade lane can be seen as the part of a supply chain which focusses on the collaboration between businesses and public authorities to fulfill the international shipments of goods that involves cross-border customs procedures. Therefore, in the remainder of this research the term 'trade lane' is used to emphasize that the scope is limited to the exchange of trade data (e.g., documents) between parties that plays a role in international shipping. Thus, the research does not concern supply chain activities like procurement or manufacturing.

trade lanes. Improved visibility enables customs authorities to access detailed data on a shipment that leads to better risk assessment.

However, the complexity of trade lanes makes exchange of trade data difficult. Outdated paper trails, data deficiencies and difficult to access data is common (Hesketh, 2010; Klievink et al., 2012). Exchange is often limited to immediate trade parties in the trade lane (Jensen et al., 2018) as illustrated with dotted arrows in Figure 1. The reason is that trade data is often stored in individual 'data silos' of trade lane parties. The number of data silos (e.g., ERP information systems to store trade documents) in a trade lane can be over thirty (Jensen et al., 2018). Interorganizational information systems (IOIS) to allow for exchange of trade data by connecting these data silos to create end-to-end visibility have limited adoption. Interoperability and accessibility issues make these IOIS complex and expensive to realize. This results in solutions that are only capable of exchanging trade data (e.g., documents) with other parties that a trade lane party is directly involved with, which reduces end-to-end visibility (Klievink et al., 2012; Klievink & Zomer, 2015). For example, a consignor only exchanges trade data with the consignee and Freight Forwarder. As a result, data exchange via traditional methods like paper, phone or e-mail is still common despite being inefficient as it requires manual actions (Thomas & Tan, 2015).

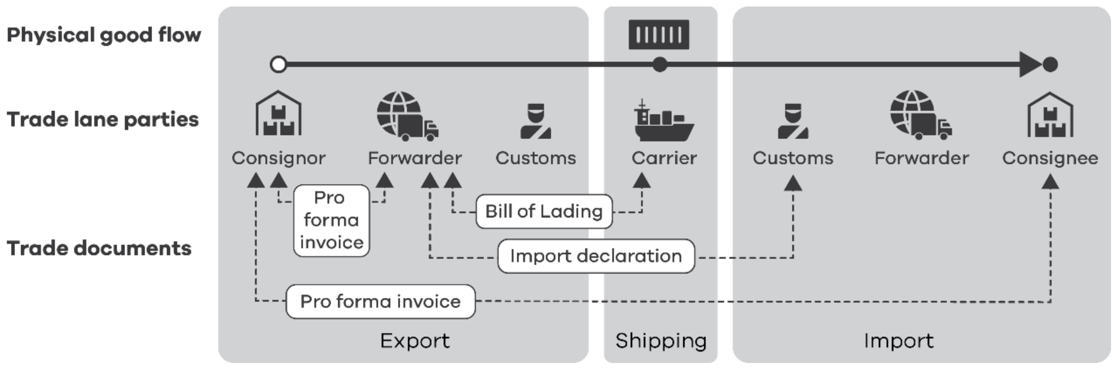


Figure 1: Exchange of some trade documents between trade lane parties without end-to-end visibility.

1.1 Data pipeline concept

Interorganizational information systems that support exchange of trade data by connecting individual data silos of trade lane parties in order to create end-to-end visibility of trade data can lead to more efficient international shipping (Jensen et al., 2018; Knol, Klievink, & Tan, 2014). These systems can be referred to as digital trade infrastructures (Rukanova, Henriksen, Henningsson, & Tan, 2016). A digital trade infrastructure that tries to connect individual data silos of trade lane parties to create end-to-end visibility is the data pipeline concept (Klievink et al., 2012).

The data pipeline connects the individual data silos of all trade lane parties in a trade lane through a virtual bus to create end-to-end visibility (Klievink et al., 2012). It works by storing references to trade data in the pipeline while the actual trade data remains in the individual data silo of a trade lane party. A reference can for example be a Uniform Resource Locator (URL) that refers to a web page that contains a trade document. The benefit is that trade data only has to be exchanged once by making it available to other trade lane parties via a single infrastructure (Hulstijn, et. al, 2012; Klievink et al., 2012). Figure 2 illustrates the data pipeline.

A data pipeline enables customs authorities to what is called 'piggyback' trade data at the source. It refers to the idea that via a data pipeline, customs authorities can retrieve trade data that was not initially intended for customs but can be used for improved risk assessment. Trade data at the source is often of higher quality, as it is used by the data owners themselves (Klievink et al., 2012; Klievink et. al, 2016). For example, the consignor has the most detailed and correct data on a shipment in the form of a pro forma invoice. It knows exactly which goods are shipped. If customs authorities have access to the pro forma invoice, they can cross-validate the document with the import declaration lodged by the declarant (Freight Forwarder). This improves risk assessment and as a result reduces the need for physical container inspections by customs authorities and thus time delay and costs.

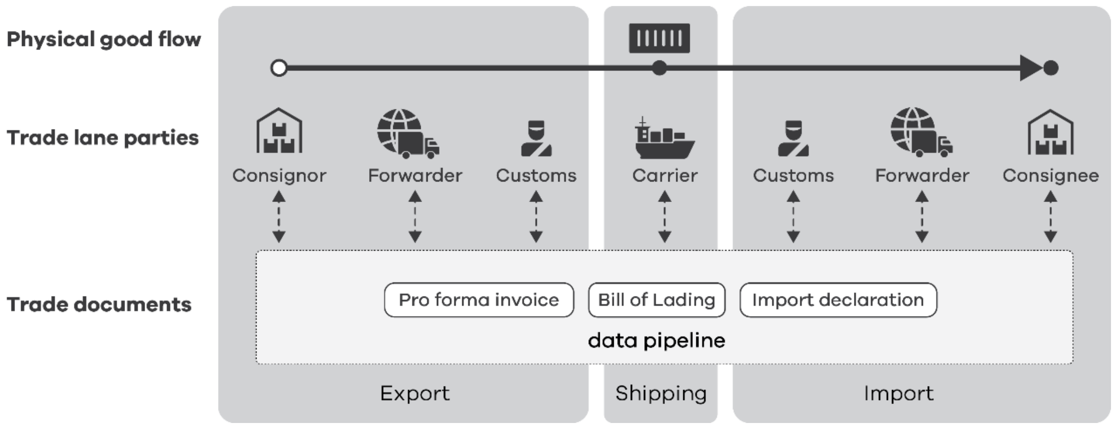


Figure 2: Exchange of some trade documents between trade lane parties based on the data pipeline concept with end-to-end visibility. Figure based on Hesketh [2010].

The data pipeline concept is one of the main outcomes of a large research project funded by the European Union called CORE. Its aim is resolving the issues customs authorities face with regard to risk assessment and validation of correctness of customs declarations. In the EU CORE project, over seventy partners consisting of businesses (e.g., carriers and Freight Forwarders), public authorities (e.g., customs authorities) and research institutions (e.g., Delft University of Technology) work together improving data exchange within trade lanes in order to improve secure transport of goods by better risk assessment and with that to reduce time delay and cost of transport (EU CORE project, 2018).

1.2 Blockchain technology to implement the data pipeline concept

More recently, partners involved in the EU CORE project proposed a new initiative that further extends the potential of the data pipeline concept by using a new technology for the exchange of trade data called: 'Blockchain technology'. This initiative is generally known as the Global Trade Digitization (GTD) platform (Tan, Kouwenhoven, & Buhmann, 2018). The initiative, developed by technology company International Business Machines Corporation (IBM) and container carrier Maersk Line evolved from pilot projects that implemented the data pipeline concept (Jensen, 2017). The GTD platform consists of two main components to exchange trade data: 1) a component to exchange shipping events and 2) a component to exchange trade documents. The vision of the initiative is to create a global data pipeline based on blockchain technology to exchange trade data (Tan et al., 2018). As of 2018, the platform is still in its initiation phase and currently much effort is spent at developing and implementing use cases that could benefit from the platform. The next two sections explore the general idea of blockchain technology and how the data pipeline concept can benefit from the technology.

1.2.1 A brief introduction to blockchain technology

First, a brief introduction to outline the basic concepts of blockchain technology is presented. Blockchain technology, as introduced by Nakamoto (2008) makes it possible for parties that do not (fully) trust each other to exchange data (e.g., digital assets) without the need for a central ledger that keeps track of exchange transactions or is managed by a trusted intermediary to handle these transaction to avoid fraud such as double spending digital assets (Morabito, 2017). The technology is based on cryptography and a peer-to-peer (P2P) network of parties (or nodes) in which a blockchain ledger is distributed (Drescher, 2017). Instead of a central ledger managed by a trusted intermediary, each party in the network holds its own copy of the ledger. Within the ledger, timestamped and digitally signed transactions are stored in cryptographically linked blocks consisting of multiple ordered transactions in which each block contains the cryptographic hash of the previous block, hence the name blockchain ledger (Drescher, 2017; Nakamoto, 2008). The use of cryptography ensures that no party can singlehandedly alter or add new (fraudulent) transactions to the ledger as it would invalidate the ledger. This ensures that once stored, transactions are immutable and non-repudiation by parties involved in the transaction, identifiable by their respective digital signature, is ensured (Viryasitavat, Xu, Zhuming, & Sapsomboon, 2018).

To add a new transaction to the ledger, parties rely on a consensus mechanism in which parties in the network reach agreement on the validity of transactions (Nakamoto, 2008). Only if a majority of the parties agree, a new block of transactions is appended to the blockchain ledger. Within the consensus mechanism, a consensus protocol is responsible for deciding on the order of transactions. Proof-of-Work is the most commonly known consensus protocol. It requires a party to put computational effort into solving a cryptographic puzzle before it can propose a new valid block of ordered transactions. The resulting delay makes it harder to double spend a transaction as it takes time to propose a block. To propose blocks that include double spent transactions faster than other parties, additional computational power is needed (Nakamoto, 2008). All other parties in the network check validity of the block proposal before appending it to their own copy of the blockchain ledger (Nakamoto, 2008). In return, the party that proposed the block will receive a reward. This overcomes the double spend problem as it is more beneficial for a party to act honest than to act malicious (Drescher, 2017). Without consensus, the blockchain ledger cannot be altered, thus rendering transactions (and information included in these transactions) immutable. As a result, information, for example in the form of documents, can be exchanged while at the same time integrity of the exchanged information is guaranteed (Ainsworth & Alwohaibi, 2015).

Since the introduction of blockchain technology, the use of smart contracts has gained widespread attention. Smart contracts are agreements between parties that are stored as code on the blockchain ledger that can be used to digitize assets and automate execution of business processes without involvement of a trusted intermediary (Hamida, Brousmiche, Levard, & Thea, 2017). Transactions contain operations that serve as input to the smart contract. The outcome is stored in the state of the smart contract (Sato & Himura, 2018). Contract code is stored (or deployed) on the blockchain ledger via a transaction. As a result, the code becomes immutable and thus ensures proper execution of processes (Badzar, 2016; Viryasitavat et al., 2018). Each time a party invokes the contract using the same input, the same output will be generated. This enables new ways for parties that do not (fully) trust each other to collaborate (Governatori et al., 2018; Mendling et al., 2017).

1.2.2 The potential of blockchain technology to exchange trade data

Over the last few years, the potential of blockchain technology to enable collaboration between parties that do not (fully) trust each other inside but also outside the financial domain has gained massive attention. Blockchain cannot only be used for financial transactions (e.g., Bitcoins) but can be used to exchange any type of data by storing these exchanges as transactions in the blockchain ledger (Lemieux, 2016; Pilkington, 2015). Especially within the supply chain management[4] domain, blockchain technology is seen as a potential massive disruptor that can improve exchange of supply chain data between parties that has not been possible before (Abeyratne & Monfared, 2016; Apte & Petrovsky, 2016). Global supply chain management systems can be seen as a distributed network of parties. These systems can benefit from blockchain technology, as the technology enables collaboration between parties in a distributed network by securely storing data in transactions (Gao

[4] While the scope of this research is on trade lanes, the limited research on blockchain technology - especially within international shipping – requires a broader perspective on data exchange in supply chains in general to analyze the potential of blockchain technology.

et al., 2018). Blockchain technology can be used to track the provenance of goods in a supply chain (Francisco & Swanson, 2018; Kim & Laskowski, 2018). It also allows for integration of existing supply chain management systems to improve data exchange between parties (Korpela, Hallikas, & Dahlberg, 2017; Milani, García-Bañuelos, & Dumas, 2016).

From a public authority perspective, the potential of blockchain technology has also gained attention. In a white paper, the United Nations Centre For Trade Facilitation and E-business explores the potential of blockchain technology and finds that the technology can improve exchange of trade data and facilitate trade (United Nations Economic and Social Council, 2018). Also, within the European Union, the European Commission's Directorate-General for Taxation and Customs Union (DG TAXUD) sees the potential of blockchain technology to improve efficiency and to reduce administrative costs by improved exchange of trade data (EU TAXUD, 2017). Referring back to the data pipeline concept, blockchain technology enables the exchange of any trade data (e.g., documents) with other parties within a trade lane. If customs authorities would be part of the blockchain network and have a copy of the blockchain ledger, they would also have a copy of all trade data. This creates end-to-end visibility which improves risk assessment.

1.2.3 Challenges blockchain technology

While blockchain is a new and emerging technology that has become a global hype over the last few years with new initiatives presented every day, within this technology a set of challenges that can be a barrier to actual implementation are often experienced (Zheng, Xie, Dai, & Wang, 2016). A prominent challenge relates to the security (e.g., ensuring confidentiality) of data in a blockchain network (Chalaemwongwan & Kurutach, 2018; Lin & Liao, 2017). As each party in the blockchain network has a copy of the blockchain ledger, data stored on the ledger becomes available to all parties. If data is sensitive this causes an issue as sensitive data can be read by all parties. Data security measures, for example allowing only authorized parties to access the data can overcome this issue. However, allowing only authorized parties access also implies a trade-off with the openness of the blockchain network. To allow only authorized parties access, this implies the use of a permissioned blockchain network with some form of access management that is controlled by a central authority. This contradicts the notion that blockchain technology can be used to allow any party to exchange data without a central party in control as the network is open to all parties.

Another prominent challenge relates to scalability of blockchain technology in terms of transaction throughput (Gao et al., 2018; Morabito, 2017). The issue of limited scalability is often linked to the consensus protocol used in blockchain technology to reach consensus on the order of transactions within the network to overcome the double spend problem. Because of the delay that computationally heavy consensus protocols like Proof-of-Work have, the theoretical throughput of transactions is limited. For example, the Bitcoin blockchain has a throughput limited to 8 transactions per second and Ethereum blockchain has a limited throughput of 15 transactions per second. This makes it infeasible to use blockchain technology with Proof-of-Work as consensus protocol for systems that require high volumes of transactions. In permissionless blockchain networks, the consensus mechanism is based on an economic incentive (e.g., via Proof-of-Work) as parties are anonymous and there are no restrictions. In permissioned networks, parties are known as there are restrictions on who can access the blockchain ledger, who can read the ledger and who can participate in the consensus mechanism. As a result, there is no need for an economic incentive, as the incentive is to act honest to avoid reputational damage (Ølnes, Ubacht, & Janssen, 2017). While this would argue for not using Proof-of-Work as no economic incentive is needed but instead using a simpler consensus protocol, permissioned blockchains networks available often still use concepts from permissionless networks such as Proof-of-Work as consensus protocol (Vukolić, 2017). For example, Enterprise Ethereum, a permissioned blockchain solution based on permissionless Ethereum, still uses Proof-of-Work. The underlying reason is that Proof-of-Work is the most commonly known consensus protocol and other protocols are still in development.

This sub section showed that the trade-off between openness versus security and limited scalability in terms of transaction throughput are challenges for implementation of blockchain-based systems. The next section identifies a knowledge gap this research aims to fill.

1.3 Knowledge gap identification

This section identifies a set of knowledge gaps based on the previous section.

1.3.1 Exchange of trade documents using blockchain technology

While the potential of blockchain technology to exchange trade data is known, little is known on how to actually use the technology for the exchange of trade documents. In research on the use of blockchain technology to exchange trade data the focus has predominantly been on the exchange of shipping events (van Engelenburg & Janssen, 2018; van Engelenburg, Janssen, & Klievink, 2017). In addition, there is a lack of knowledge on the use of smart contracts to represent trade documents. Therefore, this research identifies a knowledge gap for the use of blockchain technology to exchange trade documents.

Critical trade document use cases

As international shipping is complex, illustrated by the number of parties involved in an international trade lane and the amount of different (types of) documents these parties exchange (Jensen et al., 2018), the knowledge gap on the use of blockchain technology to exchange trade documents has to be narrowed. Based on the data pipeline concept, the international trade lane analysis of Jensen, Vatrapu and Bjørn-Andersen (2018) which includes export, shipping and import activities, and input from researchers involved in the EU CORE project, several trade documents that are crucial in international shipping, especially from a customs perspective, are identified. Crucial in this context means that the document either contains crucial data customs can use for different purposes (e.g., risk assessment via piggybacking) or is critical in the process of international shipping. For example, any cross-border activity (export or import) is subject to customs procedures that requires customs declarations. If crucial trade documents can be exchanged, this can improve the added value of the data pipeline for both businesses and customs authorities which can lead to higher adoption of such pipeline. The next paragraph discusses three use cases that involve critical trade documents and describes their fit within this research. In the remainder of this research, these use cases are referred to as the: critical trade document use cases.

<u>Use case I: Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration</u>

As discussed in the introduction of this chapter, for manual cross-validation of an import declaration enabling customs authorities to piggyback other trade documents is needed. To that extent, the most crucial trade document customs authorities can piggyback for manual cross-validation of an import declaration is the pro forma invoice. Therefore, the first use case focusses on the exchange of a pro forma invoice using blockchain technology that customs can piggyback for manual cross-validation of an import declaration. More specific, the use case can be described by the following process. All parties in the trade lane are part of the blockchain network and have a copy of the blockchain ledger. A consignor exchanges a pro forma invoice document with other trade lane businesses (e.g., consignee and Freight Forwarder) via a blockchain-based business-to-business exchange to enable end-to-end visibility of the document as all parties have a copy in their own blockchain ledger. A customs authority pulls (or retrieves) the pro forma invoice from this business-to-business exchange via a business-to-government exchange and uses the retrieved document for manual cross-validation of the import declaration (piggybacking principle).

Use case II: Automated generation of an import declaration by aggregating already exchanged pro forma invoice data

Within the EU CORE project, much research focusses on the role of customs authorities in cross-border activities of international shipping. One of the outcomes of this research is that customs authorities need new methods to perform risk assessments and ensure correctness of customs declarations like the import declaration. This need extends beyond the ability to pull business-to-business exchanged documents in order to perform (manual) cross-validation. The huge volume of import declarations makes it impossible to perform manual cross-validation. Only a very limited amount of import declarations can be validated via manual cross-validation. As discussed before, the import declaration consists of aggregated data from several trade documents such as the pro forma invoice or Bill of Lading.

Blockchain technology, specifically smart contracts can reduce the need for manual cross-validation. Trade data and smart contracts stored on the blockchain ledger are immutable. A smart contract could therefore be used to automatically generate an import declaration by aggregating data from the pro forma invoice (or other relevant trade documents) that is stored on the blockchain ledger. This requires pro forma invoice data to be exchanged via a business-to-business exchange beforehand. As the import declaration can only consist of aggregated data stored on the blockchain ledger, customs authorities can be ensured that the data in the import declaration is consistent with other data. As a result, there is no need for manual cross-validation. This improves the declaration process and reduces time and thus costs. The second critical trade document use case can be described as follows: a declarant (e.g. Freight Forwarder) creates an import declaration by invoking a smart contract that automatically retrieves pro forma invoice data from a business-to-business exchange (see use case I) and lodges the import declaration. Customs authorities can pull the import declaration from the business-to-business exchange.

Use case III: Transfer title to the goods using an electronic negotiable Bill of Lading

Most trade-related documents are already exchanged electronically in some form. One trade-related document that has been difficult to exchange electronically is the negotiable Bill of Lading (Dubovec, 2006). A (non-negotiable) Bill of Lading is a transport document issued by the carrier to the consignor as contract of carriage, receipt of goods loaded and title to the goods for shipment by sea (Curwen, 2007). In simple terms, title to the goods means that a party is the legal owner of the physical goods in a shipment. For a straight, non-negotiable Bill of Lading the consignor can exclusively claim title to the goods and oblige the carrier to release the goods. It can therefore be treated like any other document and easily be exchanged electronically as there is no risk of another party successfully claiming title to the goods. In contrast to a (non-negotiable) Bill of Lading, the negotiable Bill of Lading is a special type of transport document as it has an additional functionality: transferability of title to the goods (Hong, 2012). As a result, the consignor cannot exclusively claim title to the goods. Instead, title to the goods can be transferred to another holder. From a legal point-of-view, this means that the holder, who is in physical possession of the negotiable Bill of Lading, is seen as owner and can thus claim title to the goods (Curwen, 2007). The carrier is legally obligated to release the goods to the claiming party upon surrendering the document. Therefore, to avoid fraud with fake copies of the negotiable Bill of Lading uniqueness of the document must be guaranteed. However, guaranteeing uniqueness of an electronic document is extremely difficult as electronic copies can easily be made (Lee, Nguyen, & Pagnoni, 2008). As a result, the negotiable Bill of Lading is still subject to cumbersome paper-based processes with significant time delay and high cost. In the past, there have been several initiatives to implement an electronic negotiable Bill of Lading. Due to the difficulty of fulfilling the requirement of guaranteeing uniqueness these previous initiatives had to rely on complex systems with a centralized title registry to keep track of transfers of ownership and additional legal frameworks to recognize the electronic negotiable Bill of Lading as a legal negotiable Bill of Lading (Dubovec, 2006; Goldby, 2008). Blockchain technology has been proposed as new technology to implement an electronic negotiable Bill of Lading without the use of a centralized title registry or additional legal framework to guarantee uniqueness (Nærland, Müller-Bloch, Beck, & Palmund, 2017; Takahashi, 2016). The underlying reason is that guaranteeing uniqueness looks similar to overcoming the double spend problem in decentralized systems. For example, in cryptocurrency (e.g., Bitcoin) the coin must be unique to avoid spending of the same coin twice (Nakamoto, 2008). While some research conceptualizes the use of blockchain technology for an electronic negotiable Bill of Lading, guidance on how blockchain technology can actually be used to create a solution is lacking (Nærland et al., 2017; Takahashi, 2016). Therefore, the third use case focusses on the transfer of title to the goods of an electronic negotiable Bill of Lading using blockchain technology. The use case can be described as follows: a carrier issues a blockchain-based electronic negotiable Bill of Lading to the consignor. The consignor transfers ownership to a new holder to transfer title to the goods. This holder can re-transfer ownership to a new holder. This process repeats until ownership is transferred back to the carrier when claiming the goods.

In conclusion, the knowledge gap on the exchange of trade documents using blockchain technology is limited to the identified three critical trade document use cases as shown in Table 1. All use cases are further developed in chapter 2.

Table 1: Overview critical trade document use cases.

Use case	Description
I	Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration
II	Automated generation of an import declaration by aggregating already exchanged pro forma invoice data
III	Transfer title to the goods using an electronic negotiable Bill of Lading

This leads to the following knowledge gap: '*the potential of blockchain technology to exchange trade documents is known, but there is a lack of knowledge on how the technology can support such exchange (for the given three critical trade document use cases)*.' To emphasize more on what this knowledge gap entails, the next three sub sections combine the knowledge gap with the identified challenges of blockchain technology.

1.3.2 Openness versus security for exchange of trade documents

The trade-off between openness and security is a common topic for the exchange of trade data. In context of the exchange of trade data using blockchain technology, van Engelenburg and Janssen find a trade-off between accessibility and confidentiality (2018). This can be translated in terms of openness (how accessible is the blockchain ledger) and security (how can the blockchain ledger ensure confidentiality of exchanged data). This shows that addressing this trade-off is relevant to provide guidance on how blockchain technology can support the exchange of trade documents. Therefore, the knowledge gap is extended with addressing the trade-off between openness and security.

1.3.3 Scalability exchange of trade documents

Another challenge of blockchain technology that should be addressed is: scalability of blockchain technology for the exchange of trade data, specifically trade documents. The relevance of addressing the scalability challenge for a blockchain-based system to exchange high volumes of trade documents is also stressed by researchers that are involved in the development of the GTD platform. The exchange of trade documents globally would require a solution that can handle hundreds of millions of exchange transactions per year. For example, within the Netherlands alone around 180 million declarations are processed each year (Customs Administration of the Netherlands, 2017). The Customs Administration of the Netherlands expects this number to grow significantly over the next few years, as mentioned by the Head of Trade Relations: 'In 2017 over 180 million digital entry, exit, import, export and transit line items have been processed by the Dutch customs declaration processes AGS, DMF and GPA. This figure is expected to grow due to e-commerce trade up to 300 million in 2020, not even taking in account any Brexit consequences.' (Heijmann, F., personal quote, February 3rd, 2019). However, current (pilot) blockchain-based systems often use computational-heavy consensus protocols like Proof-of-Work that cannot scale sufficiently in terms of transaction throughput as only between eight to fifteen transactions per second can be handled depending on the implementation. Implementing the critical trade document use cases using such systems would therefore not be feasible under the assumption that each document exchange is at least one transaction. For example, implementing use case II to automate generation of an import declaration would already require a system that can handle around ten transactions per second[5] within the Netherlands alone. Knowledge on how to address the challenge of limited scalability in context of the exchange of trade documents is non-existing. In conclusion, the knowledge gap is extended with addressing the challenge of scalability as a system has to be able to handle hundreds of millions of trade document exchanges per year.

1.3.4 Knowledge gap

The previous two sub sections showed that the challenges regarding the trade-off between openness versus security and scalability should be addressed to find a solution that supports data exchange between trade lane parties. Even further, a trade-off between security versus openness and scalability can also be identified. To improve scalability, in literature it is proposed to use a permissioned blockchain network to control which parties can participate in the consensus mechanism. This enables the use of protocols that have higher transaction throughput, however at the cost of limited scalability in terms of nodes involved (Gramoli, 2017). As discussed in section 1.2.3, this shows a trade-off with the need for openness. Also, less parties participating implies some form of centralized control which can in potential be a trade-off with the requirement for security as the concept of blockchain technology introduces trust in a system that cannot be manipulated by a central authority but relies on consensus in the network (see section 1.2.1).

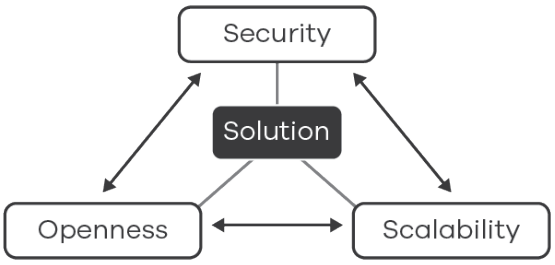


Figure 3: Trade-offs implementation blockchain technology.

Figure 3 depicts the trade-offs between the three challenges. A solution for a blockchain-based system to support exchange of trade documents can be found at the center of all trade-offs. Therefore, this research takes these trade-offs as starting point to analyze how blockchain technology can support exchange. To that extent, the challenges are posed as requirements that should be taken into account. In conclusion, the combined knowledge gap that this research addresses is stated below:

'The potential of blockchain technology to exchange trade documents in trade lanes is known, but there is a lack of knowledge how the technology can support such exchange (for the given three critical trade document use cases) that takes into account the requirements of openness, security and scalability of blockchain technology.'

[5] Computation: 300 million / 365 days / 24 hours / 3600 seconds = 10 transactions per second.

The next section defines a research objective that addresses the selected knowledge gap.

1.4 Research objective

To address the knowledge gap as stated in section 1.3, this section defines a research objective. The knowledge gap shows the need to provide guidance on how blockchain technology can support the exchange of trade documents (illustrated by the three critical trade document use cases), specifically with regard to addressing the requirements of openness, security and scalability of blockchain technology. Architecture design is one of the guidance aspects for digital trade infrastructure development. Use cases I and II are based on the data pipeline concept. A data pipeline can be seen as a type of digital trade infrastructure (Rukanova, Huiden, & Tan, 2017). In digital trade infrastructure research, that includes research on the data pipeline concept, architecture design is often used to describe system elements and their relations within the system and the environment, or problem domain (Rukanova et al., 2016). The International Organization for Standardization (ISO) defines an architecture as: 'the (system) fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.' (ISO, 2011, p. 1). The distributed nature of blockchain technology implies a change in the architecture of the data pipeline from a Service Oriented Architecture with a central bus to a distributed blockchain-based architecture (Smolander, Rossi, & Pekkola, 2017). Thus, an architecture design could provide guidance on how blockchain technology can support use cases I and II that evolve from the data pipeline concept using blockchain technology. Architecture design could also provide guidance on how blockchain technology can support exchange of an electronic negotiable Bill of Lading. Therefore, the research objective - using the research objective template by Wieringa (2014, p. 16) - can be stated as follows:

To design an open, secure and scalable blockchain-based architecture to support exchange of trade documents in trade lanes (as illustrated by the three identified critical trade document use cases).

The next two sub sections discuss the societal and scientific relevance of the research.

1.4.1 Societal relevance

Customs authorities need exchange of trade documents that enables piggybacking of data for better risk assessment and validation of correctness of the import declaration. This reduces tax fraud and improves security of goods entering a customs territory. In addition, there is a need to automate the cross-validation of an import declaration. This research develops a blockchain-based architecture design that supports these exchanges. In the context of the negotiable Bill of Lading, businesses need a shift from paper-based to electronic to reduce time delay, fraud and costs. Blockchain technology has the potential to enable this shift. This research explores how blockchain technology can support the exchange of the electronic negotiable Bill of Lading. Thus, a blockchain-based architecture can potentially provide added value to both customs authorities and businesses as it provides guidance on how blockchain technology can support exchange of trade documents (with regard to the three critical trade document use cases). Therefore, this research is relevant for parties involved in the development of blockchain-based systems that support the exchange of trade documents.

1.4.2 Scientific relevance

Besides societal relevance, the scientific relevance of this research is that it adds to research within the international shipping domain (GTD platform) by providing an architecture that supports exchange of trade documents. More specific, the research contributes to knowledge on the scalability challenge of blockchain technology. Scalability is often mentioned as one of the challenges of implementation of blockchain-based systems, but literature mainly addresses this in general terms and provides generalized solutions. This research focusses specifically on addressing the scalability challenge currently experienced with pilot projects that implement use cases for the exchange of trade documents. Previous research on the use of blockchain technology to exchange trade data, for example Van Engelenburg, Janssen and Klievink (2017) focused on shipping events in general and did not address the specificity of use cases that can affect development of a blockchain-based architecture design. As the research revolves around the critical trade document use cases that either relate to the data pipeline concept or electronic negotiable Bill of Lading concept, the research also contributes to the knowledge base of these concepts.

Departing from the research objective, the next sections derive a main research question (section 1.5) and develop a research approach (section 1.6) that together fulfil the research objective.

1.5 Main research question

By re-formulating the research objective as presented in section 1.4, the main research question (MRQ) of this thesis is derived that translates the challenges to requirements that are to be taken into account for development of a blockchain-based architecture design:

Which design of a blockchain-based architecture can be developed to support exchange of trade documents in trade lanes that takes into account the requirements of openness, security and scalability?

The next section discusses the research approach this thesis adopts in order to answer the MRQ.

1.6 Research approach

In this section, the research approach of this thesis is discussed. The objective of the research is the design of a blockchain-based architecture which categorizes as a design-oriented research. Therefore, this research adopts a Design Science Research (DSR) approach that guides the design of the architecture. DSR is an approach within information systems research with the aim to design and evaluate information technology (IT) artifacts that solve organizational or societal problems (Hevner & Chatterjee, 2010; Hevner, March, Park, & Ram, 2004). These problems can be referred to as the environment (or problem domain) in which the IT artifact is positioned. The DSR approach applies research and environmental knowledge to the design and evaluation of IT artifacts to contribute to existing research knowledge. There is no clear definition of what an IT artifact within DSR should constitute. Often used examples of artifacts are: constructs, models, methods or instantiations (Dresch, Pacheco Lacerda, & Valle Antunes, 2015). Wieringa (2014) introduces the use of architectural structures to describe IT artifacts. This fits closely to the objective of designing a blockchain-based architecture. Therefore, this thesis adopts DSR as research approach.

1.6.1 DSR cycles

Hevner (2007) presents the DSR approach as three cycles: the relevance cycle, the rigor cycle and the design cycle (see Figure 4). These cycles bring together the environment, the knowledge base and design of the IT artifact (Hevner, 2007). The relevance cycle focusses on the application domain. In the rigor cycle, the focus is on the knowledge base to support the design cycle and scientific contribution of the artifact to the knowledge base. The design cycle brings together the environment and knowledge base to design and evaluate the artifact (Hevner et al., 2004).

Within the MRQ, several elements that need to be addressed in order to answer the MRQ can be identified. These elements are: blockchain technology architecture, exchange of trade documents, critical trade documents use cases, and design of the blockchain-based architecture. These elements can be mapped on the environment, knowledge base, design components and relevance, rigor and design cycles of the DSR approach. Figure 4 shows the mapping of the MRQ elements on the DSR approach by Hevner (2007). The critical trade documents use cases concern the environment of the architecture design and the relevance cycle is used to define requirements of the design based on the use cases. It also includes evaluation of the blockchain-based architecture design. The knowledge base draws insights from literature on the exchange of trade documents (which focusses predominantly on the data pipeline and electronic negotiable Bill of Lading concepts) and blockchain technology architecture literature that ground the architecture design in the knowledge base. The design cycle combines the environment and knowledge base to design the blockchain-based architecture.

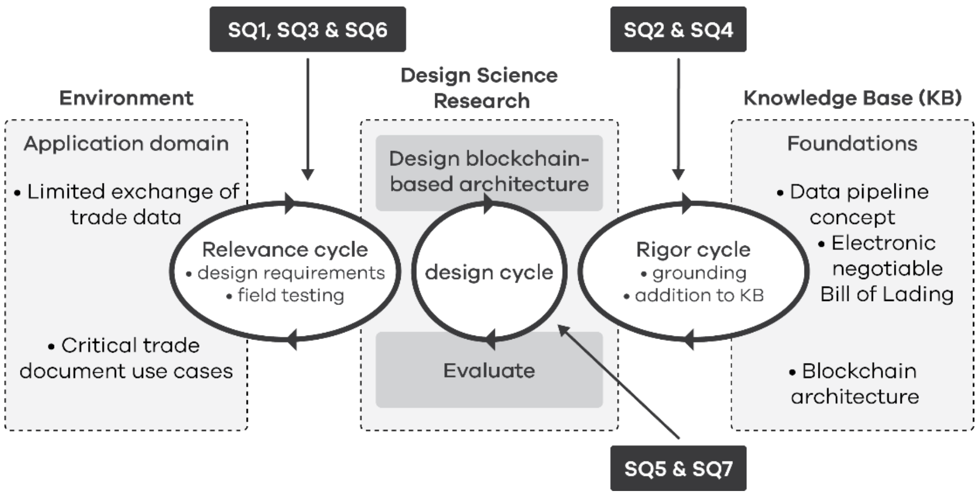


Figure 4: MRQ elements and sub questions research approach fitted on the DSR cycles by Hevner (2007, p. 88).

To answer the MRQ, a set of sub questions that address the identified elements and bring together the environment, knowledge base and design component has to be defined. To structure these sub questions, the research uses a step-wise DSR approach. There is no unambiguous definition of the steps a typical DSR approach follows. To structure the research and provide a logical presentation of the research, the 'Method Framework for Design Science Research (DSR)' as introduced by Johannesson and Perjons (2014) is used. The framework consists of five main steps: 1) explicate problem, 2) define requirements, 3) design and develop artifact, 4) demonstrate artifact and 5) evaluate artifact. The 'explicate problem' step 1 identifies the problem and analyses what the knowledge base on the problem is. The 'define requirements' step 2 departs from the explicate problem step and defines requirements for the artifact. In step 3 'design and develop artifact', the artifact is designed based on the requirements. Step 4 'demonstrate artifact' demonstrates the designed artifact by positioning the artifact in the environment to show it can solve the problem. The final step 5 'evaluate artifact' assesses to what extent the requirements are fulfilled and the actual problem is solved (Johannesson & Perjons, 2014).

The next section defines the sub questions relating to the MRQ elements and research approach in order to answer the MRQ.

1.6.2 Sub questions

To answer the MRQ, a set of sub questions is defined. The following sub questions relate to the MRQ elements and Method Framework for DSR approach defined by Johannesson and Perjons (2014) which is used in this research:

1. What do the three identified critical trade document use cases that can be enabled by blockchain technology look like?
2. What are the design requirements for a blockchain-based architecture design that supports the critical trade document use cases?
3. What are the core blockchain technology architecture components?

4. What does the blockchain-based architecture design look like for each of the critical trade document use cases?
5. How, for demonstration purposes, can the blockchain-based architecture design be used for the critical trade document use cases?
6. To what extent, for evaluation purposes, does the blockchain-based architecture design fit with the design requirements and problem domain of the critical trade document use cases?

Figure 4 maps the six sub questions on the DSR cycles by Hevner (2007). The next section describes the links between the sub questions and the steps in more detail.

1.6.3 Research steps and methods

Whereas the framework by Johannesson and Perjons (2014) has five steps ending with the 'evaluate artifact' step, the research approach is extended with an additional step. The underlying reason is that the 'evaluate artifact' step does not include deriving conclusions and answering the MRQ of this research. Therefore, an additional step is added to the approach to derive conclusions and answer the MRQ as well as reflect on the research.

This section describes each step of the research approach in more detail. Each paragraph describes a step, the sub question that is (or sub questions that are) answered in the step and the methods used to answer the sub questions.

Step 1: Explicate problem
The first step in this research is explication of the problem. This step identifies the research problem and analyses the knowledge base. As this is a broad step, the step is divided into two sub steps. Sub step 1.1 is used for problem introduction. Sub step 1.2 analyses the critical trade document use cases and answers sub question 2. Both sub steps are discussed below.

Sub step 1.1 introduces the problem. The sub step is presented in chapter 1. In the chapter, a knowledge gap regarding the exchange of trade documents (illustrated by the three identified critical trade document use cases) using blockchain technology that addresses the requirements for openness, security and scalability of the technology is identified. From the knowledge gap, a research objective and MRQ are derived.

Next, sub step 1.2 analyses the three identified critical trade document use cases. The sub step answers sub question 2: '*What do the three identified critical trade document use cases that can be enabled by blockchain technology look like?*'. The sub step consists of two elements. First a literature review on the critical trade documents is conducted to identify current issues. Next, the potential use of blockchain technology in context of these use cases is analyzed. The critical trade document use cases serve as input for the definition of design requirements in step 2 and as input for demonstration and evaluation of the blockchain-based architecture design in steps 4 and 5. Chapter 2 develops the use cases.

Step 2: Define requirements
The second step of the research approach is defining requirements (Johannesson & Perjons, 2014). This step defines requirements for the blockchain-based architecture design that enable the critical trade document use cases. Wieringa (2014, p. 51) defines a requirements as: 'a property of the treatment[6] desired by some stakeholder, who has committed resources (time and/or money) to realize the property. In other words, it is a goal for the to-be-designed treatment.'. In this step, sub question 2 is answered: '*What are the design requirements for a blockchain-based architecture design that supports the critical trade document use cases?*'. For each use case, both functional and

[6] Treatment can be read as: artifact.

non-functional design requirements are derived. Functional requirements are derived from the critical trade document use case descriptions in chapter 2. Non-functional requirements are derived from literature. Use cases I and II are closely related to the data pipeline concept as both focus on exchange of documents from which customs authorities (and businesses) can benefit. Therefore, for the non-functional requirements definition of use cases I and II a literature review on the main design principles of the data pipeline concept is conducted. Use case III differs from use cases I and II as the focus is on the transferability of an electronic negotiable Bill of Lading (transferable document) between businesses. Therefore, a literature review on the electronic negotiable Bill of Lading is conducted to define non-functional requirements for use case III. The literature reviews ensure that the blockchain-based architecture design is grounded in the knowledge base of the data pipeline concept and electronic negotiable Bill of Lading. The step is presented in chapter 3.

Step 3: Design and develop artifact
After definition of the requirements, step 3 of the research is design of the actual blockchain-based architecture. This step consists of two sub steps. As the goal is to design a blockchain-based architecture, sub step 3.1 first analyses the knowledge base on blockchain technology architecture components to answer sub question 3: '*What are the core blockchain technology architecture components?'*. A literature review is conducted to identify the core components. This also includes a more technical deep-dive on the functioning of the components. The underlying reason for this technical deep-dive is to make good design choices, especially in context of scalability as it relates to the technical aspects of blockchain technology (e.g., the consensus mechanism). As blockchain technology is a new technology, scientific literature is limited. Therefore, to mitigate the issue of a lack of literature, this research also turns towards grey literature such as white papers, books or technology blog sites on blockchain technology. This reduces the scientific validity but is unavoidable. The identified components serve as basis for the actual design in sub step 3.2.
The step is discussed in chapter 4.

Sub step 3.2 designs the actual blockchain-based architecture for the critical trade document use cases. The sub step answers sub question 4: '*What does the blockchain-based architecture design look like for each of the critical trade document use cases?*'. The identified core blockchain architecture components identified in sub step 3.1 form the basis of the design. Blockchain architecture design taxonomies, for example Xu et al. (2017), are used to guide a creative design process by supporting design choices. Design choices are made based on the defined design requirements in step 2. If required, the architecture design is extended with non-blockchain specific components if the core components of the blockchain technology architecture cannot fulfill the requirements. Chapter 5 describes this step.

Step 4: Demonstrate artifact
The fourth step of the research approach demonstrates the design blockchain-based architecture for each of the critical trade document use cases. The step answers sub question 5: '*How, for demonstration purposes, can the blockchain-based architecture design be used for the critical trade document use cases'*. Using the three use cases as developed in sub step 1.4, the blockchain-based architecture design that supports the use cases is demonstrated to showcase the functioning of the architecture components and how the parties interact with these components. The step is presented in chapter 6.

Step 5: Evaluate artifact
After demonstration, step 5 of the research approach is evaluation of the blockchain-based architecture design to determine the fit with the design requirements of the use cases and the relevance within the problem domain. This step answers sub question 6: '*To what extent, for evaluation purposes, does the blockchain-based architecture design fit with the design requirements and problem domain of the critical trade document use cases*?'. To answer this sub question, two

methods are used. First, the blockchain-based architecture design is mapped on the design requirements of the use cases to evaluate its fit with the defined requirements. Next, expert interviews are used to evaluate the extent to which the use cases are relevant and how the blockchain-based architecture design supports the use cases to assess its effectiveness. This provides insight in the extent to which the problem is solved. Chapter 7 presents the evaluation.

Conclude research
The final part of this research that is outside the DSR approach consists of two sub parts. Sub part 6.1 derives conclusions from the research in order to answer the MRQ: *'Which design of a blockchain-based architecture can be developed to improve data exchange between trade lane parties that takes into account the requirements of openness, security and scalability?'* Next, the societal and scientific contribution towards the identified knowledge gap is discussed. Sub part 6.2 presents a reflection on the research by discussing the research approach to reflect on the choices made, resulting limitations and potential future research. Also, a link between the thesis and the CoSEM programme is made. Conclusion of the research is presented in chapter 8. In the next section, the research approach is visually represented to provide an overview.

1.6.4 Research Flow Diagram

To summarize the research approach visually, Figure 5 presents a Research Flow Diagram. The diagram shows the research steps to be taken, how these steps link to the sub questions, which methods are used to answer the sub questions and the deliverables of the research steps.

1.7 Conclusion chapter 1

In this chapter, a knowledge gap on how a blockchain-based architecture design can support exchange of trade documents (for the given three critical trade document use cases) that takes into account the requirements of openness, security and scalability was identified. The three identified use cases are: 1) Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration, 2) Automated generation of an import declaration by aggregating already exchanged pro forma invoice data and 3) Transfer title to the goods using an electronic negotiable Bill of Lading. Three trade-offs between the challenges of openness versus security, openness versus scalability and security versus scalability were found that need to be addressed for development of the blockchain-based architecture design. To that extent, openness, security and scalability are identified as requirements. The research adopts a Design Science Research (DSR) approach to design a blockchain-based architecture that answers the MRQ: '*Which design of a blockchain-based architecture can be developed to support exchange of trade documents in trade lanes that takes into account the requirements of openness, security and scalability?'*. The Method Framework for DSR by Johannesson and Perjons (2014) is used to structure the research. Chapter 2 continues the problem explication step and focusses on step 1.2 of the research approach that consists of a literature review on the three identified critical trade document use cases and potential of blockchain technology to enable the use cases.

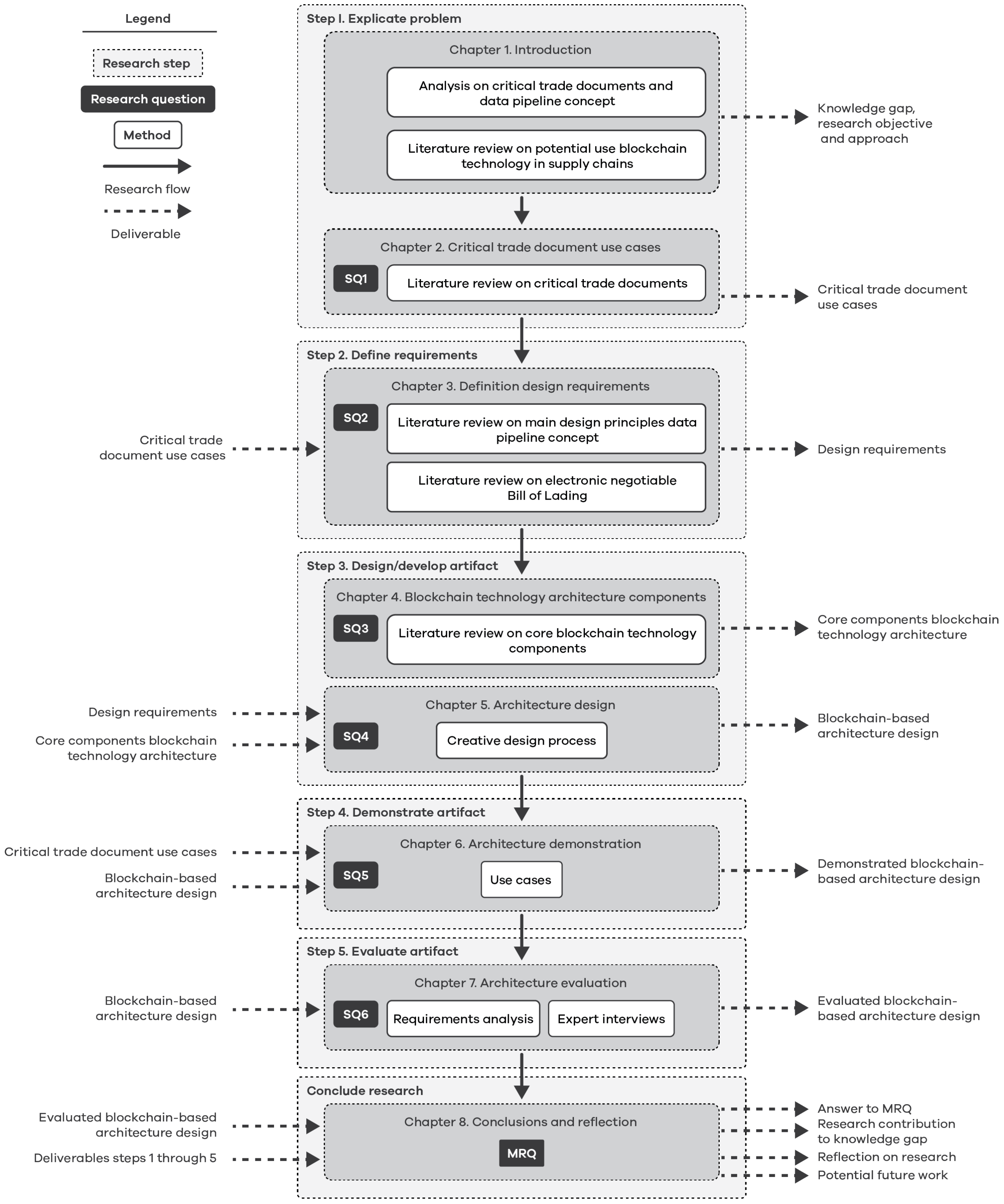

Legend
Research step
Research question
Method
Research flow
Deliverable
Step I. Explicate problem
Chapter 1. Introduction
Analysis on critical trade documents and data pipeline concept
Literature review on potential use blockchain technology in supply chains
Knowledge gap, research objective and approach
Chapter 2. Critical trade document use cases
SQ1
Literature review on critical trade documents
Critical trade document use cases
Step 2. Define requirements
Chapter 3. Definition design requirements
SQ2
Literature review on main design principles data pipeline concept
Literature review on electronic negotiable Bill of Lading
Critical trade document use cases
Design requirements
Step 3. Design/develop artifact
Chapter 4. Blockchain technology architecture components
SQ3
Literature review on core blockchain technology components
Core components blockchain technology architecture
Chapter 5. Architecture design
SQ4
Creative design process
Design requirements
Core components blockchain technology architecture
Blockchain-based architecture design
Step 4. Demonstrate artifact
Chapter 6. Architecture demonstration
SQ5
Use cases
Critical trade document use cases
Blockchain-based architecture design
Demonstrated blockchain-based architecture design
Step 5. Evaluate artifact
Chapter 7. Architecture evaluation
SQ6
Requirements analysis
Expert interviews
Blockchain-based architecture design
Evaluated blockchain-based architecture design
Conclude research
Chapter 8. Conclusions and reflection
MRQ
Evaluated blockchain-based architecture design
Deliverables steps 1 through 5
Answer to MRQ
Research contribution to knowledge gap
Reflection on research
Potential future work

Figure 5: Research Flow Diagram.

2. Critical trade document use cases

In this chapter, sub step 1.2 of the problem explication, that focusses on the three identified critical trade document use cases of this research approach, is presented to answer sub question 1: *'What do the three identified critical trade document use cases that can be enabled by blockchain technology look like?'*. Based on the initial identification of the critical trade document use cases in section 1.3, the three use cases are further developed in this chapter. The selection is repeated briefly in section 2.1 for readability purposes. Next, section 2.2 develops use case I. Section 2.3 develops use case II. Use case III is developed in section 2.4. Finally, section 2.5 concludes this chapter by answering the sub question. The developed critical trade document use cases serve as input for the definition of design requirements in chapter 3, demonstration of the blockchain-based architecture design that implements the use cases in chapter 6 and as input for the expert interviews to determine the relevance of the use cases (see chapter 7).

2.1 Selected crucial trade documents use cases

In the knowledge gap this research addresses (see section 1.3), the exchange of three critical trade documents was discussed. An overview of the use cases is provided in Table 2. For all use cases, the scope is limited to shipment in a trade lane consisting of a consignor, freight forwarder, carrier, consignee and customs authority.

Table 2: Overview use cases.

Use case	Description
I	Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration
II	Automated generation of an import declaration by aggregating already exchanged pro forma invoice data
III	Transfer title to the goods using an electronic negotiable Bill of Lading

Sections 2.2, 2.3 and 2.4 present each of the use cases. For each use case, the problem at hand is discussed to show the relevance of the use case and how blockchain technology can in potential be used to implement the use cases is discussed.

2.2 Use case I: Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration

In international shipping customs procedures play a central role in the shipment of goods. All cross-border events are subject to both customs export and import procedures (Martincus, Carballo, & Graziano, 2015). The goals of these procedures are to regulate import and export of goods for security purposes and government tax revenue via customs import and export duties[7]. Physical shipment inspections are part of these procedures. However, due to the high volume of shipments, not every shipment can be physically inspected. Therefore, customs authorities rely on data-based risk assessment to determine which shipments are subject to physical inspection.

[7] Custom duties serve two functions: 1) provide a source of government tax revenue, 2) protection against unfair competition. Customs value is the primary determinant of customs duties. The value is calculated using customs valuation (see Box 1). In this case, unfair competition means that the inner market of a customs territory is harmed by the import of cheaper goods from a market with lower production cost (Han & McGauran, 2014).

2.2.1 Using the import declaration for risk assessment

For customs authorities, knowing the exact details of the goods in a shipment is essential for risk assessment and correct customs valuation (see Box 1) to derive custom duties (Triepels et al., 2018). Therefore, for each shipment a declarant[8] is required to lodge an import declaration before a shipment enters a customs territory. Import declarations are either lodged electronically or on paper and should specify: information on consignor and consignee, date of issue, description of goods, country of origin, goods classification (Harmonized Systems Code), invoice value, duties and taxes, terms of delivery (Incoterm) and means of transport (European Commission, 2018).

2.2.2 Issues with the import declaration

While customs authorities require exact details of the goods for effective risk assessment and correct customs valuation, these details are often not available. The data provided on the import declaration is often vague or inaccurate (Hesketh, 2010; WCO, 2005). One reason is that the import declaration contains aggregated data from multiple documents like the pro forma invoice or Bill of Lading. For example, in case of a Bill of Lading, the carrier does often not know exactly which goods are shipped. To avoid liability for the goods, vague terms like 'said to contain' are used. As a result, the import declaration also contains vague terms. In addition, the consignor and consignee specified on the declaration are often not the true consignor and consignee (Hesketh, 2010). Due to the complexity of international shipping, Freight Forwarders are used to manage required paperwork. Often, a Freight Forwarder declares itself as consignor and consignee, making it harder to determine the true consignor and consignee.

Due to vague or inaccurate data, cross-border tax and duty fraud via falsified documentation is a growing problem in international trade (Triepels et al., 2018). Falsification is the manipulation of data elements in trade documents for financial gain (Feelders, & Daniels, 2015). Document fraud in international trade can cost nations billions annually. A common type of document fraud is the undervaluation of the transaction value on import of goods to reduce due custom duties. Box 1 provides some background on customs valuation.

Box 1: Customs valuation basics.

Customs valuation

This box introduces customs valuation in more detail. The World Trade Organization (WTO) defines customs valuation as: "a customs procedure applied to determine the customs value of imported goods. If the rate of duty is ad valorem, the customs value is essential to determine the duty to be paid on an imported good." (WTO, n.d.). Ad valorem means that the duty is calculated as a percentage (rate) of the dutiable value of the goods (Forrester & Odarda, 2005). Six methods for customs valuation exist: 1) transaction value method, 2) the value of identical goods, 3) the value of similar goods, 4) the deductive value, 5) the computed value and 6) the fallback method (Wolffgang & Dallimore, 2013). A declarant cannot freely select any customs valuation method. Which method for customs valuation is to be used, is determined by a fixed order. The transaction value method is the first method in order and thus primary valuation method. Only in case this method is not applicable, a different method may be used. As a result, the transaction value is the most prominent customs valuation method used (Wolffgang & Dallimore, 2013). Therefore, this research only discusses the transaction value method.

[8] In the use case, the assumption is made that a Freight Forwarder acting on behalf of either consignor or consignee acts as declarant or customs agent for a shipment and is thus responsible for lodging an import declaration.

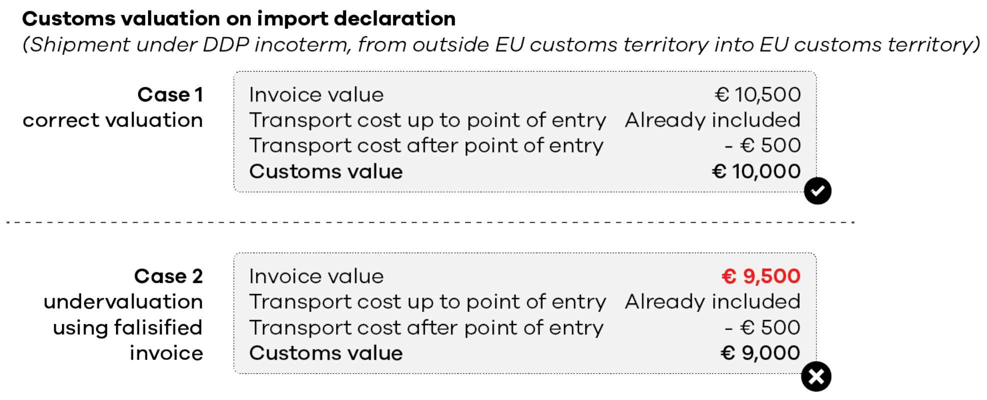
Transaction value method
According to the WTO, the transaction value method to calculate the customs value uses the following rule: "The price actually paid or payable is the total payment made or to be made by the consignee to or for the benefit of the consignor for the imported goods, and includes all payments made as a condition of sale of the imported goods by the consignee to the consignor, or by the consignee to a third party to satisfy an obligation of the consignor." (WTO, n.d.). The price actually paid or payable means that the customs value is based on the entire contract price (or invoice value). Depending on the shipment and terms of delivery adjustments to this value can be made (Forrester & Odarda, 2005). These adjustments are either additions or deductions. Examples of additions are packing cost or license fees paid by the consignee. Deductions can be costs accruing after importation, such as transport cost between the port of entry and the shipment's final destination (WCO, 2005). Terms of delivery, or, are an important determinant for the addition or subtraction of fees and transport cost to the transaction value (Single Window for Logistics Luxembourg, 2017). Within international shipping, Incoterms® are a set standardized and internationally recognized rules to specify the terms of delivery of a shipment (Ramberg, 2011). The next paragraph discusses Incoterms in general.

Incoterms® rules
Incoterms® rules define the responsibilities of both consignor and consignees in sales contracts for the delivery of goods. The rules specify the terms of delivery. Who is responsible for payment of custom duties and taxes, insurance or carriage charges are typical rules to specify these terms (ICC, 2010; Ramberg, 2011). The rules also define at which point ownership of the goods is transferred and thus liability for loss or damages of the goods shifts between businesses. In the most recent Incoterms® 2010, a set of eleven rules are defined that each assign different responsibilities to consignor and consignee. Consignor and consignee negotiate which Incoterm applies to a shipment. The Incoterm is added as required field on the (pro forma) invoice (UPS, 2014). Two popular Incoterms, Delivery Duty Paid (DDP) and Free On Board (FOB), are briefly introduced.

DDP is a popular Incoterm that makes the consignor responsible for all risk and costs associated with delivery of the goods to the place of destination. For example, the consignor must pay any customs export or import duties. The International Chamber of Commerce (ICC) defines DDP as follows: "*The seller bears all the costs and risks involved in bringing the goods to the place of destination and has an obligation to clear the goods not only for export but also for import, to pay any duty for both export and import and to carry out all customs formalities."* (ICC, 2010).

Another often used Incoterm is FOB. FOB makes the consignor responsible for delivery up until the 'critical point', which is the loading of a shipment onto a vessel selected and paid for by the consignee. This is de defined by the ICC as: *"The risk of loss of or damage to the goods passes when the goods are on board the vessel, and the buyer bears all costs from that moment onwards."* (ICC, 2010).

Undervaluation
Custom duties are determined based on the customs value calculated using the transaction value method. As duties are calculated ad valorem, a fraudulent party can lower the customs value to reduce payment of duties. One way of lowering the customs value is by undervaluation of the goods as specified on a pro forma invoice.



Customs valuation on import declaration

(Shipment under DDP incoterm, from outside EU customs territory into EU customs territory)

Case 1
correct valuation

Invoice value	€ 10,500
Transport cost up to point of entry	Already included
Transport cost after point of entry	- € 500
Customs value	**€ 10,000** ✓

Case 2
undervaluation using falisified invoice

Invoice value	**€ 9,500**
Transport cost up to point of entry	Already included
Transport cost after point of entry	- € 500
Customs value	**€ 9,000** ✗

Figure 6: Example correct customs valuation and undervaluation using falsified invoice.

In the example of Figure 6 the customs value of a shipment under DDP from outside the EU customs territory into the EU customs territory is calculated. The point of entry can be for example the Port of Rotterdam. In the first case, the customs value is correctly calculated. The invoice value is used as starting point. Under DDP all transport costs up to final delivery are included into the invoice value. However, customs valuation is limited to the point of entry into the customs territory. Therefore, costs that are made after entry (and importation) may be deducted. The final value is the customs value. In the second case, a falsified invoice is used in which the goods are undervalued. While the correct invoice is used for payment between consignor and consignee, the falsified invoice is used to determine the customs value. If a customs authority would request the invoice for further investigation, the falsified invoice is provided. Thus, to make undervaluation more difficult, customs authorities need more detailed and accurate data to validate the import declaration.

2.2.3 Use of pro forma invoice for manual cross-validation

One approach customs authorities take to validate the import declaration is to cross-validate it using other documents. As the import declaration contains aggregated data from other documents, any mismatch between for example specified consignor and consignee can be an indicator for fraud. If the customs authority thus has access to these other documents, it can perform cross-validation to validate correctness of the import declaration.

Documents further upstream in the trade lane provides customs authorities with more details on the shipment and can help overcome the issue of vague or inaccurate data. This can reduce fraud and thus reduce lost tax revenue. The pro forma invoice, generated by the consignor, is one of these upstream data sources. In a trade lane, the consignor is the principal provider of data regarding a shipment. It knows exactly which goods are packed and under which sales terms. These terms - the agreement on the purchase and delivery of goods between consignor and consignee like the invoice value - are specified in a commercial invoice. However, in international shipping a commercial invoice is often not generated and sent to the consignee the moment a shipment is sent. Instead, the commercial invoice is generated and sent to the consignee upon final delivery, when all costs (relating to transport and customs duties) are known.

Therefore, a different type of invoice, called pro forma invoice exists. A pro forma invoice acts as a temporary measure before a commercial invoice is prepared. It represents the starting point of shipment of goods. If goods are shipped a pro forma invoice can be sent to notify the consignee of the shipment or provide a Freight Forwarder with relevant data for handling other paperwork such as lodging an import declaration. The pro forma invoice can thus be seen as a preliminary invoice or type of transport document (U.S. Customs and Border Protection, 2017). The invoice contains at least the following data elements according to the European Commission (2018): information on consignor and consignee, date of issue, invoice number, description of goods, unit of measure, quantity of goods, goods classification, unit value, total item value, total invoice value, terms of payment, terms of delivery (or Incoterm) and means of transport. Thus, while the pro forma invoice is only a preliminary invoice and is not used for final payment, it still contains all relevant data elements required for manual-cross validation. For sake of simplicity, in this research manual-cross validation only considers the following data elements: true consignor and consignee, Incoterm, invoice value and (transport) cost after importation. Thus, the exchange of a pro forma invoice can play an important role in providing customs authorities with detailed and accurate data on a shipment that can be used for cross-validation. Figure 7 depicts how customs authorities can manually cross-validate the consignor, consignee and customs valuation of the import declaration and pro forma invoice. Each corresponding data element from both documents is checked for consistency. If all elements are consistent between both documents, the import declaration is deemed correct.

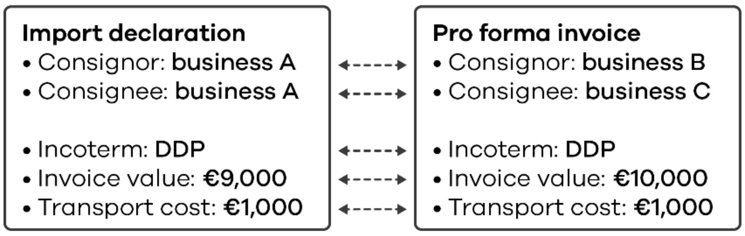


Figure 7: Cross-validation of data elements import declaration and pro forma invoice with mismatch in data elements.

2.2.4 Exchange of a pro forma invoice

In the current situation, customs authorities often request the consignor (or representing Freight Forwarder) to exchange the pro forma invoice when the customs authority identifies irregularities in a lodged import declaration and cross-validation is required. To that extent, the pro forma invoice has to be sent to the customs authority. Figure 8 provides a BPMN to depict the process. This causes delay and also does not ensure that the provided pro forma invoice is the original. The data pipeline concept as introduced in section 1.1 can overcome this issue by providing a business-to-business exchange that enables end-to-end visibility of the pro forma invoice. It allows customs authorities to pull the document via a business-to-government exchange and piggyback the pro forma invoice to cross-validate the import declaration. The next sub section explores how blockchain technology can enable this exchange.

2.2.5 Exchange of a pro forma invoice using blockchain technology

In this sub section, enabling the exchange of a pro forma invoice using blockchain technology that customs authority can use to piggyback data is explored. The exchange follows the piggybacking idea of the data pipeline concept, which is also part of the 2016 Union Customs Code (UCC) that aims at improving data exchange between businesses and authorities within the European Union (European Parliament & European Council, 2013). All parties in the trade lane, both businesses and customs authority, are part of the blockchain network and have a copy of the blockchain ledger. A consignor exchanges a pro forma invoice by appending it to a transaction. The transaction is added to the blockchain ledger. As the blockchain ledger is immutable, customs authorities are ensured that the exchanged pro forma invoice cannot be altered after being exchanged (Ainsworth & Alwohaibi, 2015) (Botton, 2018). This makes the pro forma invoice usable for cross-validation. The customs authority can simply pull the pro forma invoice from its own copy of the blockchain ledger which enables business-to-government exchange. Figure 9 provides a BPMN of the use case.

This section developed use case I. The next section develops use case II.

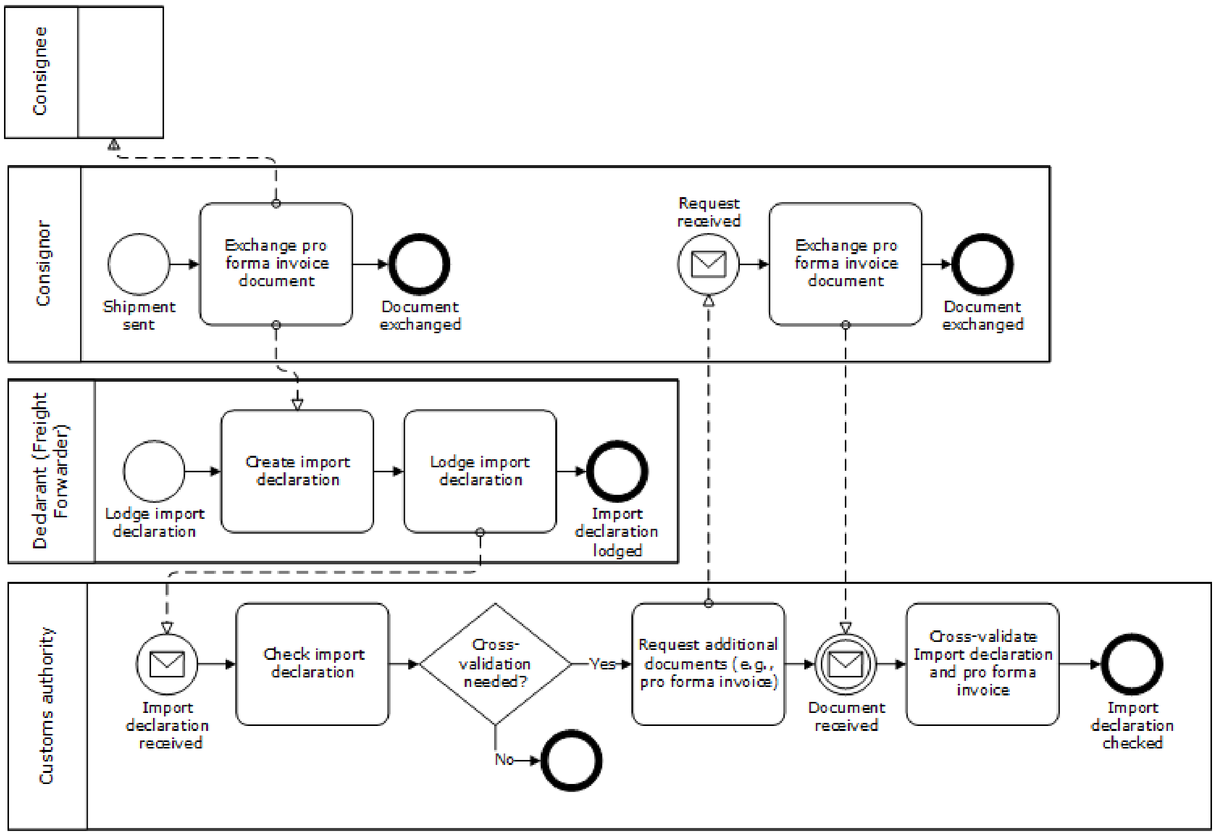

Consignee
Consignor
Shipment sent
Exchange pro forma invoice document
Document exchanged
Request received
Exchange pro forma invoice document
Document exchanged
Declarant (Freight Forwarder)
Lodge import declaration
Create import declaration
Lodge import declaration
Import declaration lodged
Customs authority
Import declaration received
Check import declaration
Cross-validation needed?
Yes
No
Request additional documents (e.g., pro forma invoice)
Document received
Cross-validate Import declaration and pro forma invoice
Import declaration checked

Figure 8 BPMN exchange of pro forma invoice for manual cross-validation.

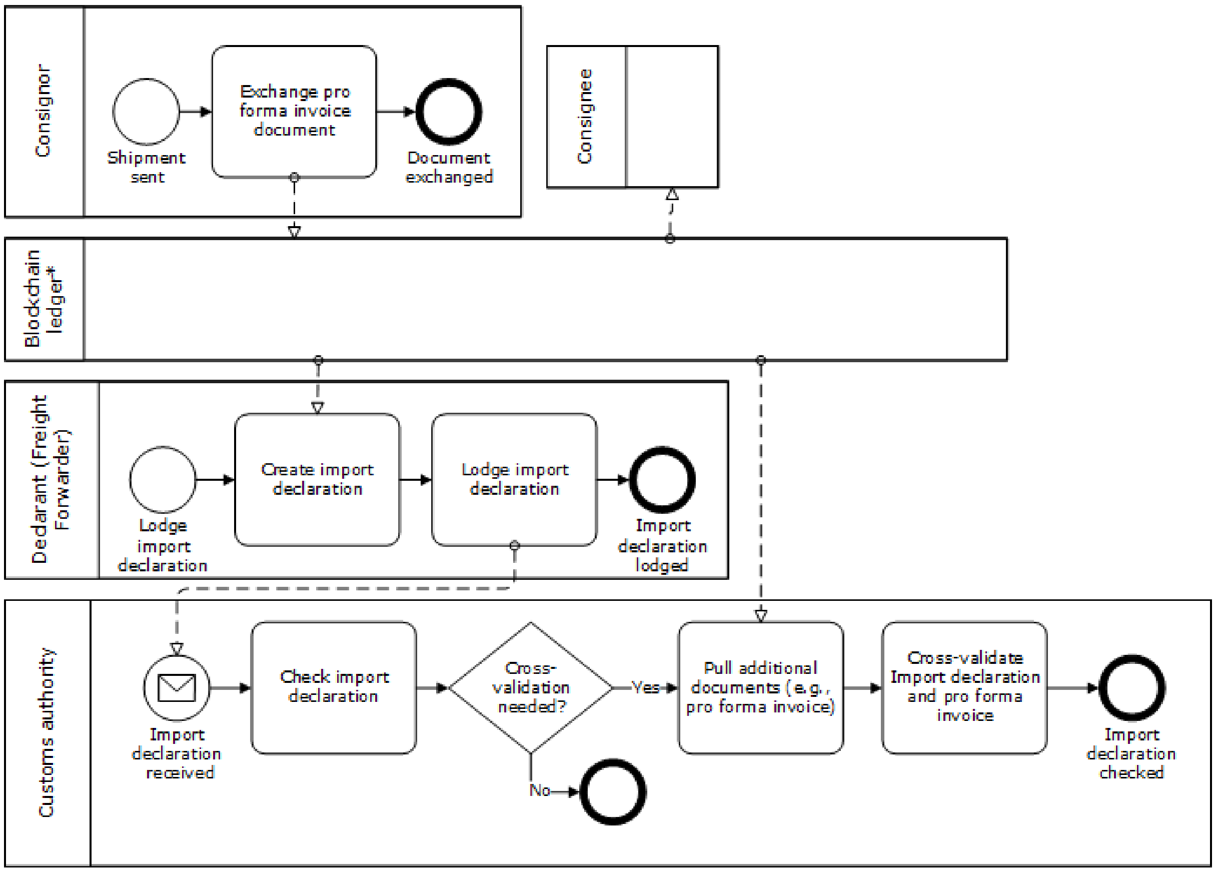

Consignor
Shipment sent
Exchange pro forma invoice document
Document exchanged
Consignee
Blockchain ledger*
Declarant (Freight Forwarder)
Lodge import declaration
Create import declaration
Lodge import declaration
Import declaration lodged
Customs authority
Import declaration received
Check import declaration
Cross-validation needed?
Yes
No
Pull additional documents (e.g., pro forma invoice)
Cross-validate Import declaration and pro forma invoice
Import declaration checked

Figure 9: BPMN exchange of pro forma invoice supported by blockchain technology.

2.3 Use case II: Automated generation of an import declaration by aggregating already exchanged pro forma invoice data

The use of blockchain technology can enable piggybacking of a pro forma invoice for manual cross-validation of the import declaration by a customs authority, as was explored previously in use case I. While this provides an improvement compared to the current situation, the use case still has some shortcomings. Manual cross-validation requires more personnel, which leads to higher costs. Also, customs authorities have to first identify irregularities in the import declaration, then pull the document from their copy of the blockchain ledger and then manually check correctness of the import declaration (e.g. if customs valuation is correct). As mentioned in section 1.3.3, given the fact that within the Netherlands alone around a 180 million declarations are processed each year with numbers expected to rise to at least 300 million in the coming years, manual cross-validation cannot be used for each import declaration (Customs Administration of the Netherlands, 2017). In addition, to lodge an import declaration a declarant (e.g., Freight Forwarder) has to manually generate the import declaration using data from other documents like the pro forma invoice. There is thus a need for fast and efficient processing of import declarations that shortens the lead time of shipments which reduces costs (Jiang, Aldewereld, Dignum, Wang, & Baida, 2015). To reduce time-consuming manual cross-validation, automation is therefore needed (Weigand & Bukhsh, 2011). Use case II explores automation of the import declaration and can therefore be seen as an extension of use case I.

2.3.1 Aggregating exchanged pro forma invoice data to automatically generate an import declaration

Recall the fact that the import declaration is based on aggregated data of several other trade documents of which the pro forma invoice is the most crucial document. As a result, for manual cross-validation data elements from both documents can be compared. For example, the invoice value as specified on the import declaration should match with the invoice value specified on the pro forma invoice. If data elements match, data integrity between the documents can be proved.

As the import declaration is based on aggregated data, the process of lodging an import declaration and performing manual cross-validation can be automated and simplified. Instead of a declarant manually generating an import declaration based on other documents and customs authorities manually cross-validating the import declaration, the import declaration could be generated automatically by aggregating relevant pro forma invoice data elements from a business-to-business exchange. For example, the customs valuation can be calculated automatically via a simple algorithm if the invoice value, Incoterm and transport cost are already exchanged (see customs valuation in Box 1). For example, if a shipment is shipped under DDP Incoterm, the customs value can be calculated via customs value = invoice value – transport cost. As a result, there is no need for manual cross-validation as already exchanged pro forma invoice data is used. This however requires pro forma invoice data elements to be exchanged beforehand via a business-to-business exchange. Use case I enables this exchange under the assumption that in this use case individual data elements rather than a pro forma invoice document are exchanged. This would require individual data elements to be submitted directly into the system, for example as electronic invoice. Challenges regarding data harmonization and standardization are left outside the scope of this research as this use case is more explorative of nature.

However, automatically generating an import declaration does not ensure data integrity per se and thus would still require manual cross-validation. For customs authorities there should be a guarantee that the pro forma invoice data elements cannot be altered during generation of the import declaration. Ensuring this within an electronic environment is difficult. Over the last few years, customs authorities have made efforts to ensure data integrity by focusing on system-based control. System-based control relies on the auditing of information systems used by trade lane parties to determine trustworthiness of data (Weigand & Bukhsh, 2011). The guarantee of immutability of data within blockchain technology offers a solution to the data integrity problem and can simplify auditing. In addition, smart contracts used within blockchain technology have been proposed as means for the automation of business processes as contract code is deployed on the blockchain ledger and is therefore also immutable (Viryasitavat et al., 2018). This ensures that the same input always renders the same output. The next sub section therefore explores how blockchain technology could be used to automatically generate an import declaration by aggregating exchanged pro forma invoice data.

2.3.2 Automatically generate an import declaration using blockchain technology

In this sub section, the use of blockchain technology to automate the generation of an import declaration by aggregating pro forma invoice data is explored. The declarant (e.g., Freight Forwarder) and customs authority both have a copy of the blockchain ledger. This ledger contains the pro forma invoice data elements for a specific shipment. The declarant invokes a smart contract to automatically generate an import declaration for a specific shipment. The contract retrieves relevant pro forma invoice data elements from the blockchain ledger. Next based on implemented business logic in the smart contract, the import declaration is generated. The final import declaration is added as transaction to the blockchain ledger. The customs authority can easily access its own copy of the blockchain ledger to pull the import declaration. The BPMN of Figure 10 depicts this process. This use case thus enables automatic generation of an import declaration by aggregating already exchanged pro forma invoice data. This takes away the need for manual generation of the import declaration as well as manual cross-validation as the generated document can only be based on exchanged data.

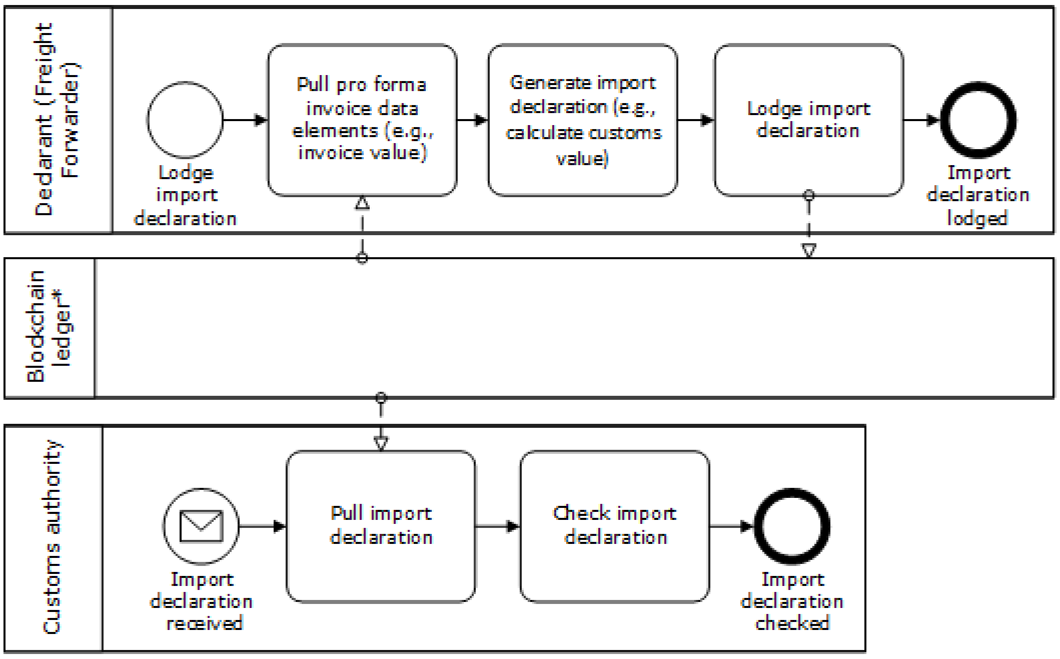

Dedarant (Freight Forwarder)
Lodge import declaration
Pull pro forma invoice data elements (e.g., invoice value)
Generate import declaration (e.g., calculate customs value)
Lodge import declaration
Import declaration lodged
Blockchain ledger*
Customs authority
Import declaration received
Pull import declaration
Check import declaration
Import declaration checked

Figure 10: BPMN automated generation of import declaration using blockchain technology.
* Each party involved holds its own copy of the blockchain ledger.

This section developed use case II. The next section develops use case III.

2.4 Use case III: Transfer title to the goods using an electronic negotiable Bill of Lading

Within international shipping by sea, consignors and consignees rely on carriers for physical transport of the shipment. To realize this transport, for centuries the Bill of Lading has been an important transport document (Sparka, 2010). The Bill of Lading is an agreement between the consignor of the goods and the carrier for the transport of goods to the designated consignee in the port of discharge (Dubovec, 2006). It therefore serves three purposes: 1) evidence of contract carriage, 2) receipt for goods loaded unto the carrier vessel and 3) title to the goods (Curwen, 2007; Van Boom, 1997). After goods are loaded onto the carrier vessel, the carrier issues the Bill of Lading to the consignor of the goods. The consignor sends the Bill of Lading to the consignee. The consignee can then use the Bill of Lading as title to the goods in order to claim the goods from the carrier in the port of discharge (Pagnoni & Visconti, 2010).

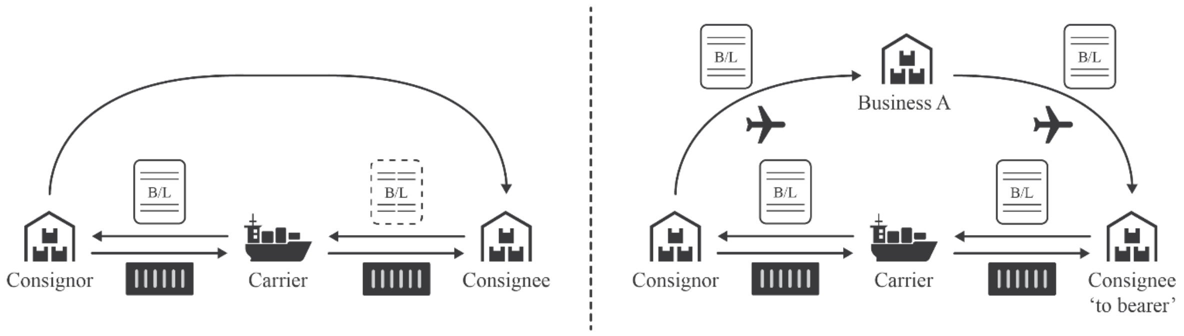


Figure 11: General process non-negotiable (left) versus negotiable Bill of Lading (right).

Two types of Bill of Ladings can be distinguished: 1) non-negotiable and 2) negotiable. Figure 11 depicts the difference in process between a non-negotiable and negotiable Bill of Lading. A non-negotiable Bill of Lading specifies a designated consignee that can exclusively claim title to the goods. Therefore, the consignee does not need to surrender the Bill of Lading physically (Dubovec, 2006). In contrast to a non-negotiable Bill of Lading, the negotiable Bill of Lading does not specify a designated consignee. Instead, it specifies the consignee as 'to the bearer'. From a legal perspective, this means that any party that physically presents the document to the carrier can claim title to the goods. A carrier is obligated to release the goods to the party (to the bearer) that surrenders the negotiable Bill of Lading. This makes the negotiable Bill of Lading transferable between parties (Dubovec, 2006). By physically transferring the document a party legally transfers title to the goods to the new holder. The added benefit is that it allows for the selling (and reselling) of goods in transit by simply transferring the document. The BPMN in Figure 12 shows the general process of a negotiable Bill of Lading.

The transferability of a negotiable Bill of Lading implies that there must be a guarantee that the document is unique (Takahashi, 2016). In context of a negotiable Bill of Lading, uniqueness means that at any given time only one party (holder) can be in possession of the original document. If multiple copies of the negotiable Bill of Lading would exist, a falsified negotiable Bill of Lading could be used to claim title to the goods in order to receive the goods. Given the challenge of guaranteeing uniqueness, this use case therefore focusses on the transfer of title to the goods using a negotiable Bill of Lading.

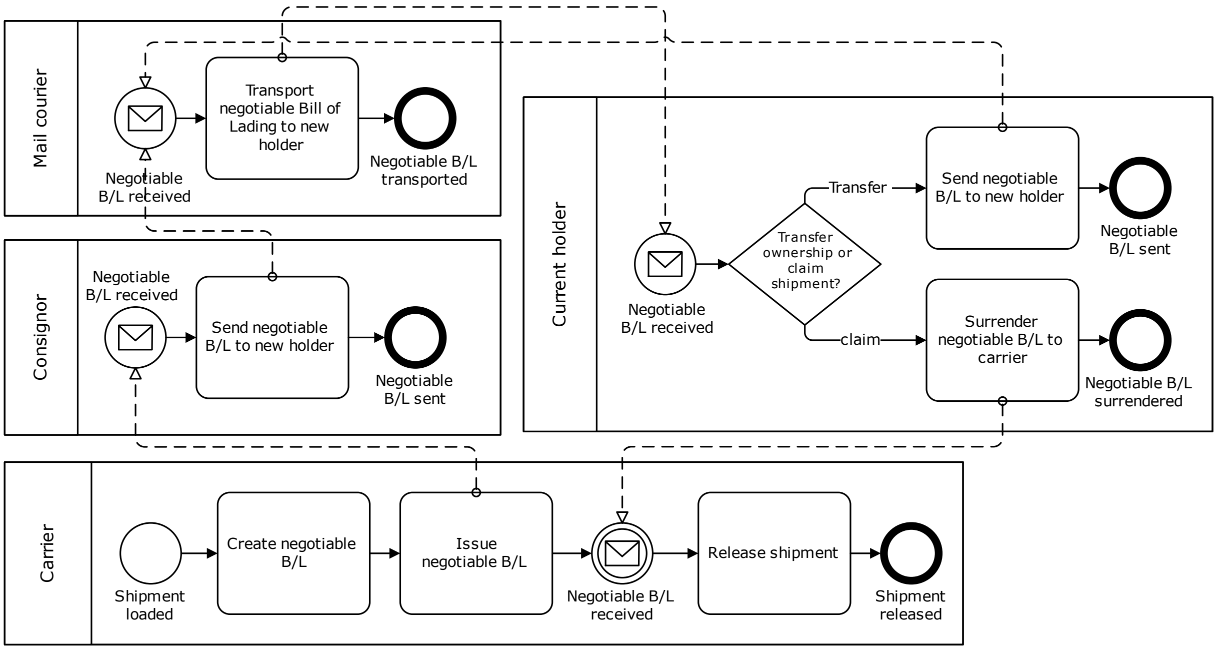


Figure 12: BPMN current negotiable Bill of Lading process.

2.4.1 Issues regarding use of negotiable Bill of Lading

Currently, most negotiable Bill of Ladings are paper-based. Mail couriers are used to exchange the negotiable Bill of Lading (Wu, Starr, & Tan, 2017). A negotiable Bill of Lading exchanged via courier can take up to ten days to reach the new holder. If the claiming party is not in possession of the physical document, it cannot successfully claim the goods from the carrier. This causes delays in the shipping process when the claiming party wants to claim the goods from the carrier at the port of discharge. Second, the issuing and exchange of paper-based negotiable Bill of Ladings is expensive. A single negotiable Bill of Lading can cost over €100 (Lee et al., 2008). Besides delay and costs, a lack of security causes problems related to fraud. A paper-based negotiable Bill of Lading is easily lost, stolen or forged. Uniqueness can therefore not easily be guaranteed.

2.4.2 Use of electronic negotiable Bill of Lading

To overcome the issues of delay, high costs and fraud of paper-based negotiable Bills of Lading, electronic versions of the document can potentially be used as no physical exchange takes place (Van Boom, 1997). Information technology to electronically exchange documents used in international shipping has reduced the need for paper documents over the years. Technologies like Electronic Data Interchanges (EDIs) have supported electronic exchange for many types of documents like invoices and import declarations (Karan, 2011). However, the exchange of an electronic negotiable Bill of Lading has proved difficult (Wu et al., 2017; Dubovec, 2006; Karan, 2011).

The underlying reason is that an electronic negotiable Bill of Lading is not simply an electronic version of a paper-based negotiable Bill of Lading due to the requirement of guaranteeing uniqueness whilst being transferable. Within electronic environments, guaranteeing uniqueness of a document (or record) is difficult to realize. Electronic documents can easily be copied or manipulated (Takahashi, 2016). This relates to the double spend problem. Therefore, for many years the idea was that an electronic negotiable Bill of Lading could only be implemented if some centralized system that keeps track of who is the current holder of the electronic negotiable Bill of Lading was used.

Electronic negotiable Bill of Lading initiatives

Currently, three initiatives that guarantee uniqueness of an electronic negotiable Bill of Lading by using a centralized system exist: Bolero, essDOCS and e-title™. These initiatives all rely on a central 'title registry' that is managed by the system provider itself. The central title registry is used to store the current holder of the electronic negotiable Bill of Lading in records (Wu et al., 2017). A record in the title registry contains a reference to the negotiable Bill of Lading document and the current holder. The actual document is stored by the owner. No documents are stored in the title registry. Only the current holder can send a signed message to transfer ownership to a new holder.

The adoption of the electronic negotiable Bill of Lading initiatives is currently limited (Goldby, 2008). Several reasons for limited adoption can be identified. First, the initiatives focus on a specific use case: the transfer of tittle to the goods of using an electronic negotiable Bill of Lading. Their integration into other systems is limited which limits perceived benefit and thus adoption by trade parties. A second reason relates to the legal aspect of an electronic negotiable Bill of Lading (Takahashi, 2016). Current international maritime law, such as The Hague-Visby Rules, do not mention the use of electronic negotiable Bill of Lading as functional equivalent of a paper-based negotiable Bill of Lading. While this does not prohibit the use of an electronic negotiable Bill of Lading, uncertainty amongst businesses regarding the legal validity of an electronic negotiable Bill of Lading rises (Mcdermott, Nagle, Horowitz, Johnson, & Nagle, 2017). To overcome this uncertainty, current initiatives rely on additional legal agreements to assure businesses that the central title registry to record the current holder of the electronic negotiable Bill of Lading acts as the legal functional equivalent of a paper-based negotiable Bill of Lading. For example, Bolero uses a Rule Book that sets the rules for agreement between parties and the system provider. Each party involved in a shipment has to be registered and agree with the rules from the Rule Book (Bolero International Ltd., 1999). Similar to Bolero, e-Title also requires all parties to register and agree with a set of rules. Parties register at the Electronic Title User Group and sign an Electronic Title User Agreement before they can use the e-title™ system (e-title, n.d.). These legal agreements thus ensure that registered parties treat the electronic negotiable Bill of Lading as the functional equivalent of a paper-based negotiable Bill of Lading.

However, the use of a centralized title registry in combination with registration and signing of additional legal agreements can be a barrier for adoption. The underlying reason is increased bureaucracy and a lack of system reliability and transparency (Goldby, 2008; Pagnoni & Visconti, 2010; Takahashi, 2016). There is thus a need for a solution to support an electronic negotiable Bill of Lading without the use of a centralized title registry to keep track of the current holder whilst still guaranteeing uniqueness.

2.4.3 Transfer title to the goods with an electronic negotiable Bill of Lading using blockchain technology

Over the last two years several initiatives that use blockchain technology to develop electronic negotiable Bills of Lading without a centrally managed title registry have gained attention. The general idea is that blockchain technology can guarantee uniqueness without the use of a centrally managed title registry as it solves the double spend problem. Parties use transactions to transfer title to the goods to a new holder, as depicted in Figure 13. In addition, the use of blockchain technology can reduce time and costs significantly compared to paper-based negotiable Bill of Ladings (CargoX, 2018). A quick scan using the search terms ‘blockchain’ and ‘bill of lading’ leads to two blockchain-based initiatives that focus specifically at the electronic negotiable Bill of Lading use case: CargoX and Wave. Other initiatives focus on a broader set of trade-related use cases, including the electronic negotiable Bill of Lading. The Global Trade Digitization (GTD) platform as introduced in section 1.2 is one of these initiatives. Documentation on CargoX and Wave is limited. Knowledge on CargoX is derived from the CargoX Whitepaper (CargoX, 2018). For the Wave initiative, knowledge is derived from multiple news articles (Rizzo, 2015). CargoX relies on the use of smart contracts on the Ethereum blockchain platform to represent electronic negotiable Bills of Lading as smart contracts. Wave uses blockchain technology to create a distributed title registry of which all parties maintain a copy.

While some blockchain-based initiatives exist, implementation and adoption are in an early stage and uncertainties related to legal applicability and scalability in terms of transaction throughput are present. For example, Wave states that the uncertainty on the legal impact of blockchain technology results has led to a design that uses blockchain solely for managing title to the goods via a distributed title registry (Rizzo, 2015). A CargoX representative[9] verbally communicated that the currently used Proof-of-Work consensus protocol may result in scalability issues in terms of transaction throughput as the system grows in size. The uncertainties experienced by the blockchain-based initiatives make the use case in this research relevant to further explore.

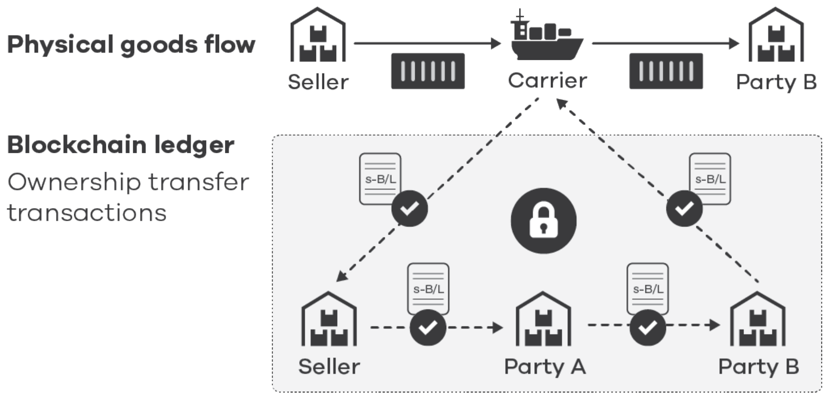


Figure 13: General process of transfer of title to the goods with an electronic negotiable Bill of Lading using blockchain technology.

In the remainder of this sub section, the transfer of title to the goods with an electronic negotiable Bill of Lading using blockchain technology is explored. All businesses involved in the process of an electronic negotiable Bill of Lading, including carrier, consignor and previous and current holder (any business) are part of the blockchain network and have a copy of the blockchain ledger. A carrier issues a new electronic negotiable Bill of Lading, which includes specification of the holder (consignor), by appending it to a transaction. The transaction is added to the blockchain ledger. As the electronic negotiable Bill of Lading as well as the current holder are stored on the blockchain ledger, uniqueness of the document is guaranteed. Only one copy can exist and only the party specified as holder controls further action because transactions have to be digitally signed[10] by the holder in order to be valid. The current holder can either choose to claim title to the goods at the carrier by proving it is the current holder as stored on the blockchain ledger, or transfer title to the goods to a new holder. In the BPMN of Figure 14 the process of an electronic negotiable Bill of Lading is conceptualized.

This section developed use case III. The next section concludes this chapter and answers the sub question.

[9] Igor Jakomin, Chief Business Development Officer of CargoX at Blockchain Workshop organized by ALICE in Barcelona, Spain on May 9th 2018.

[10] The concept of digital signatures is explained in detail in Box 2.

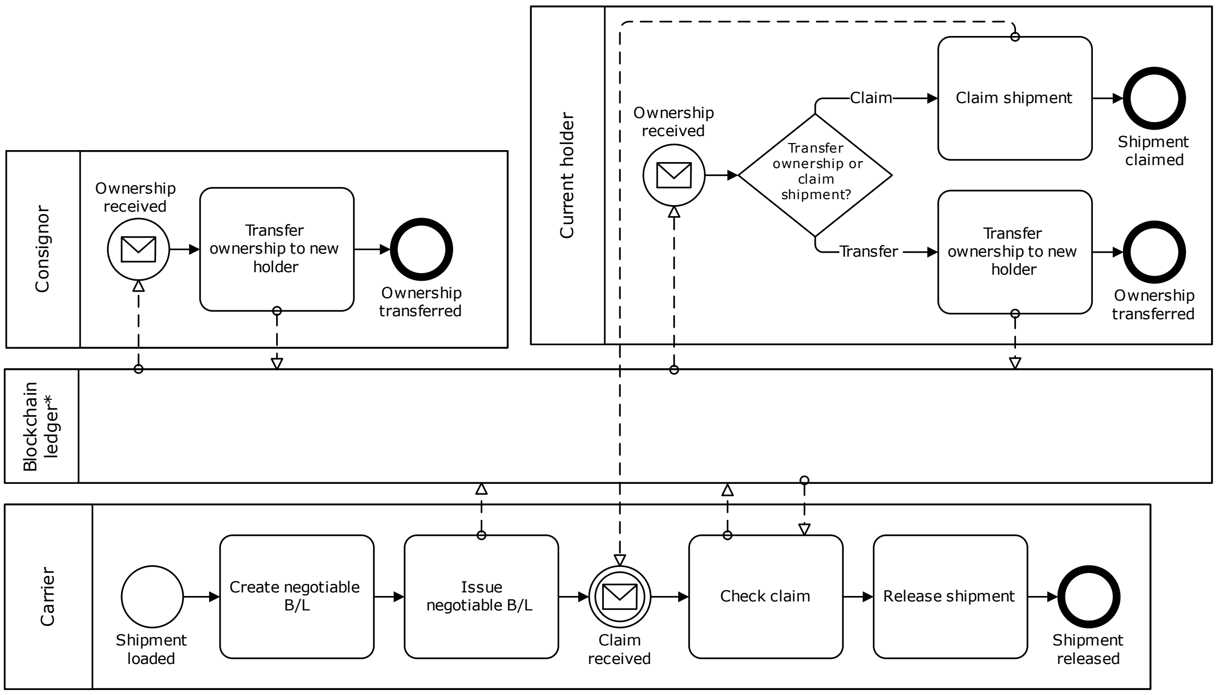


Figure 14: BPMN Blockchain-based electronic negotiable Bill of Lading (B/L) process.
* Each party involved has its own copy of the blockchain ledger.

2.5 Conclusion chapter 2

Based on the analysis of the critical trade document use cases, sub question 1 can be answered: *'What do the three identified critical trade document use cases that can be enabled by blockchain technology look like?'*. Three use cases that can be enabled by blockchain technology were developed: 1) exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration, 2) automated generation of an import declaration by aggregating already exchanged pro forma invoice data and 3) transfer title to the goods using an electronic negotiable Bill of Lading.

Use case I finds that blockchain technology can be used for the business-to-business exchange of a pro forma invoice by storing the invoice on the blockchain ledger. It also enables business-to-government exchanges as a customs authority can use its own copy of the blockchain ledger to pull the pro forma invoice and piggyback the document for manual cross-validation purposes.

Use case II finds that blockchain technology can be used to automatically generate an import declaration by aggregating exchanged pro forma invoice data as enabled by use case I. Pro forma invoice data is stored on the blockchain ledger. Using smart contracts, data can be retrieved to automatically generate an import declaration. This takes away the need for manual-cross validation.

Use case III finds that blockchain technology can be used to transfer title to the goods of an electronic negotiable Bill of Lading. The immutability of the blockchain ledger can guarantee uniqueness of the electronic negotiable Bill of Lading as it overcomes the double spend problem.

In the next chapter, use case specific design requirements for the blockchain-based architecture design are defined based on the findings of this chapter.

3. Definition design requirements

Chapter 2 explicated the exchange of trade documents by describing three critical trade document use cases that can potentially be enabled by blockchain technology. This chapter defines design requirements for the blockchain-based architecture design that supports the use cases and answers sub question 2:*' What are the design requirements for a blockchain-based architecture design that supports the critical trade document use cases?'.* Drawing on the findings of chapter 2 and literature on the data pipeline concept regarding data security and governance as well as literature on the electronic negotiable Bill of Lading a set of requirements for the blockchain-based architecture per use case is defined.

Design requirements do not only prescribe the functionalities an IT artifact, but also other aspects like security constraints, performance or reliability (Ramachandran & Mahmood, 2017). Therefore, for each use case two types of requirements are defined: functional and non-functional. Functional requirements are: 'Statements regarding the services which the system should provide, how the system should react to particular inputs and how the system should behave in particular situations' (Ramachandran & Mahmood, 2017, p. 6). In other words, what the system is ought to do. Non-functional requirements are: 'Constraints related to the services or functions offered by the system, such as timing constraints, security issues, constraints on the development process and so on' (Ramachandran & Mahmood, 2017, p. 6). Non-functional requirements specify under what constraints the system should function.

First, section 3.1 presents the derived design requirements of use case I. Second, section 3.2 presents the design requirements of use case II. Third, in section 3.3 the design requirements of use case III are presented. Finally, section 3.4 presents an overview of the design requirements to answer the sub question and conclude this chapter. The design requirements are used as input for the design of the blockchain-based architecture in chapter 5 and evaluation of the architecture in chapter 7.

3.1 Design requirements use case I

In this section, the design requirements of use case I are defined. First, sub section 3.1.1 defines functional requirements. Next, sub section 3.1.2 defines non-functional requirements. Sub section 3.1.3 presents a sub conclusion for the design requirements of use case I.

3.1.1 Functional requirements

In use case I, the goal is to enable customs authorities to piggyback an exchanged pro forma invoice document[11]. To that extent, the document first has to be exchanged between businesses (via a business-to-business exchange) in the trade lane in a way that enables end-to-end visibility within the trade lane. End-to-end visibility is required as the consignor that exchanges the document is positioned upstream from the customs authority in the trade lane. With end-to-end visibility customs can pull the pro forma invoice more easily than having to rely on the consignor to push the document and use it for piggybacking (Hesketh, 2009, 2010; Jensen et al., 2018; Klievink et al., 2012; Overbeek, Klievink, Hesketh, Heijmann, & Tan, 2011). Therefore, the first functional requirement is defined as follows:

UC1-req. 1: The architecture should provide business-to-business exchange of a pro forma invoice document to enable end-to-end visibility

To enable piggybacking, a customs authority has to be able to pull the document from the business-to-business exchange via business-to-government exchange (Jensen et al., 2018; Klievink et al.,

[11] The format of the pro forma invoice document is not specified but could for example be a regular .PDF document or electronic document in .JSON format.

2012; Overbeek et al., 2011). Therefore, the second functional requirement of use case I is defined as follows:

UC1-req. 2: The architecture should provide business-to-government exchange to enable customs authorities to piggyback an exchanged document for manual cross-validation of an import declaration

In the next sub section, the non-functional requirements of use case I are defined.

3.1.2 Non-functional requirements

The business-to-business and business to-government exchange to enable end-to-end visibility of a pro forma invoice and enable the pull of the document for piggybacking is not straightforward as was identified in the knowledge gap identification (see section 1.3) . Use case I is closely related to the base case of the data pipeline concept which is the exchange of a pro forma invoice document that customs authorities can pull and use to piggyback for manual cross-validation of the import declaration (Hesketh, 2010; Jensen et al., 2018; Overbeek et al., 2011). Therefore, for the definition of non-functional requirements for a blockchain-based architecture that supports use case I literature on the data pipeline concept is used. Within literature on the data pipeline concept, a major research topic relates to the willingness of businesses to exchange commercially sensitive data via a data pipeline (Hulstijn et al., 2012; Knol et al., 2014; Pruksasri, Van Den Berg, Hofman, & Daskapan, 2013; Thomas & Tan, 2015; van Engelenburg, Janssen, & Klievink, 2017). A pro forma invoice contains sensitive data, for example on the consignor and consignee involved or the value of the goods. Other parties may use the pro forma invoice to gain a competitive advantage. A consignor will therefore only exchange the document if confidentiality is guaranteed so that only authorized parties can access the document. Thus, one important driver for the exchange of sensitive data like trade documents can be defined as the willingness of businesses to exchange (Fawcett, Osterhaus, Magnan, Brau, & McCarter, 2007).

Data security is one of the basic requirements of the willingness to exchange (Jarman & Luna-Reyes, 2016). It usually focuses on three key items: 1) confidentiality, 2) integrity and 3) availability, also known as the CIA Triad (Johnson 2010). Another basic requirement relates to the governance of a data pipeline as the architecture should be open to businesses willing to exchange (Klievink et al., 2016; Klievink & Lucassen, 2013; Thomas & Tan, 2015; van Engelenburg, Janssen, Klievink, & Tan, 2017). In addition, part of the knowledge gap this research addresses is the challenge of scalability research on blockchain-based initiatives, like the Global Trade Digitization Platform that aims at improving data exchange, currently experience with regard to the volume of shipping events and documents that can be handled. As the use cases involve the exchange of a pro forma invoice document, sufficient scalability is a relevant requirement to consider. In conclusion, the elements discussed above show the relevance of addressing data security, architecture governance and scalability of the blockchain-based architecture to exchange a pro forma invoice. To that extent, three requirements are defined regarding confidentiality and integrity of the exchanged pro forma invoice, and governance of the architecture related to the openness of the architecture. In addition, a requirement regarding scalability of the architecture is defined.

<u>Confidentiality of document and businesses involved</u>

The pro forma invoice document contains sensitive data. Therefore, the consignor is not willing to openly share the invoice (Fawcett et al., 2007; Urciuoli, Hintsa, & Ahokas, 2013; van Engelenburg et al., 2015; van Engelenburg, Janssen, & Klievink, 2017). Only parties that are involved in a shipment should therefore be authorized to access the document (Pruksasri, Berg, Hofman, & Tan, 2014; Pruksasri et al., 2013). In addition, carriers see a potential risk in end-to-end visibility of for example a pro forma invoice. Currently, detailed data on a shipment, as specified in a pro forma invoice, is not available to carriers. They are therefore only limited liable for the shipment (van Engelenburg et al., 2015). However, if the carrier would have access to the pro forma invoice, it could become fully liable in case of damages or loss according to the Hague-Visby Rules (Klievink et al., 2012; E. Lee, 2012; van Engelenburg, Janssen, & Klievink, 2017). This could lead to insurance and legal claims at the cost of the carrier. A carrier should therefore be restricted from access to the pro forma invoice (Knol et al., 2014). Besides carriers, Freight Forwarders also see a potential risk in the exchange of a pro forma invoice with end-to-end visibility. The pro forma invoice contains sensitive data, especially on the consignor and consignee. A Freight Forwarder fears that it will be bypassed by other parties like the carrier that can offer its services directly to the consignor if its identity is known (Knol et al., 2014). Thus, the pro forma invoice should not be accessible by businesses downstream in the trade lane, like the carrier, to overcome the issue of bypassing the Freight Forwarder.

In conclusion, access to the pro forma invoice and the identity of the parties involved in a shipment should be restricted and only be accessible to authorized parties. Figure 15 provides an overview of which parties in a trade lane should be authorized to access the pro forma invoice document within the architecture. In the use case, the shipment is shipped under DDP Incoterm and the import declaration is lodged by the Freight Forwarder at export side on behalf of the consignor. Thus, only the consignor, consignee, Freight Forwarder at export side and customs authorities at export and import side should have access to the exchanged document[12]. By combining the need for confidentiality of both the pro forma invoice document and identities of the businesses involved in the exchange of the document, the following requirement is defined:

UC1-req. 3: The architecture should ensure confidentiality of both the exchanged document and identities of businesses involved in the exchange transaction and should allow only authorized parties access to the document or identity to overcome legality and liability issues

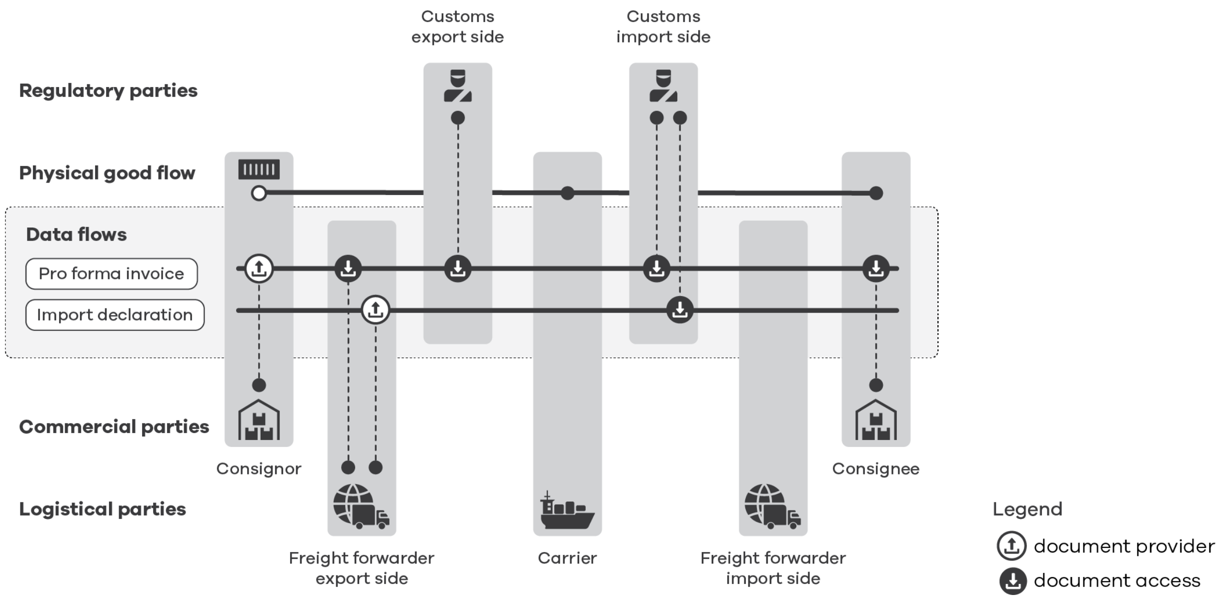


Figure 15: Pro forma invoice and import declaration document flows and access authorization.

<u>Integrity of exchange of documents</u>
To enable piggybacking of a pro forma invoice document, customs authorities have to be ensured that the document pulled via a business-to-government exchange from the business-to-business exchange is the original document. In other words, customs authorities have to trust that the exchanged document is reliable (Hulstijn et al., 2012). In that context, integrity of the document has to be ensured to avoid the possibility of a malicious consignor making alterations to the document after the document is exchanged (Pruksasri et al., 2013). As ensuring integrity of the exchanged

[12] CORE FloraHolland Deliverables D12.11, D12.12, D12.13 and D12.14 provide further details on this process.

document is crucial for customs authorities in order to enable piggybacking, the following requirement is defined:

UC1-req. 4: The architecture should ensure integrity of the exchanged document

Architecture governance
The previous two non-functional requirements can be seen as 'general' data security requirements for the exchange of commercially sensitive data. A business-to-business and business-to-government exchange with end-to-end visibility of a pro forma invoice, following the data pipeline concept, requires some special consideration. While restricting access is required to ensure confidentiality of the pro forma invoice (see UC1-req. 3), this can potentially conflict with the general concept of the data pipeline: open access to businesses that are willing to exchange trade data (e.g., documents) using the architecture (Knol et al., 2014). As a balance between restricted access and openness has to be found, the architecture being open to businesses to participate is required (Knol et al., 2014). In addition, the concept of blockchain technology is based on creating an open and transparent network in which parties can exchange any type of data without a central authority (or trusted intermediary) in control. Van Engelenburg and Janssen (2018) address the challenge of storing trade data in a blockchain-based architecture. They find a trade-off between data security and openness. The openness blockchain technology offers regarding transparency and decentralization of the authority in control is the major benefit of using the technology in context of exchange of trade data and should therefore be leveraged as much as possible.

Thomas & Tan (2015) find that an architecture based on the data pipeline concept should avoid 'Big Brother' issues. This means that no central authority can be in control of a central database (e.g., a document storage) as it could be a barrier for the willingness of businesses to exchange commercially sensitive data. If a central authority would be in control of a central database, this could conflict with the confidentiality and integrity requirements as discussed in requirements UC1-req. 3 and 4. In addition, it could hamper (authorized) parties from accessing the pro forma invoice document. An architecture to exchange trade data thus cannot be controlled by central authority, whether being a single business or government agency (Klievink et al., 2016). Regarding storage of trade data within a data pipeline architecture, two main types can be distinguished: 1) a thick and 2) a thin data pipeline (Van Engelenburg, Janssen, Klievink, et al., 2017). Van Engelenburg et al. (van Engelenburg, Janssen, Klievink, et al., 2017) find that an architecture to exchange only sensitive documents, like a pro forma invoice document, can best be achieved by using a thin data pipeline. A thin data pipeline means that only a reference to the document is exchanged. The document itself is stored in the data storage of the owner of the document. Therefore, the architecture should not have a centralized document storage and should only be used to exchange metadata of the document such as a document reference. By combining the elements for openness, decentralization of storage and decentralization of the authorities in control, the following requirements is defined:

UC1-req. 5: The architecture should be open to businesses, without the use of a centralized document storage and without a central authority in control

Scalability
As mentioned in section 1.3.3, the volume of declarations within the Netherlands alone is expected to rise up to 300 million per year over the next few years. If the customs authority wants to cross-validate all import declarations using a pro forma invoice, this would require an architecture that can handle at least the exchange of 300 million pro forma invoices per year. This would mean that the architecture should be able to handle 300.000.000 [import declarations] / (365 [days] x 24 [hours] x 3600 [seconds]) ≈ 10 pro forma invoice document exchanges per second for the Netherlands alone. An architecture that also needs to handle documents outside the Netherlands, for example within the EU customs territory, should be able to handle a multitude of ten transactions per second. Current

blockchain-based initiatives experience limited scalability in terms of transaction throughput as the often used Proof-of-Work consensus protocol can only handle between eight and fifteen transactions per second and has a delay of sixty minutes to confirm transactions (Chalaemwongwan & Kurutach, 2018; Narayanan, Bonneau, Felten, Miller, & Goldfeder, 2016). This architecture can therefore not rely on Proof-of-Work as its use will be limited by the scalability issue. Alternatives to overcome this issue have to be considered. Therefore, the need for scalability in terms of transaction throughput is included explicitly as design requirement and is defined as follows:

UC1-req. 6: The architecture should be able to handle high volumes of document exchanges in real-time

3.1.3 Sub-conclusion design requirements use case I

In this section, a set of six design requirements for the blockchain-based architecture that supports use case I was defined based on the use case description in chapter 2 and literature on the data pipeline concept. Table 3 provides an overview of the defined requirements, specifying code, definition and related sources. The next section defines design requirements of use case II.

Table 3: Overview of defined design requirements use case I.

Design requirements use case I: Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration		
Code	*Definition*	*Related sources*
Functional		
UC1-req. 1	The architecture should provide business-to-business exchange of (a pro forma invoice) document to enable end-to-end visibility	(Hesketh, 2010), (Klievink et al., 2012), (Overbeek et al., 2011), (Hesketh, 2009), (Jensen et al., 2018), Explicated via use case I description
UC1-req. 2	The architecture should provide business-to-government exchange to enable customs authorities to piggyback an exchanged document for manual cross-validation of an import declaration	(Overbeek et al., 2011), (Klievink et al., 2012), (Jensen et al., 2018), Explicated via use case I description
Non-functional		
UC1-req. 3	The architecture should ensure confidentiality of both the exchanged document and identities of businesses involved in the exchange transaction and should allow only authorized parties access to the document or identity to overcome legality and liability issues	(Pruksasri et al., 2013), (Knol et al., 2014), (van Engelenburg et al., 2015), (van Engelenburg, Janssen, & Klievink, 2017), (Pruksasri et al., 2014), (Hofman & Bastiaansen, 2014), (Jarman & Luna-Reyes, 2016), (Fawcett et al., 2007), (Urciuoli et al., 2013), (Klievink et al., 2012), (E. Lee, 2012)
UC1-req. 4	The architecture should ensure integrity of the exchanged document	(Hulstijn et al., 2012), (Pruksasri et al., 2013)
UC1-req. 5	The architecture should be open to businesses, without the use of a centralized document storage and without a central authority in control	(Thomas & Tan, 2015), (Klievink et al., 2016), (van Engelenburg & Janssen, 2018), (Knol et al., 2014), (Van Engelenburg, Janssen, Klievink, et al., 2017)
UC1-req. 6	The architecture should be able to handle high volumes of document exchanges in real-time	Explicated via Use case I description and introduction, (Hofman & Bastiaansen, 2014)

3.2 Design requirements use case II

This section defines requirements for the blockchain-based architecture to support use case II. As this use case is an extension of use case I, the assumption is made that use case I is already in place and pro forma invoice data is exchanged. The focus of the requirements for the architecture that supports use case II is therefore on the automatic retrieval of pro forma invoice data and calculation of the customs value to generate an import declaration.

3.2.1 Functional requirements

From the use case description in section 2.3 it follows that the architecture should enable a declarant to lodge an import declaration that is automatically generated by aggregating pro forma invoice data like consignor, consignee, invoice value, Incoterm and transport cost to calculate the customs value. A customs authority should be able to retrieve the generated import declaration. Therefore, the functional requirement of use case II is defined as follows:

UC2-req. 1: The architecture should enable a declarant to automatically generate an import declaration by retrieving already business-to-business exchanged pro forma invoice data and calculate customs value, and allow business-to-government exchange of the generated import declaration

The next sub section defines the non-functional requirements of use case II.

3.2.2 Non-functional requirements

In this sub section, the non-functional requirements of use case II are defined.

<u>Ensure correctness</u>

For the automated generation of an import declaration to be relevant to customs authorities, correctness of the import declaration should be ensured. Correctness can be ensured if exchanged data and the business-to-business or business-to-government exchange is reliable. To that extent, integrity of both the data itself and integrity during exchange should be ensured (Hulstijn et al., 2012; Pruksasri et al., 2013). No party should be able to single-handedly alter data. In context of this use case, this means that the architecture should guarantee correct retrieval and calculation. Therefore, the following requirement is defined:

UC2-req. 2: The architecture should guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value

<u>Scalability</u>

Similar to use case I on the exchange of a pro forma invoice, the architecture should be able to handle high volumes of import declarations. Each generation of an import declaration is essentially a transaction. The architecture should thus be scalable to handle at least ten transactions per second. In addition, generation should take place in real-time. High delay of confirmation of transactions could result in delay of a shipment during importation as the shipment cannot be cleared by a customs authority without an import declaration being lodged first. Given the scalability issue in terms of limited transaction throughput of eight to fifteen transactions per second current blockchain-based initiatives experience, for a blockchain-based architecture to be relevant to both businesses and customs authorities it should be able to generate high volumes of import declarations in real-time. Therefore, the second non-functional requirement is defined as follows:

UC2-req. 3: The architecture should be able to generate high volumes of import declarations in real-time

This sub section defined the non-functional design requirements for the blockchain-based architecture that supports use case II. The next sub section provides a summary of the defined design requirements.

3.2.3 Sub-conclusion design requirements use case II

In this section, a set of three design requirements of use case II was defined based on the use case description in chapter 2 and a literature review on the data pipeline concept regarding data integrity. Table 4 provides an overview of the defined requirements, specifying code, definition and related sources. The next section defines the design requirements of use case III.

Table 4: Overview of defined design requirements use case II.

Design requirements use case II: Automated generation of an import declaration by aggregating already exchanged pro forma invoice data		
Code	*Definition*	*Related sources*
Functional		
UC2-req. 1	The architecture should enable a declarant to automatically generate an import declaration by retrieving already business-to-business exchanged pro forma invoice data and calculate customs value, and allow business-to-government exchange of the generated import declaration	Use case II description
Non-functional		
UC2-req. 2	The architecture should guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value	(Hulstijn et al., 2012), (Pruksasri et al., 2013)
UC2-req. 3	The architecture should be able to generate high volumes of import declarations in real-time	Use case II description, Introduction

3.3 Design requirements use case III

This section defines the design requirements of use case III based on the use case description in chapter 2 and a literature review on the electronic negotiable Bill of Lading.

3.3.1 Functional requirements

The goal of use case III is to transfer title to the goods using an electronic negotiable Bill of Lading. Therefore, the architecture should provide a means to transfer ownership of the electronic negotiable Bill of Lading to a new holder between businesses. The following design requirement can thus be defined:

UC3-req. 1: The architecture should provide business-to-business exchange to transfer ownership of the electronic negotiable bill of lading to a new holder

3.3.2 Non-functional requirements

Similar to use case I, the willingness of a business to use an architecture to exchange documents is also relevant for an electronic negotiable Bill of Lading. The main challenge of an electronic negotiable Bill of Lading is, according to Dubovec (2006, p. 447): ‘In an electronic environment, the challenge is to preserve the marketability of electronic records that replicate paper data, in particular by securing their authentic, unique, and confidential nature so as not to diminish confidence in the electronic system.’. In other words, the willingness of a business to use an electronic negotiable Bill of Lading depends on the ability to guarantee uniqueness and ensure confidentiality. To that extent, two non-functional requirements based on these elements are defined. In addition, current electronic negotiable Bill of Lading initiatives lack adoption. The use of a centralized title registry is often mentioned as barrier for adoption (Dubovec, 2006; Goldby, 2008). The use of a decentralized title registry has been proposed to overcome this barrier. In addition, part of the knowledge gap this research addresses is the scalability issue blockchain-based initiatives currently experience. In conclusion, the elements discussed above show the relevance of addressing confidentiality, guarantee of uniqueness, the need for a decentralized title registry and scalability of the architecture. To that extent, four requirements are defined regarding guarantee of uniqueness, confidentiality, decentralized title registry and scalability.

Guarantee of uniqueness

For an electronic negotiable Bill of Lading to be used to transfer title to the goods of a shipment, uniqueness should be guaranteed to overcome the double spending problem which is common in electronic environments. The underlying reason is that an electronic negotiable Bill of Lading is not simply an electronic version of a paper-based negotiable Bill of Lading due to the requirement of guaranteeing uniqueness whilst being transferable. Within electronic environments, guaranteeing uniqueness of a document (or record) is difficult to realize. Electronic documents can easily be copied or manipulated (Takahashi, 2016). This relates to the double spend problem. For example, business A is the holder of an electronic negotiable Bill of Lading, which is stored in the business' own document storage. Business A can use that electronic negotiable Bill of Lading to claim title to the goods. If business A is fraudulent, it can send a copy of the electronic negotiable Bill of Lading to business B whilst keeping its own copy. Business B is thus not guaranteed of uniqueness of the document. Business A can easily re-sell the electronic negotiable Bill of Lading to another party. In another example, a different fraudulent business can gain access to the document storage of business A and create a copy of the document. If the fraudulent business arrives at the port of discharge before business A, it can claim the goods from the carrier as it can surrender the negotiable Bill of Lading. As the carrier is required by law to release the goods upon surrendering, the fraudulent party will receive the goods. Business A, arriving second at the port of discharge will as a consequence not be able to claim the goods as the negotiable Bill of Lading was already surrendered and the goods are already released. An electronic negotiable Bill of Lading is thus prone to double spending and therefore uniqueness should be guaranteed.

In literature, the guarantee of uniqueness is often mentioned as part of the functional equivalence of an electronic negotiable Bill of Lading compared to a paper-based negotiable Bill of Lading (Dubovec, 2006; Lee et al., 2008; Takahashi, 2016; Van Boom, 1991). It is difficult to exactly define what constitutes functional equivalence. In short, functional equivalence is essentially the underlying legal infrastructure that should enable the electronic negotiable Bill of Lading to – similar to a paper-based negotiable Bill of Lading - 1) act as contract of carriage, 2) be a receipt of goods under contract of carriage and 3) be used to transfer title to the goods (Karan, 2011; Van Boom, 1997). The third element, transfer title to the goods implies the need of guaranteeing uniqueness as only one party at any given time can be in control of the original copy of the negotiable Bill of Lading (see section 2.4). To define functional equivalence more specific, this research refers to the United Nations Convention on Contracts for the International Carriage of Goods Wholly or Partly by Sea (also known as Rotterdam Rules) and the UNCITRAL Model Law on Electronic Transferable Records (in short UNCITRAL Model Law) that are aimed specifically at 'electronic negotiable transport records' (UNCITRAL, 2017; United Nations, 2008). As an electronic negotiable Bill of Lading can be seen as a type of negotiable electronic transport record, the Rotterdam Rules and UNICTRAL Model Law would apply (Karan, 2011). Both the Rotterdam Rules and the UNCITRAL Model Law address what constitutes functional equivalence of a negotiable electronic transport record. Takahashi (2016) combines the definitions of functional equivalence of the Rotterdam Rules and UNCITRAL Model Law to identify a set of guiding principles. A small adaption is made to fit with the electronic negotiable Bill of Lading of this use case. An electronic negotiable Bill of Lading should be subject to procedures that provide for: 1) exclusive control over issuance and transfer of the electronic negotiable Bill of Lading to the intended holder, 2) ensure the electronic negotiable Bill of Lading retains its integrity throughout its lifecycle, 3) enable the holder to demonstrate ownership of the electronic negotiable Bill of Lading and 4) provide confirmation that delivery to the holder has been effected or that the electronic negotiable Bill of Lading has ceased to have any effect or validity. Combing the guarantee of uniqueness and guiding principles for functional equivalence, the following design requirements is defined:

UC3-req. 2: The architecture should guarantee uniqueness of the electronic negotiable Bill of Lading following the principles for functional equivalence

<u>Data confidentiality</u>
The transfer of title to the goods by transferring ownership of an electronic negotiable Bill of Lading to a new holder via business-to-business exchange implies that the requirement regarding confidentiality (UC1-req. 5) as discussed previously in section 2.2 also applies to the architecture for this use case. Similar to a pro forma invoice document, a negotiable Bill of Lading contains sensitive commercial data which must be kept confidential and can only be accessed by authorized parties (Jarman & Luna-Reyes, 2016; Pruksasri et al., 2014). In case of a paper-based negotiable Bill of Lading, the document is sent in a closed envelop by mail courier to the new holder. This provides to some extent confidentiality of the businesses involved as only the businesses involved in the transfer know who the current holder is. However, in a typical blockchain ledger other businesses can see all transactions as the ledger is distributed. These transactions include the identity of the businesses involved in the transfer of ownership. This might reduce the willingness of businesses to use a blockchain-based architecture that implements an electronic negotiable Bill of Lading. Other businesses can see the content of the Bill of Lading as well as the identities of the businesses involved in transactions. Therefore, the architecture must provide confidentiality of both the negotiable Bill of Lading and the identities of business involved in the transfer of ownership transaction (Takahashi, 2016). A small adaption of requirement UC1-req. 5 to fit with this use case results in the following requirement definition:

UC3-req. 3: The architecture should ensure confidentiality of both the electronic negotiable Bill of Lading and identities of businesses involved in the transfer of ownership and should allow only authorized parties access to the electronic negotiable Bill of Lading or identity

<u>Lack of adoption of centralized title registry systems</u>
As discussed in the description of use case III in chapter 2, the use of an electronic negotiable Bill of Lading can have major benefits for carriers and other businesses as the cumbersome paper-based process to transfer ownership to a new holder becomes obsolete. Previous initiatives have attempted to create a system for an electronic negotiable Bill of Lading. However, there is no widespread adoption of these systems (Goldby, 2008; Pagnoni & Visconti, 2010; Takahashi, 2016). Insight into the barriers of adoption can support the design of a blockchain-based architecture that can overcome these barriers. Therefore, in this paragraph a short literature review is conducted to identify the barriers for adoption of a system that supports an electronic negotiable Bill of Lading. The blockchain-based architecture design should overcome these barriers in order to provide an electronic negotiable Bill of Lading that would be acceptable to carriers and other businesses.

Dubovec (2006) finds that previous initiatives failed or have limited adoption because of the use of a closed, centralized title registry that requires registration with a central authority in control. For example, the Bolero Title Registry is controlled by Bolero and use the Bolero Rule Book as legal agreement that requires registration. As previously introduced in section 2.4, a centralized title registry with a central authority in control results in (perceived) issues regarding system reliability and transparency (Goldby, 2008; Pagnoni & Visconti, 2010; Takahashi, 2016). A central title registry is prone to the single point of failure issue. If the title registry fails, no party can use the electronic negotiable Bill of Lading to transfer title to the goods. There is thus a need for an electronic negotiable Bill of Lading without the use of a centralized title registry to keep track of the current holder whilst guaranteeing uniqueness. For many years the idea was that an electronic negotiable Bill of Lading could only be implemented if some centralized title registry that keeps track of who is the current holder of the electronic negotiable Bill of Lading was required. The double spend problem in electronic environment prevented the use of a decentralized title registry. A blockchain-based architecture can in potential overcome this issue.

To scope the research, the assumption is made that following the guiding principles for functional equivalence as defined for requirement UC3-req. 2 is sufficient to legally recognize an electronic negotiable Bill of Lading as functional equivalent of a paper-based negotiable Bill of Lading. There would thus be no need for additional legal agreements that would require some form of registration. The underlying reason is that this research takes a more technical-oriented approach. It aims at developing an open, secure and scalable blockchain-based architecture that supports the functionality (negotiability) of an electronic negotiable Bill of Lading as it is a fundamentally different type of document to exchange electronically compared to other trade documents (e.g., pro forma invoice or non-negotiable Bill of Lading). In addition, blockchain technology has the potential to fulfill the principles for functional equivalence without a centralized title registry and central authority in control. Thus, this research focusses on addressing the barrier regarding the use of a centralized title registry with a central authority in control. The fourth design requirement is therefore defined as follows:

UC3-req. 4: The architecture should be open to businesses, without the use of a centralized title registry and without a central authority in control

Scalability

Between eighty and ninety percent of international shipping is by sea (Dubovec, 2006; IMO, 2018). While only a percentage of the shipments uses a negotiable Bill of Lading, still millions of negotiable Bill of Ladings are issued and transferred to new holders each year. For example, in the United States alone over 20 million negotiable Bill of Ladings are used each year for shipping goods (IHS Markit, 2017). Each issuing or transfer action is essentially a transaction and a negotiable Bill of Lading is often transferred multiple times during its lifecycle. For example, the shortest lifecycle consists of two transactions: 1) transfer from carrier to holder (consignee) and 2) transfer from holder to carrier). However, the lifecycle of a negotiable Bill of Lading will be longer than two transactions as otherwise a non-negotiable Bill of Lading would suffice. Therefore, in a global setting the architecture would have to be able to handle hundreds of millions of transactions per year. In addition, this handling should take place in real-time because delay in the transfer of ownership could affect the physical transport as goods cannot be released without surrendering a negotiable Bill of Lading. Given the scalability challenge in terms of transaction throughput blockchain-based initiatives currently experience (see section 1.2.3), taking into account the requirement to be able to handle high volumes of ownership transfer transactions is therefore very relevant.

The relevance of taking into account the scalability issue in context of an electronic negotiable Bill of Lading was addressed by other blockchain-based initiatives. For example, a representative from the CargoX initiative[13], that currently uses Proof-of-Work as consensus protocol, acknowledged that a scalability issue in terms of transaction throughput using the current architecture is very likely. Another initiative, called 'Wave', that presumably also uses Proof-of-Work might face the same issue

[13] Confirmed by Igor Jakomin, Chief Business Development Officer of CargoX at Blockchain Workshop organized by ALICE in Barcelona, Spain on May 9th 2018.

(Rizzo, 2015). Therefore, the need for a scalable blockchain-based architecture that supports an electronic negotiable Bill-of-Lading is included explicitly as the following requirement:

UC3-req. 5: The architecture should be able to handle high volumes of ownership transfers in real-time

The next sub section provides an overview of the defined design requirements of use case III.

3.3.3 Sub-conclusion design requirements use case III

In the previous sub section, a set of five design requirements for the blockchain-based architecture to implement use case III was defined. Table 5 provides an overview of the defined requirements' code, definition and related sources. The next section concludes this chapter.

Table 5: Overview of defined design requirements use case III.

Design requirements use case III: Transfer title to the goods using an electronic negotiable Bill of Lading		
Code	*Definition*	*Related sources*
Functional		
UC3-req. 1	The architecture should provide business-to-business exchange to transfer ownership of the electronic negotiable bill of lading to a new holder	Use case III description
Non-functional		
UC3-req. 2	The architecture should guarantee uniqueness of the electronic negotiable Bill of Lading following the principles for functional equivalence	(UNCITRAL, 2017), (Takahashi, 2016), (Dubovec, 2006), (Hong, 2012), (Lee et al., 2008), (Van Boom, 1997), (United Nations, 2008), (Karan, 2011)
UC3-req. 3	The architecture should ensure confidentiality of both the electronic negotiable Bill of Lading and identities of businesses involved in the transfer of ownership and should allow only authorized parties access to the electronic negotiable Bill of Lading or identity	(Pruksasri et al., 2014), (Jarman & Luna-Reyes, 2016), (Fawcett et al., 2007), (Takahashi, 2016)
UC3-req. 4	The architecture should be open to businesses, without the use of a centralized title registry and without a central authority in control	(Dubovec, 2006), (Goldby, 2008) (Mcdermott et al., 2017), (Takahashi, 2016)
UC3-req. 5	The architecture should be able to handle high volumes of ownership transfers of electronic negotiable Bill of Ladings in real-time	(IMO, 2018), (Dubovec, 2006), Use case III description, Introduction

3.4 Conclusion chapter 3

The definition of design requirements for the blockchain-based architecture to support the critical trade document use cases in this chapter answers sub question 4: '*What are the design requirements for a blockchain-based architecture design that supports the critical trade document use cases?*'. Based on the critical trade document use case descriptions in chapter 2 and literature reviews on the data pipeline concept and electronic negotiable Bill of Lading three sets of design requirements, both functional and non-functional, for each use case were defined. Table 6 provides an overview of the design requirements per use case. The requirements serve as input for the design choices for the blockchain-based architecture design in chapter 5 and architecture evaluation in chapter 7. The next chapter presents sub step 3.1 that identifies core blockchain technology architecture components.

Table 6: Overview design requirements.

Design requirements use case I: Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration	
Code	*Definition*
Functional	
UC1-req. 1	The architecture should provide business-to-business exchange of (a pro forma invoice) document to enable end-to-end visibility
UC1-req. 2	The architecture should provide business-to-government exchange to enable customs authorities to piggyback an exchanged document for manual cross-validation of an import declaration
Non-functional	
UC1-req. 3	The architecture should ensure confidentiality of both the exchanged document and identities of businesses involved in the exchange transaction and should allow only authorized parties access to the document or identity to overcome legality and liability issues
UC1-req. 4	The architecture should ensure integrity of the exchanged document
UC1-req. 5	The architecture should be open to businesses, without the use of a centralized document storage and without a central authority in control
UC1-req. 6	The architecture should be able to handle high volumes of document exchanges in real-time

Design requirements use case II: Automated generation of an import declaration by aggregating already exchanged pro forma invoice data	
Code	*Definition*
Functional	
UC2-req. 1	The architecture should enable a declarant to automatically generate an import declaration by retrieving already business-to-business exchanged pro forma invoice data and calculate customs value, and allow business-to-government exchange of the generated import declaration
Non-functional	
UC2-req. 2	The architecture should guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value
UC2-req. 3	The architecture should be able to generate high volumes of import declarations in real-time

Design requirements use case III: Transfer title to the goods using an electronic negotiable Bill of Lading	
Code	*Definition*
Functional	
UC3-req. 1	The architecture should provide business-to-business exchange to transfer ownership of the electronic negotiable bill of lading to a new holder
Non-functional	
UC3-req. 2	The architecture should guarantee uniqueness of the electronic negotiable Bill of Lading following the principles for functional equivalence
UC3-req. 3	The architecture should ensure confidentiality of both the electronic negotiable Bill of Lading and identities of businesses involved in the transfer of ownership and should allow only authorized parties access to the electronic negotiable Bill of Lading or identity
UC3-req. 4	The architecture should be open to businesses, without the use of a centralized title registry and without a central authority in control
UC3-req. 5	The architecture should be able to handle high volumes of ownership transfers of electronic negotiable Bill of Ladings in real-time

4. Core blockchain technology architecture components

This chapter analyses blockchain technology architecture literature to identify the core components of a blockchain technology architecture. The goal of this chapter is to answer sub question 3*: 'What are the core blockchain technology architecture components?'*. To answer the sub question, this chapter takes several steps. First, a literature review on blockchain technology architecture design is performed in section 4.1 to identify core components on a high level. After the literature review, for each of the core components identified, a more technical-oriented literature analysis is conducted in sections 4.2 through 4.5. The sub deliverable that results from answering the sub question consists of a set of core blockchain technology architecture components. For each component, the component's relevance in context of the architecture, technical functioning and sub components (including design alternatives) are discussed. The components are used as basis for the blockchain-based architecture design that is described in chapter 5.

4.1 Identifying core components

The objective of this research is the design of a blockchain-based architecture. As previously described in section 1.4, an architecture can be defined as: 'the (system) fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.' (ISO, 2011, p. 1). Insight in the environment in which the system (the blockchain-based architecture) should function was already discussed in chapters 2 and 3 that described the use cases and defined requirements to support the use cases. For the design of the blockchain-based architecture that describes the fundamental concepts or properties of the system, insight in its elements and relationships is required. These elements and relationships can be described as the (technical) components of the architecture design. As the architecture is blockchain-based, core blockchain technology architecture components need to be identified that serve as basis for the blockchain-based architecture design in chapter 5.

4.1.1 Search strategy

To identify the core components, academic databases Google Scholar, Scopus and SpringerLink with search terms 'blockchain' or 'distributed ledger' and 'architecture (design)' were used to retrieve relevant literature. The search showed that literature on blockchain technology is wide-spread and only a limited amount of literature focusses on architecture design specifically. In this research, the objective is to develop an open, secure and scalable blockchain-based architecture design. This requires a technical-oriented approach to make well-grounded design choices for the architecture, especially regarding data security and scalability as it requires analysis on a transactional level. Only a handful of papers classify blockchain system architecture components via taxonomies or ontologies that provide sufficient technical depth. A taxonomy is used within architecture design to: 'classifying the existing work into a compact framework, which allows an architect to explore the conceptual design space and enables rigorous comparison and evaluation of different design options' (Xu et al., 2017, p. 3). Tasca & Tessone perform a review on existing blockchain technology literature and propose a 'reference architectural model for blockchains and their possible configurations' (2018, p. 37). Xu et al. (2017) provide a taxonomy that includes detailed architectural characteristics, the impact of design decisions and provide a design process for blockchain-based systems. An in-depth analysis on blockchain technology concepts and systems from a data processing perspective is provided by Dinh et al. (2018). Xu et al. (2016) discuss blockchain technology from the perspective of a software architecture and provide a set of technical design decisions that support development of a blockchain-based architecture. Because the four retrieved articles provide sufficient technical depth and support development of blockchain-based architectures, these articles are used to identify core blockchain architecture components.

4.1.2 Identified core components

In this section, a literature review is conducted to identify core blockchain architecture components that form the basis for a blockchain-based architecture design.

Table 7: Overview blockchain technology architecture taxonomy or ontology articles.

Article	Taxonomy or ontology main elements/components	Sub elements/components
(Tasca & Tessone, 2018)	Consensus	[consensus network topology, consensus immutability & Fault Tolerance, gossiping, consensus agreement]
	Transaction capabilities	[data structure in block header, transaction model, server storage, block storage]
	Native currency/tokenization	[native asset, asset supply management, tokenization]
	Extensibility	[interoperability, governance, script language]
	Security & privacy	[data encryption, data privacy]
(Xu et al., 2017)	(de)centralization	[fully centralized, partially centralized & partially decentralized, fully decentralized, verifier]
	Storage and computation	[item data, item collection, computation]
	Blockchain configuration	[blockchain scope, data structure, consensus protocol, protocol configuration]
(Dinh et al., 2018)	Application	[smart contracts]
	Execution engine	[runtime environment]
	Data model	[structure]
	Consensus	[Proof-of-Work variants, Proof-of-Stake, BFT variants, Federated, Non-Byzantine, others]
(Xu et al., 2016)	Application layer	[off-chain data, off-chain control]
	Blockchain layer	[permission management, incentive mechanism, transaction validation, mining, secure clearing payment, chain]
	Blockchain network	

Table 7 provides an overview of the components for a blockchain-based architecture as proposed by different articles. The overview shows that architecture components are widespread with different definitions used for similar components. A core component in one article is defined as sub component in another article. In addition, one should note that these articles are taxonomies or ontologies that do not necessarily aim at defining components, but rather focus on supporting development of architectures by proposing elements for which design decisions have to be made. Components are based upon these elements. In addition, to scope the research not all components and sub or sub sub components can be taken into account. Therefore, a generalization is made that groups and combines similar elements and derives core components from this generalization. A set of four core components can be identified. Table 8 provides an overview of the identified components.

Network configuration component
Within a blockchain-based architecture, a data structure (also known as blockchain ledger) is distributed through a peer-to-peer network. The network configuration component therefore concerns the peer-to-peer network of nodes that each hold a copy of the data structure and how these nodes can participate in the network (Tasca & Tessone, 2018).

Data structure and storage component
To cover the different aspects regarding how data within the network is structured and stored, the data structure and storage component is identified as core component (Xu et al., 2017).

Consensus mechanism component
To update the state of the blockchain ledger, a set of rules and procedures to validate transactions is needed. Therefore the consensus mechanism is identified as core component of which its purpose can be defined as: 'the set of rules and procedures that allow to maintain and update the ledger and to guarantee trustworthiness of the records in the ledger, i.e., their reliability, authenticity and accuracy' (Tasca & Tessone, 2018, p. 15).

Application component
To determine what is being stored on the blockchain ledger and how business logic is implemented, the application component is identified as core component (Dinh et al., 2018).

Table 8: Overview core blockchain architecture components

Core component	Purpose
Network configuration	Determine the peer-to-peer network of nodes that hold a copy of the blockchain ledger and how these nodes can participate in the network
Data structure and storage	Determine the way in which data within the network is structured and stored
Consensus mechanism	Determine the set of rules and procedures to validate transactions to update the state of the blockchain ledger.
Application	Determine what is being stored on the blockchain ledger and how business logic is implemented.

Cryptography is an important underlying technology of blockchain technology and is therefore often mentioned as architecture component in literature (Dinh et al., 2018). Because cryptography is an underlying technology that is used by all other components, in this research cryptography is not identified as a core component. However, because of its importance to analyze the other components, the Box 2 briefly discusses the basics of cryptography.

Box 2: Cryptography basics.

Cryptography
Cryptography is used as security measure in information systems to hide and reveal data using encryption and decryption. In blockchain technology, cryptography is used for two purposes: validating digitally signed transactions and cryptographically linking blocks of ordered transaction to ensure integrity of the blockchain ledger (Narayanan et al., 2016). Hashing and symmetric/asymmetric key encryption and decryption are basic concepts of cryptography. Both concepts are discussed below.

Hashing
In cryptography, hashing is used to prove integrity of data. A hash is generated via a hash function that takes any data input and converts it into a string with specific number of characters. The Secure Hash Algorithm (SHA)-256 hash function is a type of hash function that is commonly used in cryptography. The number 256 refers to the length of the hash in bits. The function is one-way, which means that it is easy to generate a hash from a specific data input, but it is impossible to retrieve the data input from a generated hash.

$$Compute\ \boldsymbol{hash}\ from\ \boldsymbol{input}\ via\ hash\ function\ \boldsymbol{H}(\boldsymbol{input})\ \rightarrow\ \boldsymbol{hash} = \boldsymbol{H}(\boldsymbol{input})$$

Formula 1: Approach for generating a hash. A data input is inserted into a hashing function. The function then generates a hash output.

The underlying reason is that a hash is unique for a specific data input. A slightly different data input results in a completely different hash output, as shown in the example below. In addition, a hash function has a collision-free property. This means that it is impossible to find different data inputs with the same hash (Garay, Kiayias, & Leonardos, 2017).

$$Hash(John\ Doe) = 6cea57c2fb6cbc2a40411135005760f241fffc3e5e67ab99882726431037f908$$
$$Hash\ (Jon\ Doe) = 03a1970e9972b543424f763c00eb48b06f92e0f44f0ea0d46882bf5449d2d3e1$$

The hash (SHA-256 function) of two similar data inputs, for example John Doe versus Jon Doe, results in two different hash outputs. As a result, generating a hash using formula 1 is simple, while retrieving the input from the hash given only the hash outcome is impossible.

Hashing can be used to prove integrity of data. As discussed above, a hash is unique for a specific data input. If the data input changes only slightly, the resulting hash will change completely. This can be used to verify that data has not been altered and thus prove integrity. To do so, first the hash of a data input is stored in a database. Next, to verify that a data input is not altered, the input is hashed. The new hash is compared to the existing hash in the database. If the hashes match, the data input has not been altered and integrity of data is proved. If the hashes do not match, it proves that two different data inputs were used. In that case, data integrity cannot be proved. For example, in a pro forma invoice document that has been hashed and where the hash is stored in some database, the contents of the document cannot be altered. If someone would try to do so, the stored hash and the hash of the altered document would not match. One could then prove that the data has been altered.

Encryption
To securely send data and avoid access by unauthorized parties, encryption is needed. Two types of encryption can be distinguished: symmetric and asymmetric key cryptography (CGI, 2004).

Symmetric key cryptography
In symmetric key cryptography a key is used to both encrypt (from plain text to ciphertext) and decrypt a message (from ciphertext to plain text). In notation, E means encryption of a message using the key and D means decryption using the key as shown below.

$$Ciphertext = E(message, key)$$
$$Plain\ text = D(message, key)$$

Asymmetric key cryptography

Asymmetric key cryptography uses public (PU) and private (PR) key pairs to encrypt data. Data that is encrypted using a PU can only be decrypted by the corresponding PR. Therefore, the PR is kept secret, while the PU is distributed to other parties. Given only the PU, it is impossible to know what the PR is. So, to securely send data a party encrypts the data using the recipient's PU. Only the recipient can then access the data by decrypting it with its PR.

$$Ciphertext = E(message, PU)$$
$$Plain\ text = D(message, PR)$$

Digital signatures

To prove ownership of a digital asset and to authorize transactions, digital signatures are required. Digital signatures have the same goals as handwritten signatures: 1) a specific signature can only be made by one individual, but validated by all individuals, 2) a signature is only valid for one specific transaction, and can thus not be copied to other transactions and 3) a signature ensures non-repudiation which means that the party that created the signature cannot deny creating the signature (Drescher, 2017). This ensures that a transaction has not been altered and originated from the initial signer of the transaction that cannot deny creating the signature. To achieve the goals as mentioned above, digital signatures are based on asymmetric key cryptography.

In digital signatures, the PR is kept secret in order to sign transactions. The PU is distributed and used by other parties to verify the signature. To create a signature, the PR and hash of the transaction are input to a sign algorithm that that encrypts the hash using the PR. To verify a signature, the signature, transaction hash and PU are input to a verification algorithm. If the outcome is true (Boolean value), the signature is valid. In formula form:

$$Signature = E(hash(transaction), \mathrm{PR})$$
$$Signature\ verification = verify\ (signature, hash(transaction), PU)$$

This section identified core components for a blockchain-based architecture on a high level. Sections 4.2 through 4.5 discuss each of the identified core components in more detail.

4.2 Network configuration component

In this section the network configuration component is discussed. It discusses the peer-to-peer network and topology that determines how nodes can participate in the network.

4.2.1 Peer-to-peer network of nodes

To enable exchange of transactions between parties without a single intermediary that has control over the network, a peer-to-peer network is needed in which a data structure can be distributed (Zhang & Jacobsen, 2018). In a peer-to-peer network a group of peers (or nodes) is interconnected via an internet protocol (e.g., TCP/IP). To participate, a party has to be represented by a node (Narayanan et al., 2016).

4.2.2 Topology

For a blockchain-based architecture, two main types of topologies can be distinguished: 1) permissionless and 2) permissioned (Xu et al., 2017). This sub section discusses both topologies.

Within a blockchain-based architecture, nodes can have three types of permission (Xu et al., 2016). First, a node can have read permission that enables the node to read the blockchain ledger. Second, a node can have write permission that enables the node to write to the blockchain ledger. This means that the node can create transactions and append transactions to the blockchain ledger. Third, a node can have permission to participate in the consensus protocol that enables the node to propose new blocks of transactions. The choice of network topology is based on four requirements: control, data ownership, privacy and access (Ølnes et al., 2017).

Permissionless
The permissionless network topology is the initial concept of blockchain technology as introduced by Nakamoto (2008) in his Bitcoin blockchain. In this type of topology anyone has read, write permission or permission to participate in the consensus protocol rendering nodes anonymous. There is no intermediary that can control permission for individual nodes. Nodes can join or leave at their own discretion (Pass, Seeman, & Shelat, 2017). Examples of permissionless blockchain networks are Bitcoin and Ethereum.

Permissioned
Not all applications of a blockchain-based architecture can use a permissionless network topology. Especially in enterprise setting, some form of control of permission is needed to security purposes. For example, if the blockchain ledger contains sensitive data, not all nodes should have read access. To control permissions of individual nodes, a permissioned network topology can be used. In this type of topology, read, write permission or permission to participate in the consensus protocol is controlled by an intermediary. In literature, this type of topology is often referred to as private or consortium depending on configuration of permissions (Tasca & Tessone, 2018). To allow for more a fine-grained design, this research only uses the term permissioned and discusses each type of permission individually.

<u>Identity and permission management</u>
If a permissioned network configuration is used, two additional components have to be considered: 1) identity management and 2) permission management. These components are needed to identify a node (party) and assign permissions to this node. Identity and permission management consist of four components: identification, authentication, authorization and decision (Karp, Haury, & Davis, 2009). Identity management is responsible for the identification and authentication components. Identification assigns a digital identity to a real-world entity. For example, a user account to a person. Authentication ensures that only the assigned real-world entity can use the digital identity and transactions are authentic. Digital signatures are used to prove authenticity. Without identity management, there is no possibility to correctly assign permissions to a node as the node can create its own pseudonym identity (public-private key pair) rendering the node anonymous like in a permissionless network configuration. As a result, no guarantee that the used public-private key pair used to signed transactions belongs to the real-world identity can be provided. The inability to identify a real-world identity is given by the following example in which Alice signs a message to authenticate the message originated from her. Alice signs the hash of a message with her PRalice to authenticate she is the actual sender of the message. This results in the following digital signature:

$$\text{Signature} = E(\text{message hash}, \text{PRalice})$$

Alice sends the message, her public key (PUalice) and signature to Bob. Bob uses the signature, and PUalice to decrypt the signature. Next, Bob hashes the message and compares both hashes. If the hashes are equal, Alice has authenticated that she is indeed the sender of the message. However, Bob cannot know if the real Alice has signed the message. Anyone can mimic Alice and state that the used public-private key pair belongs to Alice, while the PU actually belongs to someone else.

Therefore, the PU has to be bind to the real-world identity of Alice to be certain that the Alice signed the message.

Permission management takes care of authorization and decision. Authorization assigns permissions to a digital identity. The decision component provides access based on assigned permissions which are for example stored in access control policies. There are three types of permission to be considered in permission management: 1) permission to read the blockchain ledger, 2) permission to write to the blockchain ledger and 3) permission to participate in the consensus mechanism.

4.2.3 Sub conclusion network configuration component

For the network configuration component, two sub components with several design options were identified and discussed in this section. Table 9 provides an overview of the component. The next section discusses the data structure and storage component of a blockchain-based architecture.

Table 9: Overview network configuration component.

Core component	Sub components	Sub sub components	Design options	Option components
Network configuration	Peer-to-peer network			
	Topology		Permissionless	
			Permissioned	Identity management
				Permission management

4.3 Data structure and storage

This section analyses the data structure and storage component. This component determines the way in which data within the network is structured and stored. In a blockchain network there are two options of storing data. Data can be stored either on-chain on the blockchain ledger or off-chain. Both options are discussed in the next two sub sections.

4.3.1 Blockchain ledger

The main data structure of a blockchain-based architecture is the blockchain ledger. A blockchain ledger is a cryptographically linked list of blocks. Each block contains a set of order transactions. The blockchain ledger fulfills a fundamental rule related to internal consistency. Internal consistency can be described as creating an 'internally coherent data structure that can keep a consistent record of transactions' (Dhillon, Metcalf, & Hooper, 2017, p. 15). The blockchain ledger reflects 'historical and current states maintained by the blockchain' (Dinh et al., 2018, p. 1368). Each update to the state is appended to the blockchain ledger as new transaction within a block (Xu et al., 2016). In other words, transactions are operations that are used to update the state of any digital asset on the blockchain ledger (Dinh et al., 2018). A block consists of two elements: a header and a body. Transactions are stored in the body of a block (Dhillon et al., 2017). To link the blocks, hash pointers are used (Narayanan et al., 2016). A hash pointer serves two functions. First, it points to a location where data is stored. This enables data retrieval if a separate data storage is used. Second, the pointer is the hash of the data so that data integrity can be verified. Hash pointers thus ensure internal consistency of the blockchain ledger. Figure 16 depicts the linking of blocks using the hash pointer. The use of hash pointers ensure that new blocks can only be appended to the end of the blockchain ledger because otherwise the ledger would be invalidated as the cryptographic link is broken. As a result, transactions are immutable once stored on the blockchain ledger as part of a block. Consequently, data stored within the transaction also becomes immutable.

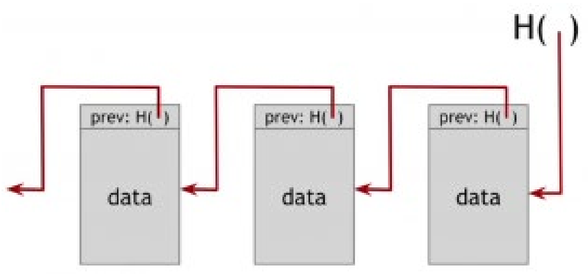


Figure 16: Blockchain ledger [as depicted by (Narayanan et al., 2016)*].*

To improve efficiency of the blockchain ledger, Merkle trees are used. In a Merkle Tree, a single value root hash is generated from the individual hashes of the transactions in the block. The root hash is stored in the header of the block (Narayanan et al., 2016). In case of four data elements A, B, C and D the tree is built as follows (see Figure 17). The corresponding hashes H are H(A), H(B), H(C), H(D). In the next level of the Merkle Tree, H(A) and H(B), and H(C) and H(D) are combined and hashed to create H(H(A)H(B)) and H(H(C)H(D)). These hashes are combined and hashed again to get the Root hash H(H(H(A)H(B))(H(H(C)H(D))). The Merkle tree can be used to prove existence of a transaction without having to check the bodies of all blocks due to the hashes that provide a cryptographic link. To prove existence, the transaction and only the hashes on the path between the transaction and Root hash have to be provided. Each hash is re-generated and compared to the old hash. In this way, all hashes can be verified. If the hashes match they are valid and the existence of a transaction is proved.

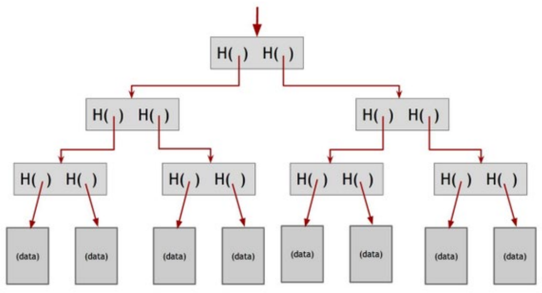


Figure 17: Simplified Merkle tree [as depicted by (Narayanan et al., 2016)*].*

Transaction model

The way in which transactions are used to keep track of the state of the blockchain ledger is known as the transaction model and is defined by Tasca & Tessone (2018, p. 18) as: 'How nodes connected to the P2P network store and update the user information in the distributed ledger'. Currently, there are two general alternatives for the transaction model: 1) Unspent Transaction Output (UTXO) and 2) Account-based (Tasca & Tessone, 2018; Xu et al., 2017).

<u>UTXO-based</u>

UTXO is a stateless transaction model that is mostly known from its use within Bitcoin. A new transaction spends the state of prior transactions that are owned by a party to create a new transaction. To that extent, each transaction has a PU that acts as address and can only be signed by the party that owns the corresponding PR to create a valid signature and thus transaction (Tschorsch & Scheuermann, 2016). This overcomes the double spend problem as an unspent transaction can only be spend once by the owner: the transaction is spent as input to a new transaction that contains the PU of the new owner. The most intuitive analogy is to see UTXO as a wallet of bank notes. If a party wants to spend €100, the party combines multiple bank notes (unspent transactions) that together add up to €100. Upon payment, the party spends the notes and thus no longer owns the notes.

Account-based

In contrast to UTXO, the account-based transaction model does keep a state for each account on the blockchain ledger. Transactions include operations to update the state based on the current state to create a new state. Accounts are in essence smart contracts. To update the state, the transaction that contains update operations, is executed. The state of the smart contract is stored as key-value pair. Where the state is stored is left outside the scope. Each node keeps track of all states of all contracts in a combined world state locally as shown in Figure 18. A new transaction is based on the world state. At any moment in time, the world state can be checked by traversing through the blockchain that acts as log to re-compute the world state. The account-based transaction model can be compared to a credit card. If a party wants to spend €100 using a credit card, the credit card company checks if the party has sufficient balance in its account to fulfill the transaction.

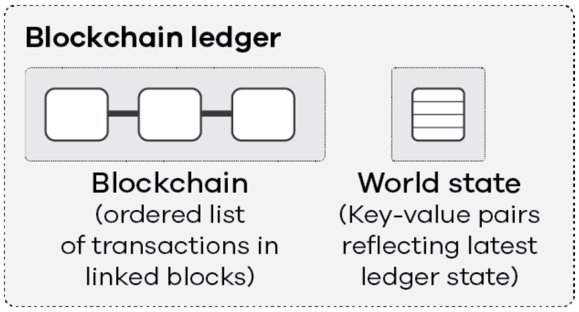

Blockchain ledger
Blockchain
(ordered list of transactions in linked blocks)
World state
(Key-value pairs reflecting latest ledger state)

Figure 18: Account-based transaction model.

Ledger structure

Within a blockchain network a single or multiple blockchain ledgers can be distributed. The use of a structure with multiple blockchain ledgers can improve data security as data can be grouped an only allow nodes with read or write permission to access the ledger. In addition, a multiple ledger structure can improve scalability because of the smaller file size (Xu et al., 2017).

4.3.2 Off-chain storage

To reduce costs, provide confidentiality of sensitive data and improve performance and flexibility, within a blockchain-based architecture data can be stored in an off-chain storage (Xu et al., 2017). In that case, only a reference to the data is stored on the blockchain ledger. The actual data is stored in the off-chain storage. To ensure integrity of the data, the reference includes some form of hashing. For example, the hash of the data can be added as meta-data to the reference or a hash pointer can be used. The off-chain storage can in principle be any type of third-party or peer-to-peer system. An often-mentioned initiative in light of blockchain technology is the InterPlanetary File System (IPFS) (https://ipfs.io/). IPFS is a distributed database that stores data and creates hash pointers to reference to the data whilst ensuring integrity.

4.3.3 Data security

To ensure confidentiality of sensitive data that is stored within the blockchain-based architecture, data security measures can be taken. In context of blockchain technology, two types of data security are proposed: 1) traditional encryption and 2) Zero-Knowledge Proof. In traditional data is encrypted before being stored. It requires additional decryption key management for the distribution of decryption keys. A new technology, Zero-Knowledge Proof does not require decryption key management. In Zero-Knowledge Proof, a special hash is used to generate validation without decryption (Tasca & Tessone, 2018).

4.3.4 Sub conclusion data structure and storage component

For the data structure and storage component, two sub components with several design options were identified and discussed in this section. Table 10 provides an overview of the component. The next section discusses the consensus mechanism component of a blockchain-based architecture.

Table 10: Overview of data structure and storage component.

Core component	Sub components	Sub sub components	Design options	Option components
Data structure and storage	Blockchain ledger	Transaction model	UTXO-based	
			Account-based	
		Ledger structure	Single ledger	
			Multiple ledger	
	Off-chain storage			
	Data security		Encryption	Decryption key management
			Zero-Knowledge Proof	

4.4 Consensus mechanism component

This section discusses the consensus mechanism component for a blockchain-based architecture. The purpose of the consensus mechanism is: 'the set of rules and procedures that allow to maintain and update the ledger and to guarantee trustworthiness of the records in the ledger, i.e., their reliability, authenticity and accuracy' (Tasca & Tessone, 2018, p. 15). The consensus mechanism is used to reach consensus on the validity of transactions within a peer-to-peer network without an intermediary to overcome Byzantine failures (Nguyen & Kim, 2018). Byzantine failure relates to the concept that in a peer-to-peer distributed system not all nodes are interconnected and nodes can be faulty or malicious (Vukolić, 2016)[14]. This can affect the propagation of transactions throughout the network as not all nodes receive the same transaction without any delay. A malicious node can exploit this failure by sending two transactions that spends the same digital assets to different nodes that do not know about the other transaction, also known as the double spend problem. Therefore, if no intermediary is used to validate transactions, a consensus mechanism is needed to validate transactions. The consensus mechanism consists of two components: 1) a transaction process and 2) a consensus protocol for the ordering step within the transaction process.

4.4.1 Transaction process

Two types of transaction process currently exist: 1) order-execute and 2) execute-order-validate. In an order-execute transaction process, transactions are ordered first using a consensus protocol and then executed by each node in the same order (Androulaki et al., 2018). This type of transaction process is used by almost every blockchain-based system. After the consensus protocol finds a block proposal with an ordered list of transactions, the transactions which include operations are executed sequentially by each node against the state of the blockchain ledger to determine validity of the transaction. Androulaki et. al (2018) find that this transaction process has several shortcomings. Because transactions are executed after ordering, the transactions need to be deterministic to reach consensus on the state between nodes. If the transactions are not deterministic, nodes will not reach consensus on the state. Smart contracts therefore need to be written in the same code language which reduces flexibility. In addition, if in a permissioned network data is encrypted to provide confidentiality, nodes that do not have access to the data that is included in transactions cannot check validity of the transaction as they cannot execute the transaction against the state. This can cause issues as transactions are not deterministic. Also, transactions have to be validated in sequence which limits scalability.

[14] Byzantine failure is described by the Byzantine Generals' Problem (Lamport, Shostak, & Pease, 1982) which is left outside the scope of this research.

To overcome these shortcomings the Hyperledger platform proposes a different process as shown in Figure 19 that can be used in a permissioned blockchain network: execute – order – validate (Androulaki et al., 2018). The main difference is that not every node executes the transaction against the state of the blockchain ledger to determine validity. Instead, an issuing node sends a transaction proposal to a limited number of nodes that confirm validity of the transaction (the confirming nodes). Which nodes should confirm validity is stored in a confirmation policy. Because the network is permissioned, identities of the nodes are known. The confirming nodes execute the transaction against the state of the blockchain ledger. The result is a write set that contains the state update and a read set for version control. After collection of the read and write sets from the confirming nodes, the issuing node sends the transaction including read and write sets to the consensus protocol for ordering. The transaction thus no longer contains an operation, but already the new state in the form of a write set. After a valid block proposal is found in the consensus protocol, nodes receive the block proposal and validate the transactions by checking for each transaction if the confirmation policy is fulfilled and if the read set has the correct version. If valid, the block is appended to the blockchain ledger and de write set is used to directly update the state of the blockchain ledger. As a result, contracts do not have to be written in the same code language which improves flexibility. Also, transactions do not have to be deterministic. The confirmation policy ensures that transactions are deterministic in case data is encrypted as the confirming nodes have checked validity and have signed the read and write sets. In addition, transactions can be validated in parallel which improves scalability.

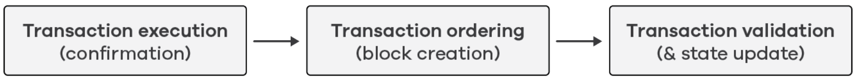


Figure 19: Execute - order - validate transaction process.

4.4.2 Consensus protocols

To be Byzantine fault tolerant, a consensus protocol is needed that allows nodes to reach agreement on the order of transactions (Chalaemwongwan & Kurutach, 2018; Tasca & Tessone, 2018). Currently, many protocols exist that are either based on probability or voting (Nguyen & Kim, 2018). Examples of probabilistic protocols are Proof-of-Work, Proof-of-Stake and Proof-of-Burn (Nguyen & Kim, 2018). These protocols are commonly used in context of blockchain technology and require a node to put in computational work to find a valid block proposal containing a set of ordered transactions. Which node finds a block proposal first is based on probability. Protocols based on probability are seen in context of permissionless blockchain networks because they allow for overcoming the double spend problem in anonymous settings by introducing an economic incentive. The probabilistic protocols solve the Sybil attack problem that voting-based protocols are prone to in permissionless networks (Tschorsch & Scheuermann, 2016). In short, in a Sybil attack a malicious node creates many nodes to have a larger chance of being selected as leader to propose a new block in a voting-based consensus protocol. As a trade-off, probabilistic consensus protocols face limited scalability in terms of transaction throughput (Chalaemwongwan & Kurutach, 2018; Scherer, 2017; Vukolić, 2016). Because permissioned blockchain networks can control who has permission to participate in the consensus protocol, the use of traditional consensus protocols that rely on voting and have high scalability in terms of transaction throughput are proposed in literature (Gramoli, 2017). The reason voting-based protocols only work in a permissioned network is because they are prone to Sybil attacks and therefore need the identities of nodes that can participate. The voting-based consensus protocols are also known as Byzantine Fault Tolerance (BFT) consensus protocols[15] that are widely used in existing distributed systems and come in many varieties.

[15] In BFT the general theory is that if at least two-thirds of the nodes is honest, the network will reach consensus as a majority will agree on the order of transactions. The number of nodes that can be faulty is given by the formula: total number of nodes = 3 * faulty nodes + 1.

In this sub section, only two consensus protocols are analyzed and compared. Within the probabilistic consensus protocols Proof-of-Work is the most well-known and used protocol as it was first introduced by Nakamoto (Nakamoto, 2008) in his Bitcoin blockchain. While many BFT protocols that rely on voting exist, Practical Byzantine Fault Tolerance (PBFT) is often mentioned in context of achieving scalability of blockchain technology (Nguyen & Kim, 2018). Therefore, in this research the scope is limited to these two protocols. This sub section first discusses Proof-of-Work. Next, PBFT is discussed. The section ends with a comparison of both protocols.

Proof-of-Work
The Proof-of-Work consensus protocol for use in blockchain technology was introduced by Nakatomo (2008). In Proof-of-Work, nodes have to solve a crypto puzzle to find a valid block of ordered transactions (see Box 3 for the functioning of a crypto puzzle). This overcomes the double spend problem as solving the puzzle requires computational effort (also known as mining) and thus takes time. It reduces the speed at which new transactions can be added to the ledger. Without the required computational effort, malicious nodes could easily create new blocks containing the same transaction, hence double spend the transaction. In Proof-of-Work, the first node to find a valid block receives a reward in the form of cryptocurrency. Which node solves the puzzle first is completely probabilistic. If all nodes have equal computational power, the change of winning is equal. This makes double spending in large networks non-profitable because the longest version of the blockchain ledger is seen as correctly representing the state as it has most time and thus work invested in it. This makes it harder for malicious nodes to double spend by adding fraudulent transactions because it has to try to create a fork of the ledger which is longer. This requires additional computational power. It is more profitable to participate as honest node and receive a reward for mining blocks (Narayanan et al., 2016; Ølnes, 2016).

Box 3: Crypto-puzzle basics.

What is a crypto-puzzle?
A crypto puzzle is a challenge - based on hashing - presented to a node that requires computing power and thus takes time to solve. The challenge is to find an input to the hash function so that the solution is lower than a target value. For example, in Bitcoin the generated hash solution of a block proposal needs to start with a specific number of leading zeros in order to be valid. More leading zeros means a lower target value. The input consists of the block proposal data (e.g., previous block hash and transactions) and an additional string, also known as the 'nonce'. In formula form:

$$Hash(block\ data + nonce) < target\ value$$

The only way to find a valid solution is by using brute-force. Brute-force is based on trial-and-error principle. A node keeps trying a new nonce for each trial until a valid solution has been found. Thus, by varying the nonce slightly for each trial, different solutions can be generated. This process of trial-and-error continues until a valid block hash solution is found that is lower than the target value. The network sets the target value to change the complexity of the crypto puzzle to control the speed of block validation. By varying the target value, the solution space is either wider or smaller. This influences the complexity of the puzzle and thus time required to find a valid solution. A wider solution space results in less complexity, a smaller space in more complexity. In Bitcoin that uses Proof-of-Work, the network aims to find a valid solution for a block proposal every ten minutes (Tschorsch & Scheuermann, 2016).

Process

The Proof-of-Work consensus protocol to find a valid block proposal consist of three main steps:

1. broadcast new transactions to the network
2. mine a block proposal
3. validate the block proposal

The next paragraph discusses each step. Figure 20 provides an overview of the functioning of the consensus mechanism with an order-execute transaction process and Proof-of-Work as consensus protocol.

First, when any node wants to add a transaction to the ledger, it signs a transaction with its PR and broadcasts the transaction to all nodes it knows in the network to be validated. Because of the peer-to-peer nature of the network in which not all nodes are fully interconnected, the receiving nodes propagate the transaction to other nodes in the network. This is known as the flooding algorithm in peer-to-peer networks (Narayanan et al., 2016). However, nodes only pass on transactions if a set of criteria are met. Mining nodes that receive the transaction, place the transaction in their own 'pool' of pending transactions after performing several checks (Karame, Androulaki, & Capkun, 2012). First, it checks if the signature is valid. Also, it checks if the transaction already exists in its pending transaction pool. In case a mining node receives a transaction that is already in the pool, the process terminates to avoid transactions ending up in an infinite loop. The reason behind the pool is to group multiple transactions into one block in order to minimize validation time. If each transaction would be validated via the Proof-of-Work protocol separately, the system would be extremely slow. The required propagation of transaction to inform the complete network of the transaction causes latency that can result in nodes having different pending transaction pools. This latency can be abused to double spend a transaction. For example, malicious node A broadcasts two transactions ADB (from node A to B) and ADC (from node A to C), simultaneously. In case node C is owned by node A, node A spends the transaction to itself. Part of the network will receive ADB first, while the other part will receive the ADC transaction first. In this way, a transaction can be spent twice if both transactions are accepted by different parts of the network. However, several measures are in place to mitigate this exploit. A node will add the first transaction it receives to its pending transaction pool. When the second transaction is received, the node will discard this transaction as it is already spent via the first transaction in the pending transaction pool. So, a transaction can never be spent twice at or be relayed to other nodes by the same node. However, this leads to nodes in disagreement of the next block of transactions to be validated and added to the blockchain ledger.

To resolve the disagreement, the concept of mining is introduced. Mining is the use of computational resources (e.g., CPU power) to 'mine' a block. In other words, to solve a crypto puzzle. To mine a block, three steps are performed. First, a mining node builds a candidate block consisting of pending transactions from its own pool. It can do so at its own discretion. For each mining node, this block can be completely different, depending on the pending transaction pool. Next, the mining node tries to find a valid block hash solution by solving the crypto puzzle using different nonces. Third, if a mining node has found a valid block hash solution, it broadcasts the candidate block and used nonce to all other nodes in the network. Other nodes validate the proposed block individually in two ways. First, a node checks whether one or more transactions in the proposed block already exist in its copy of blockchain ledger. When one or more transactions already exist, the node will discard the complete candidate block. Otherwise, if the transactions in the candidate block are new to the node, it will check if the block is valid. This check is performed by generating the hash solution using the candidate block data and the nonce provided by the mining node. If the hash solution is valid, the node approves the block and adds it to its own version of the blockchain. Next, the node propagates the proposed block to other nodes. In addition, a mining node removes any transaction from its pending transaction pool that is double spent provided the transactions in the block. So, if transaction ADB is in the validated block, the double spent ADC transaction that still resides in the pending transaction pool is removed. The mining node that mined the proposed block receives its reward if the block is added to the blockchain, as it is proof of performed work.

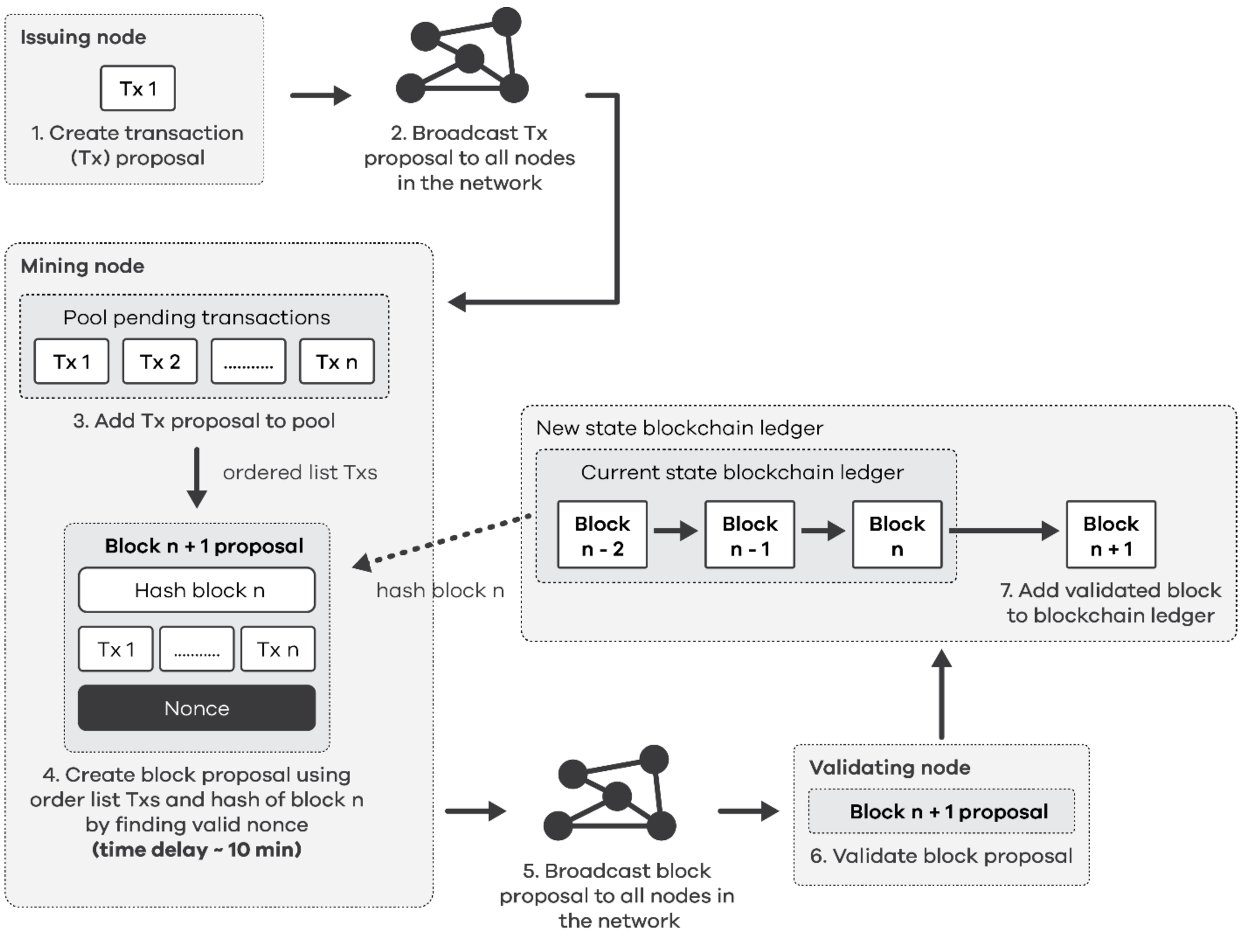

Issuing node
Tx 1
1. Create transaction (Tx) proposal
2. Broadcast Tx proposal to all nodes in the network
Mining node
Pool pending transactions
Tx 1
Tx 2
....
Tx n
3. Add Tx proposal to pool
ordered list Txs
Block n + 1 proposal
Hash block n
Tx 1
....
Tx n
Nonce
hash block n
4. Create block proposal using order list Txs and hash of block n by finding valid nonce (time delay ~ 10 min)
New state blockchain ledger
Current state blockchain ledger
Block n - 2
Block n - 1
Block n
Block n + 1
7. Add validated block to blockchain ledger
5. Broadcast block proposal to all nodes in the network
Validating node
Block n + 1 proposal
6. Validate block proposal

Figure 20: Standard consensus mechanism using Proof-of-Work as consensus protocol.

Forking

Because of network latency, different nodes might approve and append different blocks to their own copy of the blockchain ledger. As a result, different versions of the ledger can reside on the network (Tschorsch & Scheuermann, 2016). These different versions are called forks. Malicious nodes can exploit this issue by trying to build a longer malicious fork of validated blocks than the honest fork of the honest nodes (Dhillon et al., 2017; Eyal & Sirer, 2014). To be successful, the malicious node thus has to outperform other nodes. In other words, it has to solve crypto puzzles to mine blocks at a higher rate than other nodes (Narayanan et al., 2016). If the nodes that work on the honest fork mine blocks at rate N, the malicious node has to mine blocks at rate 2N to overtake the honest fork. The probability that a malicious node succeeds in double spending drops exponentially with the number of validated blocks after the initial transaction (Nakamoto, 2008; Narayanan et al., 2016). The computational power and thus cost to do so because of the mining process, quickly outweighs the potential benefits of double spending. The computational power of the majority will outperform any computational power of a malicious minority party. This is often mentioned as the 51% attack. If the malicious party has 51% of computational power in the network, it has a higher probability to mine the next block. In theory, this would enable to malicious party to create the longest fork. In general, six new validated blocks after the block of the initial transaction are sufficient to avoid double spending. Each time a new block is added, the previous blocks are essentially revalidated. A fork can also occur if honest nodes validate different blocks. This is resolved by adding the transactions of the block to a new block.

Practical Byzantine Fault Tolerance (PBFT)

PBFT, first proposed by Castro & Liskov (1999) is a BFT consensus protocol based on voting. The basic assumption of PBFT is that there are two types of nodes: leaders and ordering nodes of which all identities are known. In PBFT time is divided into rounds. For each round a different node is selected as leader that can propose a new block of ordered transactions. Because PBFT is a generic BFT consensus protocol, in this paragraph the use of PBFT is made specific for a blockchain-based architecture. The ordering process starts when invoking nodes send final transaction proposals to the ordering nodes. Next, the leader of the round orders transaction proposals stored in a container based on timestamp and creates a block proposal containing an ordered list of transactions. The leader creates a pre-prepare message containing the block proposal as shown in Table 11, signs the message with its PRleadernode and sends the message to all other ordering nodes identified by their PU in an access control list. Each ordering node checks correctness of the block proposal by checking if the transactions are ordered correctly based on timestamp and the signature originates from the leader node. Next, if correct the ordering node (e.g., X) stores the block proposal and creates a prepare message that contains the hash of the block proposal as shown in Table 12. It signs the message and sends the message to all ordering nodes. Each ordering node checks correctness of the signature of the prepare message for each message received from other ordering nodes. If an ordering node (e.g., Y) receives a correct prepare message with the same block proposal hash from at least two-thirds of the ordering nodes to ensure BFT, it creates a commit message containing the block proposal hash as shown in Table 13. It signs the message with and sends the message to all ordering nodes. If an ordering node receives a correct commit message with the same block proposal hash from at least two-thirds of the ordering nodes, it sends the block proposal to all validating nodes that have a copy of the blockchain ledger. All parties that have a copy of the blockchain ledger are validating nodes. If a validating node receives the same block proposal from two-third of the ordering nodes, the block is deemed valid and the validation process starts.

Table 11: Pre-prepare message.

Pre-prepare message			
Ledger ID	Block proposal (Transaction 1, timestamp 1; Transaction 2, timestamp 2)	Leader node PU	Leader node signature

Table 12: Prepare message.

Prepare message			
Ledger ID	Block proposal hash	Ordering node PU	Ordering node signature

Table 13: Commit message.

Commit message			
Ledger ID	Block proposal hash	Ordering node PU	Ordering node signature

The ordering process can be summarized as follows:

1. The invoking node sends the final transaction proposal to ordering nodes;
2. The leader node of the round orders transactions and creates a block proposal;
3. The leader node creates a pre-prepare message by signing the message with PRleadernode and sends the signed message to all ordering nodes;
4. Each ordering node checks correctness of the block proposal and if the signature is from leader node;

5. If correct, an ordering node (e.g., X) stores the block proposal and creates a prepare message containing the hash of the block proposal, signs the message and sends the message to all ordering nodes;
6. Each ordering node check correctness of the signature of the prepare message received from each ordering node;
7. If an ordering node (e.g., Y) receives a correct prepare message with the same block proposal hash from at least two-thirds of the ordering nodes, it creates a commit message containing the block proposal hash, signs the message and sends the message to all ordering nodes;
8. Each ordering node checks correctness of the signature of the commit message received from each ordering node;
9. If an ordering node receives a correct commit message with the same block proposal hash from at least two-thirds of the ordering nodes, it sends the block proposal to all validating nodes that have a copy of the blockchain ledger the block proposal belongs to.
10. If a validating node receives the same block proposal from at least two-thirds of the ordering nodes, it starts the validation process.

Comparison consensus protocols
The analysis has shown that the consensus protocol heavily affects scalability. Because scalability is one of the requirements for the blockchain-based architecture, in this paragraph a comparison is made between the Proof-of-Work and PBFT consensus protocols. The comparison consists of four properties: network topology, transaction throughput, number of nodes participating in the consensus protocol and delay and is shown in Table 14.

<u>Network topology</u>
Because Proof-of-Work is based on probability, the protocol is not prone to Sybil attacks. Who can make a block proposal is not voted on. The node that finds a valid block proposal gets to propose the next block. Not the number of nodes but the computational power of nodes affects which node wins (Vukolić, 2016). Therefore, Proof-of-Work can be used in both permissionless as permissioned network topologies. On the contrary, PBFT is prone to Sybil attacks as it is based on voting. In a permissionless network in which any node can participate in the consensus protocol this would allow malicious nodes to gain control over the consensus protocol and make fraudulent transactions by creating many nodes as for each round a leader is selected. Therefore, PBFT can only be used in a permissioned network topology in which the identities of the nodes are known to control which nodes can participate in the consensus protocol.

<u>Transaction throughput</u>
Proof-of-Work introduces a delay to find a valid block proposal by requiring computational work to be done by a node. On average it takes 10 minutes to find a valid block proposal. This overcomes double spending by forking the blockchain ledger. As a consequence, the transaction throughput is limited to only 8 – 15 transactions per second. PBFT does not include a delay as no computational work has to be done. As a result, the transaction throughput is in the tens of thousands of transactions per second.

<u>Number of nodes participating in consensus protocol</u>
Because Proof-of-Work does not select a leader and does not need messages between nodes to determine if a block proposal is valid, the number of nodes that can participate is unlimited. PBFT however does select a leader and uses messages between nodes to determine if a block proposal is valid to overcome Byzantine failures. Because the number of messages required is given by n^2 with n being the number of nodes, performance can be reduced up to sixty percent in terms of transaction throughput and latency can increase if many nodes participate (Rüsch, 2018). Dinh et al. (2018) find that PBFT is scalable up to 32 nodes without major performance issues.

Delay
Finality of consensus means that once a block is appended it cannot be invalidated by for example forking as the longest chain is deemed correct (Tasca & Tessone, 2018). Proof-of-Work does not guarantee consensus finality as all nodes can propose blocks. Due to latency in the network this can result in forking of the blockchain ledger that can be used by malicious nodes to double spend a transaction. To overcome the double spend problem, the general rule is to wait for six confirmations which means that six new blocks are appended to the blockchain ledger. Because on average every ten minutes a block is proposed, delay can reach up to sixty minutes. In contrast, PBFT does provide consensus finality as forking is not possible because only one block can be proposed per voting round. Therefore, there is no delay relating to confirmation time. The only delay that occurs relates to latency in the peer-to-peer network.

Table 14: Comparison consensus protocols.

Protocol property	**Proof-of-Work**	**PBFT**
Network topology	Permissionless or permissioned	Permissioned, identities of nodes are needed
Transaction throughput	8 – 15 transactions per second	10,000+ transactions per second
Number of nodes participating in consensus protocol	Unlimited	Limited, tested up to 32 nodes
Delay	60 minutes	Up to a few seconds due to latency in network

4.4.3 Sub conclusion consensus mechanism component
For the consensus mechanism, two sub components with several design options were identified and discussed in this section. Table 15 provides an overview of the component. The next section discusses the application component of a blockchain-based architecture.

Table 15: Overview of consensus mechanism component.

Core component	**Sub components**	**Sub sub components**	**Design alternatives**	**Option components**
Consensus mechanism	Transaction process		Order – execute	
			Execute – order - validate	
	Protocol		Proof-of-Work	
			PBFT	

4.5 Application component
The fourth and final core blockchain architecture component is the application. What is being stored on the blockchain ledger is determined by the application component (Dinh et al., 2018). It implements all relevant business logic. The first application of blockchain technology was the use of cryptocurrency that enables transfer of monetary funds between nodes in a peer-to-peer network without a trusted intermediary (Xu et al., 2016). A second type of application is smart contracts. Whilst cryptocurrency is in essence a smart contract, the term smart contract is often used for more advanced applications that go beyond the transfer of cryptocurrency. Examples of these smart contracts are: management of licenses, certificates, titles to ownership and provenance of goods. Therefore, in this research a distinction is made between simple cryptocurrency and smart contracts that can be used to implement any kind of business logic.

A user interacts with the blockchain ledger via the application component, for example via Application Programming Interfaces (APIs) and user interfaces. APIs and user interfaces that support interaction are left outside the scope of the research. Whenever a user interacts with the blockchain ledger via the application, it is assumed that some form of user interface is utilized.

4.5.1 Cryptocurrency

Many blockchain-based architectures include a native token, often referred to as cryptocurrency, to be used as currency within the architecture (Xu et al., 2016). Use can include payment for other digital assets that can trigger smart contract functionality or reward for work performed in the consensus protocol. Well known examples are Bitcoin and Ether (ETH) in the Ethereum blockchain.

4.5.2 Smart contract

The term ‘smart contract’ implies that a contract with some form of self-thinking mechanism is used. However, this term is a bit misleading. A smart contract is not necessarily a contract in terms of traditional legally binding paper-based contracts. Legally binding means that agreements on the relationship between parties that have legally binding effects are defined in a contract (Governatori et al., 2018). A smart contract can also be a formalization of a procedure. In addition, the contract does not have a mind of its own. Instead, the term smart contract is derived from the possibility of automatic execution based on a set of pre-defined if-then-else conditions that are programmed in computer code. The code in a smart contract is deterministic, which means that the same input will always generate the same output. Szabo (1997, p. 2) was one of the first to introduce smart contracts and defined its potential as: “Smart contracts utilize protocols and user interfaces to facilitate all steps of the contracting process. This gives us new ways to formalize and secure digital relationships which are far more functional than their inanimate paper-based ancestors.”. He argued that smart contracts would lead to a reduction in costs. In addition, the number of disputes between contractual parties could be reduced (Szabo, 1997). The concept of smart contracts is thus not new. The arrival of Electronic Data Interchanges (EDIs) to provide standards and protocols for the automation of transactions led to a first application of smart contracts. However, only a few types of transactions could be translated into smart contracts, reducing the potential due to costs.

A smart contract consists of several elements: state, functionalities and validation rules. In the next paragraph, a life cycle analysis is used to discuss these elements in more detail.

Life cycle

A smart contract, similar to a paper-based contract has a life cycle. The working of a smart contract can thus best be described by following its life cycle. For smart contracts, four phases of the life cycle can be distinguished. In phase one, a smart contract is developed. In the second phase, the contract awaits execution (Governatori et al., 2018). The third phase executes the contract. In the fourth phase, the contract is finalized (Sillaber & Waltl, 2017).

<u>Development of smart contract</u>

The first phase in the life cycle of a smart contract is the actual agreement on terms and conditions and development of the code. The conditions in a smart contract are defined by the parties involved. These can be similar to conditions in paper-based contracts. The conditions can be specified using if-then-else statements to build the code. A well-known blockchain platform that provides services for smart contracts is Ethereum. It uses the Solidity code language to build a smart contract. After agreement the code is digitally signed by the parties and is deployed on the blockchain ledger via a transaction. The deployment on the blockchain ledger shows the advantage of blockchain technology. Once deployed, the smart contract becomes immutable as it is part of an immutable transaction. This ensures correct execution. Multiple independent instances of a contract can exist. Each instance has its own state.

Awaiting execution of smart contract
A deployed smart contract does not necessarily execute immediately. It resides on the blockchain in a freeze state until invoked. A smart contract has a virtual address. This address can be used to store value in the smart contract via a transaction that is released once executed.

Execution of smart contract
A smart contract executes automatically if invoked by an event. These events are either transactions or messages that are sent by a node that wants to execute the contract. A smart contract can trigger other contracts to enable modular implementation of business logic, similar to object-oriented programming. A transaction proposes a state update by calling a function (that implements business logic) of the corresponding smart contract. Because smart contracts are executed on the node itself, without additional solutions the contract cannot access data outside the copy of the blockchain ledger. This ensures the code is deterministic.

Finalizing the smart contract
The result of an executed smart contracts is a transaction and an updated state of the contract. These outcomes are stored on the blockchain after being validated via a consensus mechanism (Sillaber & Waltl, 2017).

4.5.3 Sub conclusion application component

For the application component, two sub components were identified and discussed in this section. Table 16 provides an overview of the component. The next section concludes this chapter.

Table 16: Overview of application component.

Core component	Sub components	Sub sub components	Design alternatives	Option components
Application	Cryptocurrency			
	Smart contract			

4.6 Conclusion chapter 4

In this chapter, core blockchain technology architecture components were identified and for each identified component its relevance, technical functioning and design alternatives were discussed to answer sub question 3: ‘What are the core blockchain technology architecture components?’. Four core components for a blockchain architecture were identified: 1) network configuration, 2) data structure and storage, 3) consensus mechanism and 4) application. Table 17 provides an overview of the core components. The network configuration component is used to determine which parties in which role can participate in the peer-to-peer network by controlling permissions. Two types of configuration topologies can be distinguished: 1) permissionless, in which there is no control over permissions for read or write permissions and participation in consensus mechanism and 2) permissioned in which read or write permissions and permission to participate in the consensus mechanism are managed via additional identity and permission management components.

The data structure and storage component determines how data is stored, either on-chain on the blockchain ledger or off-chain in a separate data storage with a reference to the data stored on-chain. It also determines whether al data is stored on a single or multiple blockchain ledgers. In addition, it determines the way in which transactions are stored and update the state of the blockchain ledger. Two types of transaction models are identified: 1) UTXO which is stateless and 2) account-based which explicitly stores a state in accounts. Data security can be realized via either data encryption or Zero-Knowledge proof.

The third component identified component is the consensus mechanism that is used to determine the way in which parties reach agreement on the order of transactions in the blockchain ledger and thus state of the blockchain ledger. Two types of transaction processes can be distinguished: 1) order – execute that requires smart contracts to be deterministic which reduces flexibility as all nodes have to invoke the same smart contract for each transaction to confirm validity and thus reach consensus and 2) execute – order – validate which allows smart contracts to be non-deterministic as not all nodes have to invoke the same smart contract for each transaction. This also improves scalability. In addition, several consensus protocols are identified to be used in the consensus mechanism, of which Proof-of-Work and PBFT are currently the most prominent and well-developed. An important trade-off between Proof-of-Work and PBFT is that Proof-of-Work has high scalability in terms of nodes, but limited scalability in terms of transactions (maximum throughput of 8 to 15 transactions per second) due to a built-in delay of on average 10 minutes in case of Bitcoin for generation of new blocks using a cryptographic mining challenge in order to overcome double spending by forking of the blockchain ledger. PBFT has high scalability in terms of transaction throughput as it does not use mining but voting for generation of blocks. However, as it is subject to Sybil attacks, it is only applicable in a permissioned network configuration where the nodes that are allowed to participate in the consensus protocol are regulated. In addition, PBFT is subject to low scalability in terms of nodes participating.

The fourth core component is the application. This component implements the business logic and provides the functionality used to alter the state of the blockchain ledger by creating output transactions based on input provided by another transaction that is created by a node that wants to add a new transaction to the blockchain ledger. The contract can be used to represent the state of a physical asset or a native digital asset (e.g. cryptocurrency token). The core blockchain technology components are used as basis for the blockchain-based architecture design as presented in chapter 5.

Table 17: Overview core blockchain technology architecture components and design options

Core component	Sub components	Sub sub components	Design alternatives	Option components
Network configuration	Peer-to-peer network			
	Topology		Permissionless	
			Permissioned	Identity management
				Permission management
Data structure and storage	Blockchain ledger	Transaction model	UTXO-based	
			Account-based	
		Ledger structure	Single ledger	
			Multiple ledger	
	Off-chain storage			
	Data security		Encryption	Decryption key management
			Zero-Knowledge Proof	
Consensus mechanism	Transaction process		Order - execute	
			Execute – order - validate	
	Protocol		Proof-of-Work	
			PBFT	
Application	Cryptocurrency			
	Smart contract			

5. Architecture design

Chapter 3 defined design requirements for the blockchain-based architecture design. In the previous chapter 4, core components of a blockchain-based architecture were identified. In this chapter, the actual architecture is designed to support each use case in order to answer sub question 4: '*What does the blockchain-based architecture design look like for each of the critical trade document use cases?*'. The four core components as identified in chapter 4 form the basis of the architecture design. Using the design requirements from chapter 3, for each component design choices are made. Design options for each component are compared in order to select the most suitable option. The blockchain-based architecture design consists of four core components: 1) network configuration component, 2) data structure and storage component, 3) consensus mechanism component and 4) application component. For each use case, the blockchain-based architecture design is presented in a separate section.

Because this chapter develops a blockchain-based architecture design for each use case, redundant argumentation is present between different use cases. For readability purposes, each sub section includes a conclusion presented in an orange box.

Table 18: Crucial trade document use cases.

Use case	Description
I	Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration
II	Automated generation of an import declaration by aggregating already exchanged pro forma invoice data
III	Transfer title to the goods using an electronic negotiable Bill of Lading

The deliverable of this sub question is a blockchain-based architecture design for each use case that fulfills the design requirements. The architecture design serves as input for the demonstration sub question 5 (chapter 6) and evaluation sub question 6 (chapter 7).

5.1 Blockchain-based architecture of use case I

This section presents the blockchain-based architecture that supports use case I: Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration. Sub section 5.1.1 presents the network configuration component. The data structure and storage component is presented in sub section 5.1.2. Sub section 5.1.4 presents the consensus mechanism component and section 5.1.3 presents the application component. Sub section 5.1.5 concludes the section and provides an overview of the architecture.

5.1.1 Network configuration

In use case I a blockchain-based architecture is used to enable business-to-business exchange and business-to-government exchange of a pro forma invoice document (see UC1-req. 1 and 2).
A typical trade lane for a shipment consist of the following businesses: consignor, consignee, Freight Forwarders (export and import side) and carrier. To enable business-to-business exchange with end-to-end visibility (see UC1-req. 1), each trade lane business is represented as node[16] in a peer-to-

[16] Note that each trade lane party being a node in the blockchain network is not realistic. For example, it cannot be expected that small consignors or consignees are capable of maintaining their own node due to the technological complexity and costs. These businesses would normally rely on an information system (e.g. Enterprise Resource Planning [ERP]) provided by a software vendor to exchange documents. However, this research does not include the interoperability aspect of these systems and the architecture. For sake of simplicity

peer blockchain network in which a blockchain ledger is distributed. Each node has its own copy of the blockchain ledger. In addition, to enable business-to-government exchange (see UC1-req. 2) customs authorities are represented by nodes in the same blockchain network and also have a copy of the blockchain ledger. As international shipping consists of millions of shipments that go through thousands of trade lanes, the network in theory consists of thousands of nodes representing businesses and customs authorities.

A blockchain-based architecture to exchange trade documents is a highly technical and organizational complex endeavor with many businesses and public authorities involved (Rukanova, Wigand, van Stijn, & Tan, 2015). Governance of the architecture is therefore critical to successful implementation. This cannot be realized by a single party as it would conflict with the requirement of not having a single authority in control and ensuring openness to businesses willing to use the architecture to exchange documents. Therefore, the architecture relies on a consortium consisting of both businesses and public authorities that together govern the architecture. In previous research on the data pipeline concept it was proposed that such consortium should consist of a representation of businesses involved, customs authorities, technology companies and international organizations (Klievink et al., 2016). For this use case, the consortium would thus consist of a representation of consignors, consignees, freight forwarders and carriers, several customs authorities, a technology company and international organizations like the World Customs Organization or EU DG TAXUD. These parties together set out the governance rules for the architecture together. This ensures that no single authority is in control which satisfies requirement UC1-req. 6. An in-detail design of the governance structure is outside the scope of the research.

Topology
Within blockchain technology, two types of network configuration topologies are distinguished: permissionless and permissioned (see section 4.2). In this paragraph, the chosen topology is presented. Besides trade lane businesses and customs authorities, there is no need for other parties that are not involved in a shipment to be part of the network and have a copy of the blockchain ledger. Even if confidentiality of exchanged documents would be guaranteed, for example via encryption, business might not be willing to make the encrypted document publicly available in a permissionless blockchain network. Also, in a permissionless network nodes are anonymous. This makes it difficult to decide which parties are authorized to access documents (see UC1-req 3.). Customs authorities might have difficulties to determine if an exchanged pro forma invoice is the correct document exchanged by the actual consignor, which would conflict with the need for integrity (see UC1-req. 4). It would also be difficult to identify businesses in case of disputes. Therefore, to manage identities and read and write permission to the blockchain ledger, identity and permission management are needed which can only be realized in a permissioned network configuration topology.

In addition, the architecture should be able to handle high volumes of document exchanges in real-time (see UC1-req. 6). As discussed in section 4.4, in a permissionless network any party can become a node at their own discretion, thus rendering nodes anonymous. To overcome the double spend problem in anonymous peer-to-peer networks, computational heavy consensus protocols like Proof-of-Work that introduce an economic incentive to keep nodes honest via mining are needed (Ølnes et al., 2017; Wüst & Gervais, 2017). The inevitable limited scalability of eight to fifteen transactions per second and delay of up to sixty minutes Proof-of-Work is limited by, makes this consensus protocol infeasible for a blockchain-based architecture for the exchange of a pro forma invoice document. For example, to handle exchange of pro forma invoices within the Netherlands alone a scalability of at least ten transactions per second is needed (see UC1-req. 6). From section

a higher level of abstraction is chosen in which each business is represented by a node in the network without addressing which party maintains the node.

4.4 it also follows that consensus protocols based on voting do provide sufficient scalability in terms of transaction throughput, however at the cost of scalability in terms of nodes participating in the consensus protocol. In a blockchain network consisting of thousands of trade lanes businesses and customs authorities represented by nodes, this means that only a selected group of nodes can participate. To manage which nodes can participate, permission management is therefore needed. This can only be realized in a permissioned network configuration topology.

In conclusion, to enable identity and permission management, and use of a consensus protocol that provides sufficient scalability, the architecture uses a permissioned network configuration.

In this sub section, both identity and permission management components are presented.

Identity management (PKI)
To assign permission rights to parties, identification and authentication is needed to ensure the correct party gets permission. Therefore, the architecture design uses identity management based on Public Key Infrastructure. Within the field of data security, Public Key Infrastructure (PKI) is the international standard for secure transfer and storage of data in networks (Buchmann, Karatsiolis, & Wiesmaier, 2013). Box 4 provides the basics of PKI.

Box 4: Public Key Infrastructure basics.

Public Key Infrastructure (PKI)
PKI provides services to manage the binding of the Public Key (PU) of an asymmetric key pair (see Box 2) to a real-world identity. This guarantees that digital signatures can only be correctly created by the real-world entity. Without PKI, it cannot be trusted that the entity that signed a message is the actual real-world entity. PKI consists of multiple concept that are briefly discussed in this box.

Public key (PU) identity certificate
PKI relies on the use of public key (PU) identity certificates. These certificates bind the identity of an entity to the PU of the asymmetric key pair. Entities can range from individuals to software packages. A standard often used for PU identity certificates is X.509. This standard specifies attributes such as identity of the issuer (issuer ID), identity of the holder (holder ID), and the PU. A trusted Certificate Authority (CA) is required to sign the certificate to guarantee correctness. The PU identity certificate is shared publicly in a certificate registry and can be used by other parties to encrypt messages with the certificate holder as recipient, or to verify digital signatures. The private key (PR) that corresponds to the PU identity certificate is kept secretly and securely stored by the certificate holder.

Certificate Authority (CA)
PU identity certificates are issued and managed by a trusted third party called the Certificate Authority (CA). The CA authenticates the identity of an entity registered via a Registration Authority, which is left outside the scope of this research. It issues a PU identity certificate that binds the identity to a PU. The certificate is signed with the CA's PR to ensure validity of the certificate. To verify the signature, the CA also issues a self-signed PU identity certificate containing the CA's PU. If the PU verifies both certificate signatures, the issued certificate is valid.

Certificate registry
All PU identity certificates that are issued by a CA are stored in a public registry. Parties can download these certificates if needed. The CA also maintains a registry of withdrawn certificates. Each time a signature has to be validated, the registry is consulted. In the remainder of this chapter, if the term PU is used, this also means that a certificate is presented and validated to authenticate the holder of the certificate and thus authenticate the message.

PU identity certificates
Requirement UC1-req. 5 states that the architecture should be open to businesses willing to participate in the business-to-business exchange of trade documents such as the pro forma invoice. No central authority should be in control of deciding who can or cannot be represented as node in the blockchain network and have a copy of the blockchain ledger or be able to make alterations to the ledger without consensus. The use of a single CA controlled by a central authority for the issuing of PU identity certificates for parties involved in international trade lanes would therefore not suffice as solution for identity management. Therefore, in the architecture design a distinction is made between external CAs and internal CAs that issue certificates.

External CA
For the identification of businesses in digital (semi)-public services, many countries have already implemented national electronic identities (eID) - or PU identity certificates - for businesses issued by national CAs. The Netherlands for example uses a system called eHerkenning (Pruksasri et al., 2014; Pruksasri, Van Den Berg, & Hofman, 2012). These national PU identity certificates enable businesses to use services of different (semi)-public authorities without registering for each service individually. According to Pruksasri et al. (2014), these national PU identity certificates can be used as means of identification and authentication in data pipelines that span international trade lanes. The architecture is a joined effort of both businesses and customs authorities and the central part of this use case is the pull of a pro forma invoice by a customs authority to piggyback for manual-cross validation purposes. This provides sufficient base to view the architecture as a digital (semi)-public service. External CAs reside outside the blockchain network and are not controlled by the consortium. By using an external national CA, businesses do not have to go through the registration process to obtain a PU identity certificate which reduces bureaucracy. In addition, as the external CA is controlled by public authorities, this ensures openness of the architecture to businesses to exchange trade documents. Therefore, the architecture design supports the use of externally issued PU identity certificates by national CAs. Businesses can use their own national PU identity certificate to gain access to the blockchain network. Which national PU identity certificates are supported is decided upon by the governing consortium.

Internal network CA
Next to an external CA, the architecture design also includes an internal network CA. A business that cannot rely on an externally issued PU identity certificate, like businesses operating in countries without national eID system, can be issued an PU identity certificate via the internal CA. The registration steps are left outside the scope of this research. Also, customs authorities require an PU identity certificate issued by the internal CA as it provides an additional role attribute ‘customs’. In addition, any party that requires access to the blockchain network with specific permission, for example consortium parties with administrative rights to maintain the architecture or permission to participate in the consensus mechanism, have to be issued an PU identity certificate by the internal CA to be identifiable. The internal network CA issues PU identity certificates with specific permissions and is therefore controlled by the consortium parties. The consortium parties together decide which (type of) party should be issued a certificate. As this control is a joined effort of consortium parties no single authority is in control of the issuance of PU identity certificates.

PU pseudonym identity certificates
The use of PU identity certificates enables real-world identification of businesses involved in transactions on the network. For example, the exchange of a pro forma invoice via a transaction identifies at least the consignor involved. However, as discussed in requirement UC1-req. 3, the architecture should provide confidentiality of businesses involved in a shipment. As a transaction is stored immutably in the blockchain ledger of which all businesses involved have a copy, the use of a PU identity certificate would not satisfy the requirement for confidentiality. The carrier for example would have access to the identity of the consignor, which might reduce the willingness of freight forwarders to participate in the architecture as they can be by-passed by the carrier. On the contrary, a business cannot be anonymous because in case of disputes or fraudulent activities public authorities should have the possibility to identity the real-world identity. Therefore, the architecture uses a second type of certificates: PU pseudonym identity certificates. These certificates are cryptographically derived from the PU identity certificate of the business that represents the real-world identity in such a way that only authorized parties can link the two certificates and thus identity the real-world identity behind the pseudonym identity. How this cryptographically derivation works is left outside the scope of this research. Business use the PU pseudonym identity certificates for signing transactions which allows the business to keep its real-world identity confidential (Benjumea, Lopez, Montenegro, & Troya, 2004). The PU pseudonym identity certificate are issued by an internal network Pseudonym Certificate Authority (PCA). Businesses can use their PU identity certificate to request a PU pseudonym identity certificate from the PCA. The PCA uses the PU identity certificate for identification and authentication and if authenticated, issues an PU pseudonym identity certificate that is cryptographically derived from the PU identity certificate. The cryptographic link between the PU identity and PU pseudonym identity certificate can only be solved by the PCA. Public authorities can request the PCA to provide the real-world identity if there is sufficient legal ground such as fraud investigations.

In the architecture, the use a separate PU pseudonym identity certificate for all actions within a single shipment such as a transaction to exchange a pro forma invoice should provide sufficient confidentiality of the identity of the businesses involved.

In conclusion, for identity management the architecture design uses PU identity certificates for access to the blockchain network. These certificates are either issued by an external CA (national eID CAs) or by an internal network CA. To provide confidentiality of businesses involved in a shipment and exchange of a pro forma invoice document, PU pseudonym identity certificates are used. These certificates are issued by an internal network PCA and derived from a PU identity certificate.

Permission management
To control which parties have permission to perform actions on the blockchain network, permission management is used. Three main types of permission are distinguished in a permissioned blockchain network configuration: 1) read permission, 2) write permission and 3) consensus participation permission (see section 4.2). For more granular permission management, the architecture design adds an additional permission type besides read and write permission: network access permission. As discussed previously, identity management distinguishes between PU identity certificates to access the blockchain network and PU pseudonym identity certificates to participate in transactions. The use of read permission alone would not clearly show a distinction between access to the blockchain network and read or write permission to the blockchain ledger within the network. Having network access permission does not also imply read and write permission of the blockchain ledger. Thus, the architecture design includes four types of permission: 1) network access permission, 2) read permission, 3) write permission and 4) consensus participation permission.

Table 19 provides an overview of the identity and permission management of the architecture design and includes which certificates are used to manage which type of permission and which CA is responsible for issuing the certificate. For network access permission a PU identity certificate issued by an external CA or the internal network CA is used to manage permission. Read permission is managed using PU pseudonym identity certificate issued by the internal network PCA for businesses and PU identity certificates issued by the internal network CA for customs authorities. Write permission is managed using PU pseudonym identity certificates. Permission to participate in the consensus mechanism is managed using PU identity certificates issued by the internal network CA. For readability purposes, how the different types of permissions are managed (e.g., via Access Control Lists) is discussed in more detail in the respective components. Blockchain ledger read and write permission are discussed in section 5.1.2 on the data structure and storage component. Consensus participation permission is discussed in section 5.1.3 on the consensus mechanism component. For network access permission, control of which party has permission is simple. Any party with a valid PU identity certificate, whether issued by a supported external CA or internal network CA is granted access to the blockchain network.

In conclusion, for permission management the architecture design uses four types of permissions: network access, read, write and consensus participation. Any party with an PU identity certificate has network access permission.

Table 19: Overview identity and permission management architecture design use case I.

Party / Permission type	**Trade lane business node**	**Customs authority node**	**Other party node**
Blockchain network access permission	PU identity certificate issued by any supported external CA or internal network CA	PU identity certificate issued by internal network CA	PU identity certificate issued by internal network CA
Blockchain ledger read permission	PU pseudonym identity certificate issued by internal network PCA	PU identity certificate issued by internal network CA	Does not apply
Blockchain ledger write permission	PU pseudonym identity certificate issued by internal network PCA	Does not apply	Does not apply
Blockchain network consensus permission	PU identity certificate issued by internal network CA	PU identity certificate issued by internal network CA	PU identity certificate issued by internal network CA

In the remainder of this chapter, for readability purposes PU identity certificates are represented as PU and PU pseudonym identity certificates are represented as pseudo-PU. If the term PU or pseudo-PU is used, this also means that the certificate is presented and validated to authenticate the holder of the certificate and thus authenticate the message. The next sub section presents the data structure and storage component of the architecture design.

5.1.2 Data structure and storage

In this sub section, the data structure and storage component of the architecture design is presented. The architecture should enable the business-to-business exchange of a pro forma invoice (see UC1-req. 1). In blockchain technology, transactions are added to blocks that are appended to a blockchain ledger that is distributed between nodes in order to exchange data. Therefore, in this architecture design transactions are used to exchange the pro forma invoice. This enables both businesses and customs authorities that have a copy of the blockchain ledger to retrieve the document. However, the architecture should not have a centralized document storage (see UC1-req. 5). If the document is stored on the blockchain ledger, all nodes that have read permission and thus a copy of the ledger have access to the pro forma invoice document. Even if the document would be encrypted, storing it on the blockchain ledger might reduce the willingness of the consignor to exchange. Whilst documents are stored encrypted and thus no physical access is possible without using a decryption key, in theory a malicious party could still gain access given enough time. In addition, the pro forma invoice document is already stored in the consignor's legacy Enterprise Resource Planning (ERP) system or document cloud storage. To reduce complexity, the document is therefore not stored on the blockchain ledger as this can be seen as a form of centralized storage as each node has a copy of the ledger. This would conflict with the requirement for decentralization (see UC1-req. 5). Instead, the architecture uses an additional off-chain document storage to store the pro forma invoice document. This document storage is controlled by the consignor and can for example be its ERP or document cloud storage. The architecture only stores a reference to the pro forma invoice document on the blockchain ledger. To ensure integrity of the exchanged document (see UC1-req. 4), a hash pointer that contains the hash of the document is used as reference. The hash pointer also allows a party to identify the location of the document in the off-chain document storage. Table 20 shows the data storage of the architecture. The consortium parties would have to specify how the off-chain document storage is connected to the network such that parties can retrieve the document using the hash pointer. The assumption is made that such specification, for example via Application Programming Interfaces (APIs) is available.

Table 20: Data storage architecture design use case I.

Data storage	
Off-chain document storage	*On-chain blockchain ledger*
Pro forma invoice document (e.g., .pdf)	Document reference (hash pointer)

Ledger structure

Whilst the architecture should be open to businesses willing to exchange trade documents like the pro forma invoice, there is no clear reason why a business should have access to all documents from all shipments. If the architecture would rely on a single blockchain ledger for all shipments this could reduce the willingness of businesses to use the architecture for the business-to-business exchange. The underlying reason is that from a technical perspective a node having to maintain a ledger containing all document references can have a negative effect on performance of the blockchain-based architecture concerning scalability as more storage space is required. A business might not be willing to participate in such architecture as this would result in unnecessary higher cost. It is not relevant for a random business to keep a blockchain ledger that contains document references of millions of shipments it is not involved in. Therefore, this architecture design uses a multiple blockchain ledger structure, in which each trade lane has its own dedicated ledger as shown in Figure 21. Each blockchain ledger thus contains documents from one or more shipments within the same trade lane. To manage which parties have read or write permission to a specific blockchain ledger, the pseudo-PU of businesses or PU of customs authorities involved in a shipment within the trade lane should be stored in state of the ledger. Only parties specified in the state have read or write permission[17].

[17] The technicalities of how a node with permission receives a copy of the blockchain ledger for a specific trade lane is left outside the scope of this research.

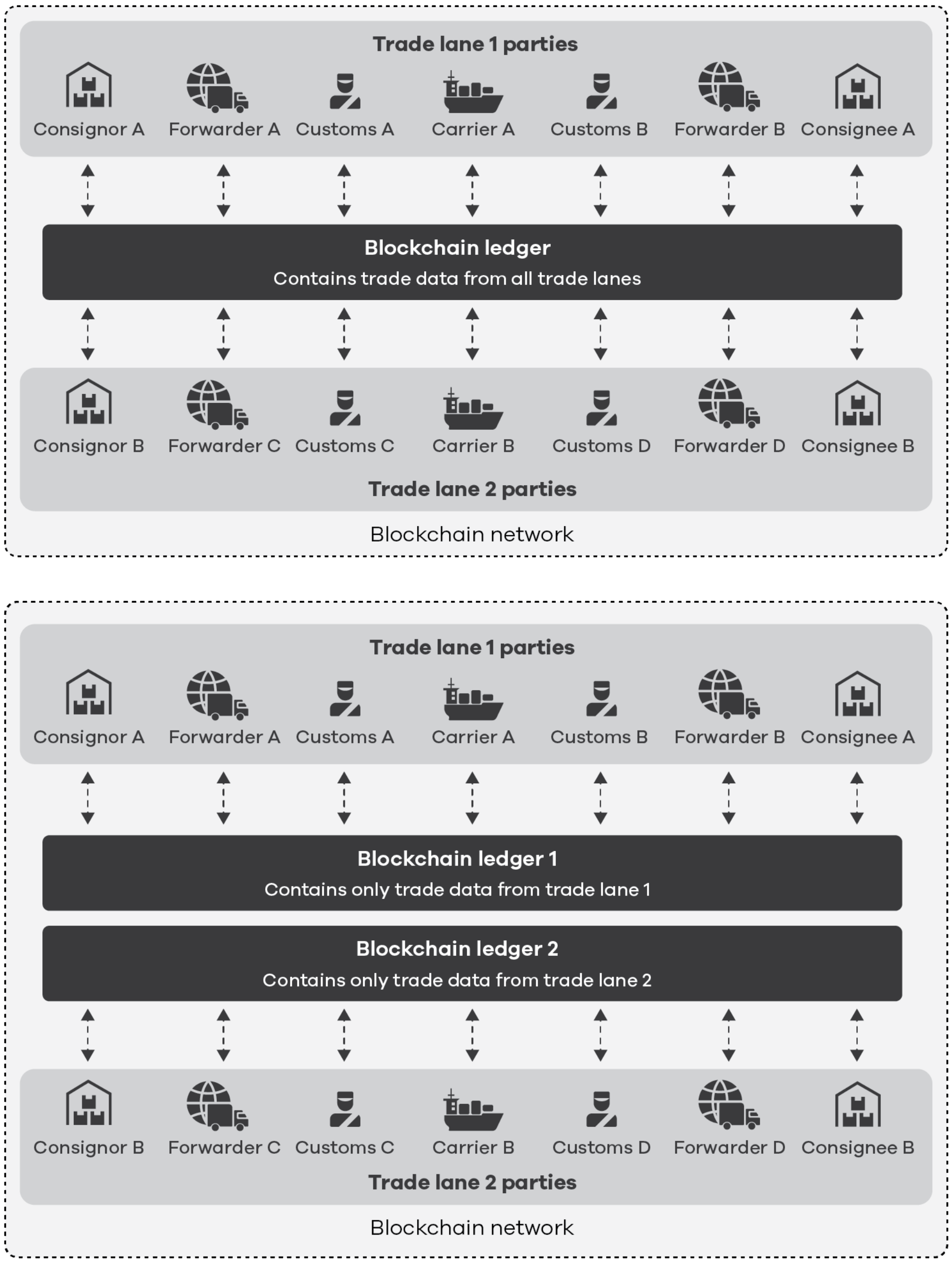


Figure 21: Single blockchain ledger (top) versus multiple blockchain ledger structure with ledger per trade lane (bottom).

Transaction model

Within the architecture design the assumption is made that documents related to a shipment are not loosely exchanged via transactions. Instead, documents are appended to a smart contract that represents a shipment by invoking the contract via a transaction. This contract is referred to as the smart shipment contract[18]. Each shipment has its own smart shipment contract. It allows for implementation of all related business logic needed for the exchange of a documents (e.g., the pro forma invoice). To that extent, the contract has a state that keeps track of at least the pro forma invoice document reference, a shipment reference number to link the pro forma invoice to a specific shipment and the PU or pseudo-PU identities of parties involved in the shipment – both businesses and customs authorities – including their role (e.g., consignor or customs authority on import side) to manage read and write permission of the blockchain ledger on which the shipment is stored.

[18] How the smart shipment contract is managed is left outside the scope of the research. The Freight Forwarder or carrier most probably manages the contract. The initial state of the contract can be based on the pro forma invoice document as it contains all relevant data.

To track updates to the state of the smart shipment contract, either an Unspent Transaction Output (UTXO) or account-based transaction model can be used. UTXO is mostly used in context of cryptocurrencies. However, for more complex smart contracts, account-based is more relevant. It allows for faster and easier updating of the state of the smart contract as the blockchain ledger does not have to be queried each time a new transaction is created to calculate the state. Therefore, the architecture uses an account-based transaction model. The blockchain ledger thus consist of two parts as shown in Table 21. The first part is a blockchain that contains an ordered list of transactions stored in blocks that represent smart shipment contract state changes to append a pro forma invoice document. The second part is the world state that reflects the latest state of the smart shipment contracts which includes the pro forma invoice document reference, shipment reference number and identities of parties involved in the shipment.

Table 21: Blockchain ledger architecture use case I.

Blockchain ledger	
Blockchain	*World state*
Ordered list of transactions stored in blocks that represent smart shipment contract state changes	Smart shipment contract states including pro forma invoice document reference, shipment reference number and identities of parties involved in the shipment.

Data security

As the pro forma invoice document contains sensitive data, for example the identity of consignor or consignee and invoice value, access should be restricted to ensure confidentiality (see UC1-req. 3). The use of multiple blockchain ledgers, one per trade lane, provides a first layer of confidentiality as only parties involved have read permission. However, the document is stored in an off-chain document storage. Read permission therefore does not ensure full confidentiality. A party that retrieves the hash pointer would still be able to access the document. In addition, even within a specific trade lane not all parties should have access to the pro forma invoice. For example, if the carrier has access to the document it can see detailed information on a shipment. This can result in liability issues as the carrier knows what is inside the shipment and can thus become liable for the goods (see UC1-req. 3). Also, if the carrier has access to the pro forma invoice and thus identity of the consignor, it can bypass the freight forwarder and offer services directly to the consignor. This can reduce the willingness of the freight forwarder to participate. Therefore, additional data security is needed. In context of blockchain technology, two types of data security are proposed: 1) traditional encryption and 2) Zero-Knowledge Proof. Traditional encryption is more complex as it requires distribution of decryption keys. However, Zero-Knowledge Proof requires computational effort, which may reduce performance. In addition, Zero-Knowledge Proof has not been successfully used in real-world application yet. Therefore, the architecture uses traditional encryption of the pro forma invoice document to ensure confidentiality. Before exchanging the pro forma invoice document reference, the consignor generates a symmetric encryption/decryption key and encrypts the document via E(document, encryption/decryption key) before storing it in the consignor's ERP or cloud storage. This enables flexible distribution of the decryption key as it is not linked to a specific party as would be the case if the document was encrypted via E(document, PUparty) which would only provide that party access. To manage distribution of the decryption key, decryption key management is needed.

Decryption key management

A party that needs access to the encrypted document can request the decryption key from the document owner (consignor). To that extent, a consignor has to first determine if the requesting party is authorized to access the document before the decryption key is provided. This requires an authorization and decision component. The authorization component holds a set of access control policies. These policies are a formalization of business rules. The decision component comes to a decision based on the output of the authorization component and provides the authenticated party access. Several methods to implement this component exist. Two common methods for authorization are role-based and attribute-based access control. Box 5 compares these two methods to find a method that is most suitable for the architecture design.

Box 5: Authorization and decision component methods comparison.

Comparison authorization and decision component methods

Role-based access control
Role-based access control (RBAC) defines roles to control access. Users are assigned one or more roles. Each role has a set of rights as determined by a rights management service. A rights management service manages the business rules that apply for granting authorization to data and controls the decision component (Park, Sandhu, & Ahn, 2001). The owner of data assigns rules to the data. For example, only users that are assigned the role = 'customs' have access. RBAC is the most dominant method to control access. An important limitation of RBAC is that for each condition a new role has to be defined. In interorganizational information systems management of these roles can become difficult (Hu, Ferraiolo, & Kuhn, 2006). For example, if encrypted data elements all have specific rights, it can quickly become complex to specify roles.

Attribute-based access control
In attribute-based access control (ABAC), access policies based on the attributes of users, data elements and the environment are used to decide if a user has sufficient access rights to access (Schläger, Priebe, Liewald, & Pernul, 2007). An example of a user attribute is the user's role. Attributes of data elements relate to its metadata. A data element can belong to a specific shipment by adding the shipment reference as shipment attribute. Also, an attribute can specify who is the owner of the data element. The decision component in ABAC takes as input an access policy and the attributes that belong to an access request. The policy is evaluated and a decision to grant or deny access is made afterwards.

XACML is an often used standard for the definition of access policies in ABAC (Schläger et al., 2007; Sukhodolskiy & Zapechnikov, 2017). This standard specifies an infrastructure for ABAC and consists of four components: 1) Policy Enforcement Point (PEP), 2) Policy Decision Point (PDP), 3) Policy Administration Point (PAP) and 4) Policy Information Point (PIP). Users interact with the infrastructure via the PEP. The PEP takes in access requests and enforces decisions made by the PDP. The PDP is the central component of ABAC which evaluates policies against provided attributes to come to a decision. If during evaluation a policy requires more attributes than provided, the PIP gathers these attributes from their respective storage. Polices are managed and stored in the PAP.

ABAC allows for more precise definition of access rules. In a blockchain network with multiple blockchain ledgers, that involve many trade lane parties and documents that require specific access policies, ABAC seems to fit best. Therefore, the architecture uses an adapted ABAC infrastructure which consist of a PEP, PDP, PAP and key store as shown in Figure 22. Each document owner manages its own ABAC infrastructure. A requesting party sends a decryption key request message to the ABAC infrastructure of the document owner (consignor) which includes the PU of the requesting party, reason for access, document reference and is signed by the requesting party as shown in Table 22. The pseudo-PU of the document owner is retrieved from the state of the smart shipment contract as in case of a pro forma invoice the document can only be exchanged by the consignor. The PEP checks the signature of the message to determine validity. If valid, the message is sent to the PDP. The PDP retrieves the access policy that belongs to the document reference from the PAP. How these access policies are defined is left outside the scope of this research. Van Engelenburg, Janssen and Klievink (2017) provide some examples of policies in context of shipping events. In this use case an example of a policy could be that the requesting party should have the role of customs authority in the trade lane of the shipment and has a valid reason to access the document. The PDP queries its own copy of the blockchain ledger to retrieve the state of the smart shipment contract that belongs to the document reference. If the requesting party PU in the request message is the same as the customs authority PU in the state of the contract and the customs authority has a valid reason, the access policy would be satisfied. In that case, the PDP sends a message to the PEP allowing the requesting party access. The PEP retrieves the decryption that belongs to the encrypted pro forma invoice document from the key storage. A message is created that sends the decryption key to the requesting party. In the message the decryption key is encrypted using the PU of the requesting party following E(decryption key, requesting party PU) as shown in Table 23. After receiving the message, the requesting party can decrypt the pro forma invoice document following D(decryption key, requesting party PR) and subsequently D(document, decryption key).

Table 22: Decryption key request message.

Decryption key request message			
Document reference	Reason	Requesting party PU	Requesting party signature

Table 23: Decryption key request return message.

Decryption key request return message			
Document reference	Encrypted decryption key	Document owner PU	Document owner signature

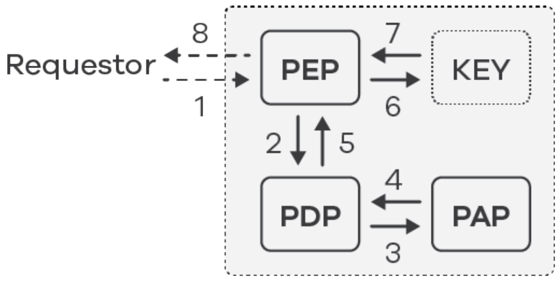


Figure 22: Adapted ABAC architecture for decryption key distribution.

In conclusion, the data structure and storage of the architecture design stores the pro forma invoice document in an off-chain document storage and stores only a document reference on the blockchain ledger. It uses a multiple blockchain ledger structure with a single ledger per tradelane. In addition, an account-based transaction model is used to store the state of the blockchain ledger. Data encryption is used to provide confidentiality of the pro forma invoice document.

The next sub section presents the consensus mechanism component of the architecture design.

5.1.3 Consensus mechanism

This sub section presents the consensus mechanism component of the architecture design to implement use case I. The component ensures that the nodes in the peer-to-peer network reach consensus on the order of transactions and thus state of the blockchain ledger. First, this sub section discusses which transaction process (see section 4.4) is used. The architecture should be open to businesses (see UC1-req. 5), which also implies that multiple smart shipment contracts that are implemented using different programming languages should be supported. However, transaction processes have to be deterministic. In case of the order - execute transaction process this means that all nodes have to invoke the same smart contract written in the same programming language in order to reach consensus on the order of transactions as each transaction contains an operation to update the world state and needs to be validated. Using the order - execute transaction process could therefore cause issues as the smart contract is not exactly the same. In addition, if every node has to invoke each smart contract for each transaction in order to update the world state, this could lead to scalability issues as contracts are invoked in sequence. The execute – order – validate transaction process splits consensus into three steps which disconnects the ordering of transactions and the invoking of smart contracts to validate transactions in order to reach consensus while still guaranteeing the processes to be deterministic. The consensus protocol is limited to the ordering of transactions and proposal of new blocks. This allows for the use of smart shipment contracts that use different programming languages and improves scalability as smart contracts do not have to be invoked in sequence. Thus, as the execute – order – validate transaction process provides more flexibility, the architecture uses this process in the consensus mechanism. In the remainder of this sub section, each step of the transaction process is discussed. The first step that is discussed is the execute step.

Execute

The first step of the transaction process is invoking the smart contract and confirmation of the transaction by confirming nodes. This step describes how an invoking node (consignor) can exchange a pro forma invoice document by appending it to smart shipment contract via a transaction. To ensure that the exchanged document is correct, the write set of the transaction has to be confirmed by a confirming node. Which node has to confirm the write set is specified for each function in the confirmation policy of the smart shipment contract. In case of appending a pro forma invoice document, the consignee can confirm correctness of the document as the pro forma invoice is a document sent to the consignee as notification of a shipment being shipped. The invoking node is always a confirming node as well. To create a transaction, the invoking node encrypts the pro forma invoice document with a freshly generated symmetric encryption/decryption key via E(document, encryption/decryption key). It stores the key in the key store and the access policy in the PAP of the node's ABAC infrastructure. For example, for a pro forma invoice document an access policy specifies that only a customs authority involved in a shipment and with a valid reason is authorized to access the document. The encrypted document is stored in the invoking node's document storage and a document reference (hash pointer) of the encrypted document is generated. The invoking node creates a preliminary transaction proposal including transaction ID, smart shipment contract ID, contract function, function input (document reference) and invoking node pseudo-PU as shown in Table 24. The transaction proposal is signed by the invoking node with pseudo-PRinvokingnode via E(transaction proposal hash, pseudo-PRinvokingnode). The invoking node invokes the smart shipment contract's 'append pro forma invoice' function using the transaction proposal. The output is a write set containing the document reference. Next, the invoking node creates a transaction proposal including transaction ID, smart shipment contract ID, contract function, function input, write set, confirming node 1 pseudo-PU, confirming node 1 signature and invoking node pseudo-PU as shown in Table 25. The transaction is signed by the invoking node. To ensure correctness of the document, the write set has to be signed by a confirming node. The invoking node sends the transaction proposal including decryption key to the confirming node based on the smart shipment contract confirmation policy to request confirmation. The confirming node retrieves the pro forma invoice document via the document reference, decrypts the document via D(document, decryption key) and checks the document for correctness. Next, the confirming node invokes the smart shipment contract using the transaction proposal. The output is a write set containing the document reference. The confirming node checks if the write set corresponds to the write set of the transaction proposal. It then creates a confirmation message including transaction ID, write set and confirming node 2 pseudo-PU as shown in Table 26. The message is signed by confirming node 2. The confirming node sends the message to the invoking node. Next, the invoking node creates a final transaction proposal using the signed write sets which includes timestamp, ledger ID, smart shipment contract ID, contract function, write set, confirming node 2 pseudo-PU, confirming node 1 signature, confirming node 2 pseudo-PU, confirming node 2 signature and invoking node pseudo-PU as shown in Table 27. The timestamp is required for the correct ordering of transactions. The ledger ID is used to separate transaction of different blockchain ledgers during ordering. The transaction is signed by the invoking node.

Table 24: Append pro forma invoice preliminary transaction proposal.

Append pro forma invoice preliminary transaction proposal			
Transaction ID	Smart shipment contract ID	Contract function	Function input
Invoking party pseudo-PU	Invoking party message signature		

Table 25: Append pro forma invoice transaction proposal.

Append pro forma invoice transaction proposal			
Transaction ID	Smart shipment contract ID	Contract function	Function input
Write set	Confirmation party 1 pseudo-PU	Confirmation party 1 signature	Invoking party pseudo-PU
Invoking party message signature			

Table 26: Append pro forma invoice transaction proposal confirmation message.

Append pro forma invoice transaction proposal confirmation message			
Transaction ID	Write set	Confirmation party 2 pseudo-PU	Confirmation party 2 signature
Confirmation party 2 message signature			

Table 27: Final append pro forma invoice transaction proposal.

Append pro forma invoice transaction proposal			
Timestamp	Ledger ID	Transaction ID	Smart shipment contract ID
Contract function	Write set	Confirmation party 1 pseudo-PU	Confirmation party 1 signature
Confirmation party 2 pseudo-PU	Confirmation party 2 signature	Invoking party pseudo-PU	Invoking party message signature

The execute step can be summarized as follows:

1. The invoking node encrypts the pro forma invoice document. The encrypted document is stored in the invoking node's document storage and a document reference (hash pointer) of the encrypted document is generated;

2. The invoking node creates a preliminary transaction proposal including transaction ID, smart shipment contract ID, contract function, function input (document reference) and invoking node pseudo-PU. The transaction proposal is signed by the invoking node;
3. The invoking node invokes the smart shipment contract using the transaction proposal. The output is a write set containing the document reference;
4. The invoking node creates a transaction proposal including transaction ID, smart shipment contract ID, contract function, function input, write set, confirming node 1 pseudo-PU, confirming node 1 signature and invoking node pseudo-PU. The transaction is signed by the invoking node;
5. The invoking node sends the transaction proposal including decryption key to the confirming node based on the smart shipment contract confirmation policy to request confirmation;
6. The confirming node retrieves the pro forma invoice document via the document reference, decrypts the document via D(document, decryption key) and checks the document for correctness;
7. The confirming node invokes the smart shipment contract using the transaction proposal. The output is a write set containing the document reference;
8. The confirming node checks if the write set corresponds to the write set of the transaction proposal.
9. The confirming node creates a confirmation message including transaction ID, write set, confirming node 2 pseudo-PU and confirming node 2 signature. The message is signed by confirming node 2;
10. The confirming node sends the message to the invoking node;
11. The invoking node creates a final transaction proposal including timestamp, ledger ID, smart shipment contract ID, contract function, write set, confirming node 1 pseudo-PU, confirming node 1 signature, confirming node 2 pseudo-PU, confirming node 2 signature and invoking node pseudo-PU. The transaction is signed by the invoking node.

This completes the execute step. The next step is the ordering process.

Transaction ordering
As previously discussed in sub section 5.1.1, the architecture must be scalable to handle at least ten transaction per second for the exchange of the pro forma invoice document within the Netherlands alone. Proof-of-Work can only handle between eight to fifteen transactions per second depending on implementation. This makes the protocol infeasible for use in the architecture design. Therefore, a different consensus protocol has to be selected for the architecture design. In the network configuration a permissioned network configuration was already selected to accommodate for a different consensus protocol. While in a permissioned blockchain network there is no need for a probabilistic consensus protocol as there is no need for an economic incentive to keep nodes honest, there is still a need to achieve Byzantine Fault Tolerance (BFT). Honest nodes that participate in the consensus protocol can still be subject to Byzantine failure. For example, a node can go down due to technical issues (Gramoli, 2017). A consensus protocol that provides sufficient scalability in terms of transaction throughput and ensures BFT is thus needed.

The architecture design therefore uses Practical Byzantine Fault Tolerance (PBFT) as consensus protocol. Section 4.4.2 introduced the basic working of the protocol. PBFT has been tested to have a transaction throughput of tens of thousands of transactions per second. A requirement for the use case is scalability of at least ten transactions per second for the exchange of pro forma invoice documents within the Netherlands alone (see UC1-req. 6). PBFT thus enables sufficient scalability in terms of transaction throughput with low latency. An issue that has to be addressed in context of PBFT is limited scalability in terms of nodes that can participate in the protocol. Dinh et al. (2018) find that PBFT is scalable up to 32 nodes without major performance issues. For the architecture design, the number of nodes participating in the consensus protocol should therefore be less than 32.

Within BFT, the number of nodes subject to Byzantine failure has to be less than one-third of the number of nodes to reach consensus, which in case of 32 nodes is ten nodes. The identities of nodes participating in the network are known, so the assumption can be made that Byzantine failures are not caused by malicious behavior and only technical of nature. The risk of not reaching consensus due to too many failures is therefore deemed minimal.

To participate, nodes need to have permission to participate. As in PBFT the identities of the nodes participating should be known to all other nodes, the architecture uses a public access control list that provides an overview of the PUs of nodes that participate. To that extent, a decision has to be made who governs the access control list and thus decides which nodes are given permission. The use of PBFT with a limited number of nodes that have permission to participate could in potential conflict with the requirement of having no central authority in control (see UC1-req. 5). The limitation could create the impression that a central authority is in control of the consensus protocol and can influence the ordering of transactions and thus world state of the blockchain ledger. It should however be noted that the only task of nodes participating in PBFT is the ordering of incoming transactions, propose blocks of ordered transactions and send block proposals to all other nodes in the network for that specific blockchain ledger. No further actions are performed on the transactions. This should provide businesses with sufficient trust that no central authority is in control of the world state of the blockchain ledger. To mitigate the potential conflict in context of who governs the access control list, the list has to be governed by multiple authorities. Within the architecture design the governing consortium would be the ideal choice as it consists of a representation of businesses, customs authorities, a technology company and international organizations. No single authority can thus decide which nodes have permission to participate.

To overcome the potential issue of Byzantine failure, the list of nodes participating should be sufficiently diverse. As participating in the consensus protocol requires additional effort regarding technical knowledge and costs, it is expected that the list mainly consists of larger consortium parties such as a technology company, international organizations such as the World Trade Organization or EU TAXUD, and some customs authorities or businesses that have the technological capabilities. For example, freight forwarders and carriers that benefit most from the architecture. The total sum of parties should be around thirty for the best performance of PBFT regarding transaction throughput and BFT. A detailed overview of the PBFT ordering process can be found in section 4.4.2. Because the architecture uses a multiple blockchain ledger structure and only parties that have a copy of the blockchain ledger should receive the transaction as part of a block proposal, the transactions are separated based on blockchain ledger.

This completes the transaction ordering process. The next step is the validation step.

Validation
The final step of the transaction process is validation of the transaction. Each node that holds a copy of the blockchain ledger acts as validating node. First, the validating node appends the block proposal (N) to the blockchain. To that extent, it adds the hash of the previous block (N-1) and the root hash the transactions to block N and appends the block to the blockchain as shown in Table 28. Next, the validating node checks if the transaction is valid by checking if the confirmation policy for the smart shipment contract ‘append pro forma invoice document’ function has been met. To do so, the confirmation policy is retrieved via the contract ID first. The confirmation policy states that the write set has to be signed by the consignor and consignee involved in the shipment. The validating node queries the smart shipment contract state to identify which pseudo-PUs act as consignor and consignee in the shipment. If these identities match with the identities of the confirming nodes in the transaction proposal that signed the write set, the transaction is deemed valid. The write set is then used to update the world state. If a block contains an invalid transaction, the block is not dismissed.

Instead, an invalid transaction is tagged. The write set of an invalid transaction is not used to update the world state. Finally, after updating the world state the hash of the state is added to the block.

Table 28: Block of transactions.

Block N			
Hash block N-1	Root hash transactions	State hash	Transactions (Transaction 1, timestamp 1; Transaction 2, timestamp 2)

In conclusion, the consensus mechanism of the architecture design uses an execute – order – validate transaction process. For the ordering step, Practical Byzantine Fault Tolerance (PBFT) is used as consensus protocol to provide high scalability in terms of transaction throughput. The consortium parties act as ordering nodes within PBFT.

This sub section presented the consensus mechanism of the architecture design. The next sub section presents the application component.

5.1.4 Application

The application component of a blockchain-based architecture can consist of two sub-components: cryptocurrency tokens and smart contracts. The architecture for use case I should be open to businesses (see UC1-req. 5), ensure integrity (see UC1-req. 4) and does not require an economic incentive. The blockchain-based architecture only uses a smart contract to implement all business logic that provides a connection between the other components. This ensures that all transactions and activities comply with standards as specified by the governing consortium of the architecture. Currently, there are no (international) standards for blockchain technology or more specifically smart contracts. The assumption is therefore made that used standards are based on existing trade data exchange standards like UN/EDIFACT, World Customs Organization (WCO) data model and GS1 standards. Within the architecture the smart contract is known as the smart shipment contract. Any party is free to deploy its own smart shipment contract as long as it complies with the standards of the governing consortium. The smart shipment contract consists of three elements: state, functions and confirmation policies. Each shipment is represented by a separate smart shipment contract. Figure 23 provides an overview of the contract.

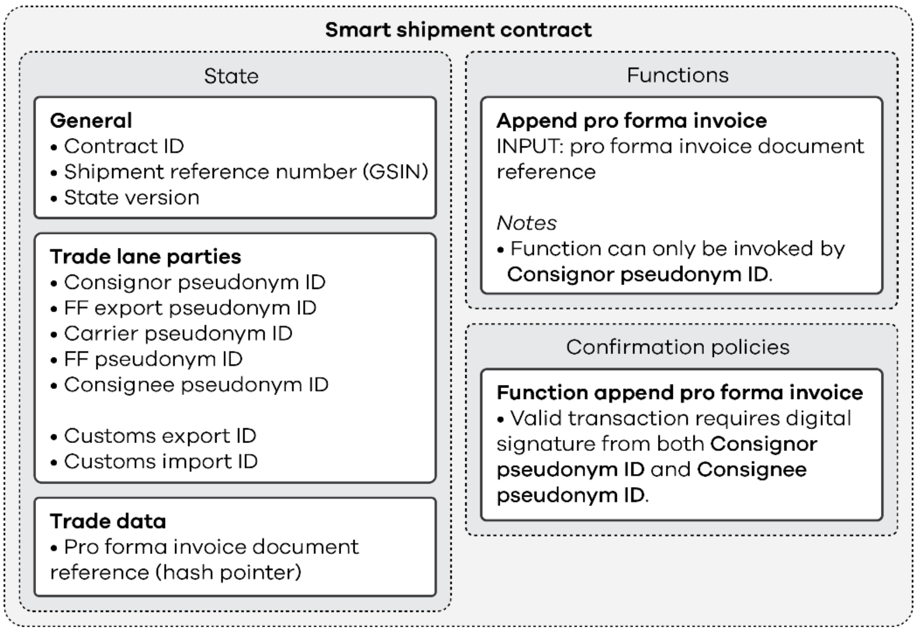

Smart shipment contract
State
General
• Contract ID
• Shipment reference number (GSIN)
• State version
Trade lane parties
• Consignor pseudonym ID
• FF export pseudonym ID
• Carrier pseudonym ID
• FF pseudonym ID
• Consignee pseudonym ID
• Customs export ID
• Customs import ID
Trade data
• Pro forma invoice document reference (hash pointer)
Functions
Append pro forma invoice
INPUT: pro forma invoice document reference
Notes
• Function can only be invoked by Consignor pseudonym ID.
Confirmation policies
Function append pro forma invoice
• Valid transaction requires digital signature from both Consignor pseudonym ID and Consignee pseudonym ID.

Figure 23: Overview smart shipment contract.

State
The state of the smart shipment contract is stored on the blockchain ledger and is divided into three parts: general, trade lane parties and trade data elements. The general state consists of a contract ID, shipment reference number and state version. Parties involved in a shipment should be able to identify the smart shipment contract in the blockchain ledger to be invoked by a transaction. Therefore, the general state includes a contract ID. To provide a reference to the shipment the pro forma invoice document belongs to, a shipment reference number is required. Several standards for shipment references exist that are based on the Unique Consignment Reference of the World Customs Organization. The most commonly used standard for shipment reference in industry is the Global Shipment Identification Number (GSIN) that is part of the GS1 standard (GS1, 2013). Therefore, the GSIN is included in the state of the smart shipment contract. As the architecture uses an account-based transaction model that requires state version control to avoid double spending, the state version is included. For read and write permission management and the confirmation policy, the trade lane parties state part consists of pseudo-PU or PU of all parties involved in the shipment. In this use case, this includes the consignor, Freight Forwarder (export and import), carrier, consignee and customs authority (export and import). To retrieve the pro forma invoice document, the state includes a document reference in the form of a hash pointer.

Functions
This use case includes only one type of transaction: 'append pro forma invoice'. The smart shipment contract implements this type of transaction that can be invoked by the consignor. The function includes a check to ensure that only the consignor of the shipment can invoke the transaction function. The check compares the pseudo-PU of the invoking node to the pseudo-PU of the node specified as consignor in the state of the smart shipment contract.

Confirmation policies
Customs authorities can only use a pro forma invoice document for manual cross-validation if correctness of the document is ensured. In a trade lane, two parties can determine if a pro forma invoice is correct: the consignor and consignee. If both parties agree that the pro forma invoice is correct, customs authorities can be certain that the document is correct. Therefore, to ensure correctness of the exchanged pro forma invoice, the confirmation policy of an 'append pro forma invoice' transaction requires signatures from both the consignor and consignee as defined in the state in order for a transaction to be deemed valid.

In conclusion, the application component of the architecture design uses a smart shipment contract to implement all business logic. The contract consists of a state, functions and confirmation policies.

5.1.5 Sub-conclusion blockchain-based architecture of use case I

This section presented the design of the blockchain-based architecture design that supports use case I: 'Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration'. The design is shown in Figure 29. The architecture uses a permissioned network configuration with identity management based on PKI managed by a consortium with support for external and internal PU identify certificates to be open to businesses and internal PU pseudonym certificates for confidentially purposes. Permission management on network access permission, blockchain ledger read and write permission and consensus participation permission ensure only authorized parties can perform actions within the architecture. Within the data structure and storage component, a multiple blockchain ledger structure (one ledger per trade lane) is used. An account-based transaction model stores transactions in a blockchain and the state of the contract, including a pro forma invoice document reference (hash pointer), in the world state. External and internal off-chain storage is used to store the actual pro forma invoice document.

Documents are encrypted and external ABAC managed by the data owner (consignor) enables decryption key distribution. The architecture uses a smart contract to implement the business logic. The smart shipment contract stores the pro forma invoice document reference and PU and pseudo-PU identity certificates of the trade lane parties in its state. An 'append pro forma invoice' function to create a transaction enables a consignor to exchange a pro forma invoice document. The trade lane party identities are used for permission management and decryption key distribution. For the consensus mechanism a contract execute - order – and validate transaction process is used. Transaction ordering uses PBFT as consensus protocol to enable high transaction throughput with consortium nodes acting as ordering nodes. The next section presents the design of the blockchain-based architecture for implementation of use case II.

5.2 Blockchain-based architecture of use case II

This section presents the blockchain-based architecture that supports use case II: Automated generation of an import declaration by aggregating already exchanged pro forma invoice data. Sub section 5.2.1 presents the network configuration component. The data structure and storage component is presented in sub section 5.2.2. Sub section 5.2.4 presents the application component and section 5.2.3 presents the consensus mechanism component. Sub section 5.2.5 concludes the section and provides an overview of the architecture.

Use case II is an extension of use case I. It assumes that the architecture to exchange pro forma invoice data is already in place. Use case II is built on top of this architecture. Therefore, some architecture components are similar to the components of the architecture to implement use case I and are therefore not discussed in detail.

5.2.1 Network configuration

In use case II a blockchain-based architecture is used to automatically generate an import declaration by aggregating already exchanged pro forma invoice data (see UC2-req. 1). This implies that architecture uses the same blockchain network that consist of a peer-to-peer network of nodes that represent businesses and customs authorities that have a copy of the blockchain ledger as use case I. Therefore, only the highlights of this component are presented. See section 5.1.1 for a more detailed description of the component. The architecture is governed by a consortium consisting of both businesses and public authorities. To manage identities and read and write access to the blockchain ledger, identity and permission management are needed which can only be realized in a permissioned network configuration topology. The inevitable limited scalability of eight to fifteen transactions per second and delay of up to sixty minutes Proof-of-Work is limited by, makes this consensus protocol infeasible for a blockchain-based architecture for the lodging of an import declaration. For example, to handle the lodging of import declarations within the Netherlands alone a scalability of at least ten transactions per second is needed (see UC2-req. 3). From section 4.4 it also follows that consensus protocols based on voting do provide sufficient scalability in terms of transaction throughput, however at the cost of scalability in terms of nodes participating in the consensus protocol. In a blockchain network consisting of thousands of trade lanes businesses and customs authorities represented by nodes, this means that only a selected group of nodes can participate. To manage which nodes can participate, permission management is therefore needed. This can only be realized in a permissioned network configuration topology.

In conclusion, to enable identity and permission management, and use of a non-computational heavy consensus protocol that provides sufficient scalability, the architecture uses a permissioned network configuration. For identity management the architecture design uses PU identity certificates for access to the blockchain network. These certificates are either issued by an external CA (national eID CAs) or by an

internal network CA. To provide confidentiality of businesses involved in a shipment and exchange of a pro forma invoice document, PU pseudonym identity certificates are used. These certificates are issued by an internal network PCA and derived from a PU identity certificate. For permission management the architecture design uses four types of permissions: network access, read, write and consensus participation. Any party with an PU identity certificate has network access permission.

5.2.2 Data structure and storage

In this sub section, the data structure and storage component of the architecture design is presented. The architecture should enable a declarant to automatically generate an import declaration by aggregating already business-to-business exchanged pro forma invoice data elements (see UC2-req. 1). If data elements are stored on the blockchain ledger, all nodes that have read permission and thus a copy of the ledger have access to these data elements. Even if encrypted, storing it on the blockchain ledger might reduce the willingness of the consignor to exchange. Whilst data elements are stored encrypted and thus no physical access is possible without using a decryption key, in theory a malicious party could still gain access given enough time. In addition, storing data elements on the blockchain ledger can be seen as a centralized storage which would conflict with the requirement for decentralization (see UC1-req. 5). However, to guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value data elements should be stored on the blockchain ledger. This guarantees that the transaction process is deterministic. Any party that aggregates data elements to generate an import declaration will have the same output. In addition, the risk of physical access if data elements are stored encrypted is low. Therefore, the architecture does not use an off-chain storage for the individual pro forma invoice data elements. All data elements are stored in the state of the smart shipment contract to guarantee correct retrieval. Table 29 shows the data storage of the architecture design.

Table 29: Data storage architecture design use case II.

Data storage				
On-chain blockchain ledger				
Consignor pseudo-PU	Consignee pseudo-PU	Incoterm	Invoice value	Transport cost after entry

Ledger structure

In the architecture design, a smart import declaration is deployed on the blockchain ledger that contains already exchanged pro forma invoice data elements stored in the state of the smart shipment contract. Therefore, the ledger structure is similar to that of the architecture design for implementation of use case I as discussed in 5.1.2.

Transaction model

Within the architecture design the assumption is made that pro forma invoice data elements related to a shipment and lodged import declarations are not loosely exchanged via transactions. Instead, pro forma invoice data elements are appended to a smart contract that represents a shipment. This contract is referred to as the smart shipment contract. Each shipment has its own smart shipment contract. To that extent, the contract has a state that keeps track of at least the data elements specified in Table 29. An import declaration is lodged using a smart import declaration. To that extent, the contract has a state that keeps track of at least the smart shipment contract ID for which an import declaration is lodged and the generated customs value. The blockchain ledger thus consist of two parts as shown in Table 30. The first part is a blockchain that contains an ordered list of transactions stored in blocks that represent smart import declaration state changes. The second part is the world state that reflects the latest state of the smart shipment contract and the smart import declaration.

Table 30: Blockchain ledger architecture use case II.

Blockchain ledger	
Blockchain	*World state*
Ordered list of transactions stored in blocks that represent smart import declaration state changes	Smart import declaration state including smart shipment contract ID and customs value.
(outside scope)	Smart shipment contract state including consignor pseudo-PU, consignee pseudo-PU, Incoterm, invoice value and transport cost after entry

Data security
The pro forma invoice data elements are commercially sensitive. As in use case II the focus is not on data security, it is assumed that data elements are encrypted. A party (declarant) that requires access to the data elements can request a decryption key from the data owner. Decryption key management is similar to section 5.1.2.

In conclusion, the data structure and storage of the architecture design stores the pro forma invoice data elements on the blockchain ledger. The architecture uses a multiple blockchain ledger structure with a ledger per trade lane. In addition, an account-based transaction model is used to store the state of the blockchain ledger which includes the states of both smart import declaration and smart shipment contract. Data encryption is used to provide confidentiality of the pro forma invoice data elements.

5.2.3 Consensus mechanism

This sub section presents the consensus mechanism component of the architecture design to support use case II. Because use case II is an extension of use case I and is built on top of the architecture that supports use case I. This implies that the architecture uses the same consensus mechanism as presented in section 5.1.3. In this sub section, some considerations are made to discuss if the consensus mechanism would be suitable for use case II. The retrieval of pro forma invoice data elements and calculation of the customs value implies that the transaction process must be deterministic. Any node that uses the same pro forma invoice data elements as input for the smart import declaration should have the same customs value as output. However, pro forma invoice data elements are encrypted and only authorized parties (e.g., declarant) have access to the data elements. As a result, not every node will render the same output as encrypted data elements cannot be used to calculate the customs value. Using the order - execute transaction process could therefore cause issues as nodes will not reach consensus due to different input. In addition, if every node has to invoke each smart contract for each transaction in order to update the world state, this could lead to performance issues as contracts are invoked in sequence. The execute – order – validate transaction process overcomes the issue of non-authorized parties not being able to calculate the customs value due to encrypted data elements and as a result not reach consensus. Only nodes involved in a ‘lodge import declaration’ transaction that have access to the data elements have to invoke the smart contract. Thus, the execute – order - validate transaction process is suitable for the architecture that supports use case II.

Execute
Because the execute step for use case II is slightly different from use case I, this sub section provides a summary of the step. The execute step can be summarized as follows:

1. The invoking node creates a preliminary transaction proposal including transaction ID, smart import declaration ID, contract function (lodge import declaration), function input (smart shipment contract ID) and invoking node pseudo-PU. The transaction proposal is signed by the invoking node;
2. The invoking node invokes the smart import declaration's 'lodge import declaration' function using the transaction proposal.
3. The smart import declaration uses the function input to get the state from the corresponding smart shipment contract. Based on the pro forma invoice data elements (Incoterm, invoice value and transport cost after entry), the contract calculates the customs value and encrypts the value. The output is a write set that containing declarant pseudo-PU, smart shipment contract ID and customs value.
4. The invoking node creates a transaction proposal including transaction ID, smart import declaration ID, contract function, function input, write set, confirming node 1 pseudo-PU, confirming node 1 signature and invoking node pseudo-PU. The transaction is signed by the invoking node;
5. The invoking node sends the transaction proposal including decryption key to the confirming node based on the smart import declaration confirmation policy to request confirmation;
6. The confirming node repeats step 2 and 3.
7. The confirming node checks if the write set corresponds to the write set of the transaction proposal.
8. The confirming node creates a confirmation message including transaction ID, write set, confirming node 2 pseudo-PU and confirming node 2 signature. The message is signed by confirming node 2;
9. The confirming node sends the message to the invoking node;
10. The invoking node creates a final transaction proposal including timestamp, ledger ID, smart import declaration ID, contract function, write set, confirming node 1 pseudo-PU, confirming node 1 signature, confirming node 2 pseudo-PU, confirming node 2 signature and invoking node pseudo-PU. The transaction is signed by the invoking node.

Transaction ordering
The transaction throughput of thousands of transactions per second PBFT offers provides sufficient scalability to handle at least ten 'lodge import declaration' transactions per second (see UC2-req. 3). Therefore, PBFT is used as consensus protocol. The consortium parties act as ordering nodes.

Validation
The validation step follows the same process as use case I, using the confirmation policy of the smart import declaration.

In conclusion, the consensus mechanism of the architecture design uses an execute – order – validate transaction process. For the ordering step, Practical Byzantine Fault Tolerance (PBFT) is used as consensus protocol to provide high scalability in terms of transaction throughput. The consortium parties act as ordering nodes within PBFT.

5.2.4 Application

The application component of the architecture design to support use case II uses two smart contracts to implement all business logic: 1) a smart shipment contract and 2) a smart import declaration. Both contracts are deployed on the blockchain ledger of the trade lane as shown in Figure 24. Within the architecture, the state of a smart shipment contract is stored on the blockchain ledger and contains all relevant pro forma invoice data elements as shown in Figure 25. For use case II, only the state of the smart shipment contract is discussed. Whilst there are no standards on blockchain technology and specifically smart contracts, the assumption is made that used standards for the exchange of individual pro forma invoice data elements are based on existing trade data exchange standards. These include UN/CEFACT, the WCO data model and GS1 standards. The state of the smart shipment contract consists of two parts: general and pro forma invoice data elements variables. The general state consists of a contract ID and shipment reference number (GSIN). To enable retrieval of pro forma invoice data elements for a shipment a reference should be provided to be able to identify the smart shipment contract in the blockchain ledger. The pro forma invoice data elements include the consignor and consignee PU, Incoterm, invoice value and transport cost after entry to be able to calculate the customs value.

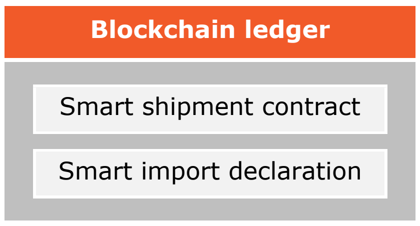


Figure 24: Blockchain ledger deployed contracts.

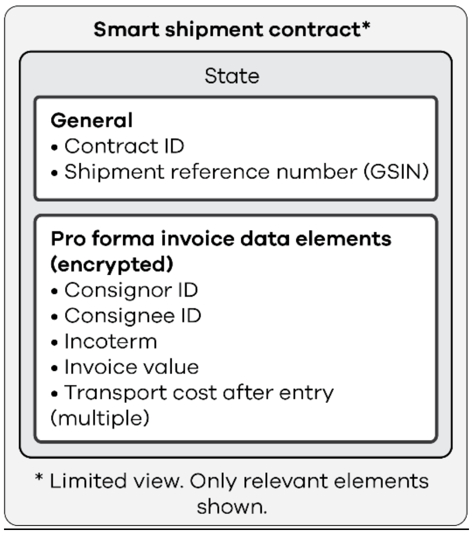

Smart shipment contract*
State
General
• Contract ID
• Shipment reference number (GSIN)
Pro forma invoice data elements (encrypted)
• Consignor ID
• Consignee ID
• Incoterm
• Invoice value
• Transport cost after entry (multiple)
* Limited view. Only relevant elements shown.

Figure 25: Smart shipment contract use case II.

The smart import declaration is used to aggregate pro forma invoice data elements from the smart shipment contract and calculate the custom value to lodge the import declaration. Any party is free to deploy its own smart import declaration on the blockchain ledger and use it to lodge import declarations. For example, a Freight Forwarder that acts as declarant for many import declarations might benefit from automation of the lodging of the import declaration. However, to ensure correct aggregation of pro forma invoice data elements and calculation of the customs value (see UC2-req. 2) the import declaration has to comply with standards specified by a customs authority. These standards are based on UN/CEFACT or the WCO data model. Compliance is realized by customs authorities performing audits on the code of a smart import declaration before its deployed on the blockchain ledger. This follows the concept of system-based control by customs authorities that aims at using audits on information systems rather than time-consuming physical inspection of shipments (Weigand & Bukhsh, 2011). To that extent, the smart import declaration code is signed by a customs authority as stamp of approval if the contract is compliant with standards before deployed on the blockchain ledger. Note that this can only be used to prove that the code complies with the standards laid out by the customs authority. A customs authority will accept import declarations lodged using approved smart import declarations. It does however not shift liability towards the customs authority in case of an incorrect import declaration that contains for example the wrong customs value. At all times, the declarant is responsible for lodging a correct import declaration. The smart shipment contract consists of three elements: state, functions and confirmation policies. Each import declaration is represented by a separate smart import declaration. Figure 26 provides an overview of the smart import declaration.

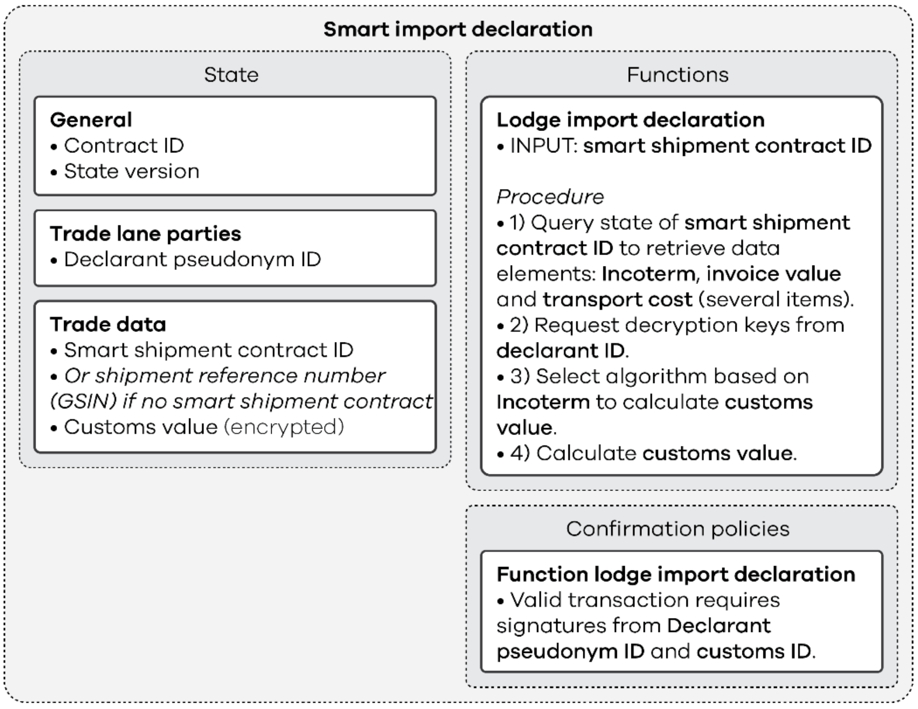

Smart import declaration
State
General
• Contract ID
• State version
Trade lane parties
• Declarant pseudonym ID
Trade data
• Smart shipment contract ID
• Or shipment reference number (GSIN) if no smart shipment contract
• Customs value (encrypted)
Functions
Lodge import declaration
• INPUT: smart shipment contract ID
Procedure
• 1) Query state of smart shipment contract ID to retrieve data elements: Incoterm, invoice value and transport cost (several items).
• 2) Request decryption keys from declarant ID.
• 3) Select algorithm based on Incoterm to calculate customs value.
• 4) Calculate customs value.
Confirmation policies
Function lodge import declaration
• Valid transaction requires signatures from Declarant pseudonym ID and customs ID.

Figure 26: Overview smart import declaration.

State
The state of the smart import declaration is stored on the blockchain ledger and is divided into three parts: general, trade lane parties and trade data elements. The general state consists of a contract ID and state version. The state includes a contract ID to provide a reference to a lodged import declaration. As the architecture uses an account-based transaction model that requires state version control to avoid double spending, the state version is included. The pseudo-PU of the declarant is included in the state for easy identification. The state also includes the smart shipment contract ID and customs value which confirms an import declaration is lodged. The contract ID enables a customs authority to retrieve pro forma invoice data elements for the specific shipment, for example for additional risk assessment. The customs value can be used by the customs authority for calculation of duties.

Functions
This use case includes only one type of transaction: 'lodge import declaration'. The smart import declaration implements this transaction type. The basic procedure of the smart import declaration consists of four steps. First, query the state of the smart shipment contract with ID provided as input to the function to retrieve pro forma invoice data elements Incoterm, invoice value and transport cost after entry. Second, request decryption keys from the declarant that has to request access to the data elements from the data element owner (consignor). Third, select an algorithm based on the Incoterm. For each of the ten Incoterms an algorithm is specified. For example, for the DDP Incoterm the algorithm is customs value = invoice value – transport cost after entry (see section 2.2 on customs valuation). Fourth, actual calculation of the customs value by running the algorithm.

Confirmation policies
Customs authorities can only use a lodged import declaration if correct retrieval of pro forma invoice data elements and calculation of the customs value is guaranteed. Two parties can determine if the import declaration is correct: the declarant and the customs authority. If only the declarant would be required to confirm if a transaction is valid, the customs authority cannot determine if the lodged

import declaration is correct unless it invokes the smart import declaration manually afterwards. Therefore, to guarantee correctness, the confirmation policy of an 'lodge import declaration' transaction requires signatures from both the declarant and the customs authority with role customs at import as defined in the state of the smart shipment contract in order for a transaction to be deemed valid.

5.2.5 Sub-conclusion blockchain-based architecture of use case II

This section presented the design of the blockchain-based architecture design for support of use case II: 'Automated generation of an import declaration by aggregating already exchanged pro forma invoice data'. The design is shown in Figure 30. The architecture is an extension of the architecture of use case I and thus uses the same permissioned network configuration. Within the data structure and storage component, a multiple blockchain ledger configuration (one for each trade lane) is used. An account-based transaction model stores transactions in a blockchain and the contract state including individual pro forma invoice data elements and customs value data element in the world state. No off-chain storage is used. Individual data elements are encrypted and external ABAC managed by the data owner (consignor) enables decryption key distribution. The same consensus mechanism as use case I is used. The architecture uses two smart contracts to implement the business logic. A smart shipment contract stores pro forma invoice data elements in its state. A declarant can use the 'lodge import declaration' function of a separate smart import declaration to retrieve these data elements and calculate the customs value based on the retrieved data elements.

The next section presents the design of the blockchain-based architecture for implementation of use case III.

5.3 Blockchain-based architecture of use case III

This section presents the blockchain-based architecture that supports use case III: Transfer title to the goods using an electronic negotiable Bill of Lading. Sub section 5.3.1 presents the network configuration component. The data structure and storage component is presented in sub section 5.3.2. Sub section 5.3.3 presents the consensus mechanism component. The application component is presented in section 5.3.4. Sub section 5.3.5 concludes the section and provides an overview of the architecture.

5.3.1 Network configuration

In use case III a blockchain-based architecture is used to enable business-to-business exchange to transfer ownership of an electronic negotiable Bill of Lading to a new holder (see UC3-req. 1). A typical life cycle of a negotiable Bill of Lading involves multiple businesses: the carrier that issues the negotiable Bill of Lading and one or more businesses that are either current holder (e.g., consignor) or become a new holder by transferring ownership. To enable business-to-business exchange, each business is represented as node in a peer-to-peer blockchain network in which a blockchain ledger is distributed. Each node has its own copy of the blockchain ledger.

A blockchain-based architecture to transfer title to the goods using an electronic negotiable Bill of Lading is a highly technical and organizationally complex. Limited adoption of previous initiatives to implement an electronic negotiable Bill of Lading underlines these complexities (Goldby, 2008). Governance of the architecture is therefore critical to successful implementation. This cannot be realized by a single party as it would conflict with the requirement of not having a central authority in control and ensuring openness to businesses willing to use the architecture to transfer title to the goods (see UC3-req. 4). Therefore, the architecture relies on a consortium of businesses and international organizations that together govern the architecture. The main business parties involved in an electronic negotiable Bill of Lading are carriers. Carriers issue a negotiable Bill of Lading and release the goods if a holder surrenders the document. Involvement of carriers is therefore needed. In addition, to reduce legal uncertainty regarding the functional equivalence of an electronic negotiable Bill of Lading (see UC3-req. 2), the International Group of Protection & Indemnity Clubs (P&I) plays an important role in the development and endorsement of electronic negotiable Bill of Lading initiatives. The group has many members like carriers and freight forwarders and provides mutual insurance to these members (P&I, n.d.). All current initiatives are endorsed by P&I. Without endorsement, businesses are unlikely to use the electronic negotiable Bill of Lading as there is uncertainty regarding insurance coverage in case of loss or theft of the negotiable Bill of Lading. Therefore, having P&I as part of the consortium will most likely convince businesses that the architecture can be used as functional equivalence of a paper-based negotiable Bill of Lading. In addition, due to the technical complexity of blockchain technology a technology company is needed for development. The consortium thus consists of carriers, P&I and a technology company. This ensures that no single authority is in control which satisfies requirement UC3-req. 4. An in-detail design of the governance structure is outside the scope of the research.

Topology
In this paragraph, the chosen topology is presented. In a permissionless network nodes are anonymous. This would overcome the issue of registration current electronic negotiable Bill of Lading initiatives face regarding adoption as no registration. However, public authorities should always be able to identify businesses involved in the transfer of a negotiable Bill of Lading in case of disputes over who is the actual holder and thus lawful owner (UNCITRAL, 2017). Also, besides carriers and other businesses that are current or new holder, there is no need for other parties not involved to be part of the blockchain network and thus have a copy of the blockchain ledger. Therefore, to manage identities and read and write access to the blockchain ledger, identity and permission management are needed which can only be realized in a permissioned network configuration topology.

In addition, the architecture should be able to handle high volumes of electronic negotiable Bill of Lading issuing and transferring of ownership in real-time (see UC3-req. 5). As discussed in section 4.4, in a permissionless network any party can become a node at their own discretion, thus rendering nodes anonymous. To overcome the double spend problem in anonymous peer-to-peer networks, computational heavy consensus protocols like Proof-of-Work are needed (Ølnes et al., 2017; Wüst & Gervais, 2017). The inevitable limited scalability of eight to fifteen transactions per second and delay of up to sixty minutes Proof-of-Work is limited by, makes this consensus protocol infeasible for a blockchain-based architecture to transfer title to the goods using an electronic negotiable Bill of Lading. An architecture to issue and transfer ownership of an electronic negotiable Bill of Lading has to handle hundreds of millions of transactions per year in real-time (see UC3-req. 5). From section 4.4 it also follows that consensus protocols based on voting do provide sufficient scalability in terms of transaction throughput, however at the cost of scalability in terms of nodes participating in the consensus protocol. In a blockchain network consisting of tens of carriers and thousands of businesses (e.g., consignors) that are holder of the electronic negotiable Bill of Lading that are represented by nodes, this means that only a selected group of nodes can participate. To manage which nodes can participate, permission management is therefore needed. This can only be realized in a permissioned network configuration topology.

In conclusion, to enable identity and permission management, and use of a non-computational heavy consensus protocol that provides sufficient scalability, the architecture uses a permissioned network configuration.

In this sub section, both identity and permission management components are presented.

Identity management (PKI)
To assign permission rights to parties, identification and authentication is needed to ensure the correct party gets permission. Therefore, the architecture design uses identity management based on Public Key Infrastructure (PKI). A short introduction to PKI was previously made in Box 4.

<u>PU identity certificates</u>
Requirement UC3-req. 4 states that the architecture should be open to businesses willing to participate in the business-to-business exchange to transfer title to the goods. No central authority should be in control of deciding who can or cannot be represented as node in the blockchain network and have a copy of the blockchain ledger or be able to make alterations to the ledger without consensus. The use of a single CA controlled by a central authority for the issuing of PU identity certificates for businesses involved would therefore not suffice as solution for identity management. Therefore, in the architecture design a distinction is made between external CAs and internal CAs that issue certificates.

<u>External CA</u>
For the identification and authentication of businesses in digital services, multiple commercial electronic identities (eID) – or PU identity certificates – issued by commercial trusted third parties exist (e.g., Verisign or DigiCert). External CAs reside outside the blockchain network and are not controlled by the consortium. By using an external CA, businesses do not have to go through the registration process to obtain a PU identity certificate which reduces bureaucracy. In addition, as the external CA is controlled by a thrusted third party, this ensures openness of the architecture to businesses to exchange trade documents. Therefore, the architecture design supports the use of externally issued PU identity certificates by external CAs. Businesses can use their own external PU identity certificate to gain access to the blockchain network. Which external PU identity certificates are supported is decided upon by the governing consortium.

<u>Internal network CA</u>
Next to an external CA, the architecture design also includes an internal network CA. A business that cannot rely on an externally issued PU identity certificate can be issued an PU identity certificate via the internal CA. The registration steps are left outside the scope of this research. Also, carriers require an PU identity certificate issued by the internal CA as it provides an additional role attribute 'carrier'. This role is needed for write permission to the blockchain ledger. Without role, any business would be able to issue negotiable Bill of Ladings while issuing should be under exclusive control of the carrier (see UC3-req. 2) to avoid fraud. Malicious parties could issue a fake negotiable Bill of Lading and sell it to a new holder. In addition, any party that requires access to the blockchain network with specific permission, for example consortium parties with administrative rights to maintain the architecture or permission to participate in the consensus mechanism, have to be issued an PU identity certificate by the internal CA to be identifiable. The internal network CA issues PU identity certificates with specific permissions and is therefore controlled by the consortium parties. The consortium parties together decide which (type of) party should be issued a certificate. As this control is a joined effort of consortium parties no single authority is in control of the issuance of PU identity certificates.

<u>PU pseudonym identity certificates</u>
The use of PU identity certificates enables real-world identification of businesses involved in transactions on the network. For example, the transfer of title to the goods via a transaction identifies the current and new holder. However, as discussed in requirement UC3-req. 3, the architecture should provide confidentiality of businesses involved as a lack of confidentiality can be a barrier for adoption. As a transaction is stored immutably in the blockchain ledger of which all businesses have a copy, the use of a PU identity certificate would not satisfy the requirement for confidentiality. On the contrary, the UNICTRAL Model Law mentions that a business cannot be

anonymous because in case of disputes or fraudulent activities public authorities should have the possibility to identify the real-world identity. However, it also mentions that identifiability does not imply that the real-world identity should be publicly known (UNCITRAL, 2017). Therefore, the architecture uses a second type of certificates: PU pseudonym identity certificates. These certificates are cryptographically derived from the PU identity certificate of the business that represents the real-world identity in such a way that only authorized parties can link the two certificates and thus identity the real-world identity behind the pseudonym identity. How this cryptographically derivation works is left outside the scope of this research. Business use the PU pseudonym identity certificates for signing transactions which allows the business to keep its real-world identity confidential (Benjumea et al., 2004). The PU pseudonym identity certificate are issued by an internal network Pseudonym Certificate Authority (PCA). Businesses can use their PU identity certificate to request a PU pseudonym identity certificate from the PCA. The PCA uses the PU identity certificate for identification and authentication and if authenticated, issues an PU pseudonym identity certificate that is cryptographically derived from the PU identity certificate. The cryptographic link between the PU identity and PU pseudonym identity certificate can only be solved by the PCA. Public authorities can request the PCA to provide the real-world identity if there is sufficient legal ground like fraud investigations. In the architecture, the use a separate PU pseudonym identity certificate for a transaction to transfer ownership to a new holder should provide sufficient confidentiality of the identity of the businesses involved.

In conclusion, for identity management the architecture design uses PU identity certificates for access to the blockchain network. These certificates are either issued by an external CA or by an internal network CA. To provide confidentiality of businesses involved in the transfer of title to the goods, PU pseudonym identity certificates are used. These certificates are issued by an internal network PCA and derived from a PU identity certificate.

Permission management
To control which parties have permission to perform actions on the blockchain network, permission management is used. Three main types of permission are distinguished in a permissioned blockchain network configuration: 1) read permission, 2) write permission and 3) consensus participation permission (see section 4.2). For more granular permission management, the architecture design adds an additional permission type besides read and write permission: network access permission. As discussed previously, identity management distinguishes between PU identity certificates to access the blockchain network and PU pseudonym identity certificates to participate in transactions. The use of read permission alone would not clearly show a distinction between access to the blockchain network and read or write permission to the blockchain ledger within the network. Having network access permission does not also imply read and write permission of the blockchain ledger. Thus, the architecture design includes four types of permission: 1) network access permission, 2) read permission, 3) write permission and 4) consensus participation permission.

Table 31 provides an overview of the identity and permission management of the architecture design and includes which certificates are used to manage which type of permission and which CA is responsible for issuing the certificate. For network access permission a PU identity certificate issued by an external CA or the internal network CA is used to manage permission. Read permission managed using PU pseudonym identity certificate issued by the internal network PCA or for businesses and PU identity certificates issued by the internal network CA for customs authorities. Write permission is managed using PU pseudonym identity certificates. Permission to participate in the consensus mechanism is managed using PU identity certificates issued by the internal network CA. For readability purposes, how the different types of permissions are managed (e.g., via Access Control Lists) is discussed in more detail in the respective components. Blockchain ledger read and write permission are discussed in section 5.1.2 on the data structure and storage component. Consensus participation permission is discussed in section 5.1.3 on the consensus mechanism component. For network access permission, control of which party has permission is simple. Any party with a valid PU identity certificate, whether issued by a supported external CA or internal network CA is granted access to the blockchain network.

Table 31: Overview identity and permission management architecture design use case III.

Party / Permission type	**Business node**	**Carrier node**	**Other party node**
Blockchain network access permission	PU identity certificate issued by any supported external CA or internal network CA	PU identity certificate issued by internal network CA	PU identity certificate issued by internal network CA
Blockchain ledger read permission	PU pseudonym identity certificate issued by internal network PCA	PU identity certificate issued by internal network CA	Does not apply
Blockchain ledger write permission	PU pseudonym identity certificate issued by internal network PCA	PU identity certificate issued by internal network CA	Does not apply
Blockchain network consensus permission	PU identity certificate issued by internal network CA	PU identity certificate issued by internal network CA	PU identity certificate issued by internal network CA

In conclusion, for permission management the architecture design uses four types of permissions: network access, read, write and consensus participation. Any party with an PU identity certificate has network access permission.

5.3.2 Data structure and storage

In this sub section, the data structure and storage component of the architecture design is presented. The architecture should enable the business-to-business exchange to transfer ownership of the electronic negotiable Bill of Lading to a new holder (see UC1-req. 1). In this architecture design, transactions are used to issue electronic negotiable Bill of Ladings and to transfer ownership of the electronic negotiable Bill of Ladings. The assumption is made that the electronic negotiable Bill of Lading is in a traditional form (e.g., .PDF document) and the architecture represents a decentralized title registry. Storing each negotiable Bill of Lading document on the blockchain ledger would require additional storage space which is unnecessary for a business only involved in a single negotiable Bill of Lading. Therefore, the architecture uses an additional off-chain document storage to store the negotiable Bill of Lading. Only a reference to the negotiable Bill of Lading is stored on the blockchain ledger. To ensure integrity of the document over its lifecycle (see UC3-req. 2), a hash pointer that contains the hash of the document is used as reference. The hash pointer also allows a party to identify the location of the document in the off-chain document storage. As ownership over the negotiable Bill of Lading shifts after each transfer to a new holder, the storage cannot be managed by a single party. For example, if a carrier manages the storage and fails, businesses can no longer transfer the negotiable Bill of Lading to a new holder as the new holder cannot check correctness. Therefore, the architecture uses a distributed document storage. The InterPlanetary File System (IPFS) is an often-mentioned initiative in light of blockchain technology (https://ipfs.io/). IPFS works similar to a blockchain-based system in that documents are stored in a permissionless distributed peer-to-peer network. Once stored, it becomes immutable and remains available as long as nodes keep the storage alive. A node in the IPFS network can be requested to provide a document. In the architecture, the consortium parties take part in the IPFS to guarantee availability of the negotiable Bill of Lading. Table 32 shows the data storage of the architecture.

Table 32: Data storage architecture design use case III.

Data storage	
Off-chain document storage (IPFS)	*On-chain blockchain ledger*
Negotiable Bill of Lading	Document reference (hash pointer)

Ledger structure

Whilst the architecture should be open to businesses, there is no clear reason why a business should have access to all negotiable Bills of Lading issued by all carriers. If the architecture would rely on a single blockchain ledger this could reduce the willingness of businesses to use the architecture. The underlying reason is that from a technical perspective a node having to maintain a ledger containing all documents can have a negative effect on scalability of the blockchain-based architecture as more storage space is required. A business might not be willing to participate in such architecture as this would result in unnecessary higher costs. It is not relevant for a random business to keep a blockchain ledger that contains millions of negotiable Bills of Lading and ownership transfers it is not involved in. Therefore, this architecture design uses a multiple blockchain ledger structure, in which each carrier has its own dedicated ledger as shown in Figure 27. Each blockchain ledger thus contains negotiable Bill of Ladings issued by a single carrier. The technicalities of how a node with receives a copy of the blockchain ledger for a specific carrier is left outside the scope of this research.

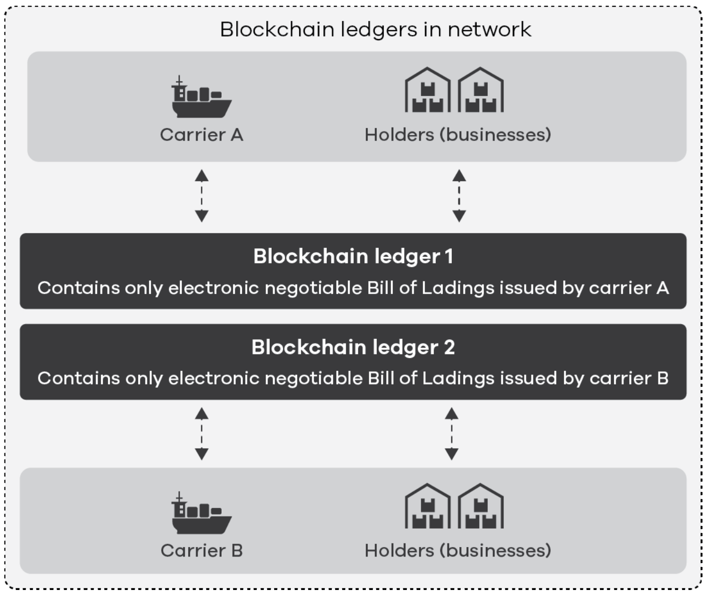

Blockchain ledgers in network
Carrier A
Holders (businesses)
Blockchain ledger 1
Contains only electronic negotiable Bill of Ladings issued by carrier A
Blockchain ledger 2
Contains only electronic negotiable Bill of Ladings issued by carrier B
Carrier B
Holders (businesses)

Figure 27: Multiple blockchain ledger structure with ledger per carrier use case III.

Transaction model

Within the architecture design the assumption is made that a smart contract is used to transfer ownership of the electronic negotiable Bill of Lading to a new holder. The contract acts as a decentralized title registry (see UC3-req. 4) and is referred to as smart negotiable Bill of Lading. Electronic negotiable Bills of Lading are issued or transferred to a new holder by invoking the smart negotiable Bill of Lading using transactions. Each negotiable Bill of Lading has its own smart negotiable Bill of Lading. It allows for implementation of all related business logic needed for issuing and transferring to a new holder. To that extent, the contract has a state that keeps track of at least the negotiable Bill of Lading document reference, a negotiable Bill of Lading reference number for identification, the PU of the carrier and the pseudo-PU of the current holder to manage write permission as only the carrier can issue a new smart negotiable Bill of Lading and only the current holder can transfer ownership to a new holder (see UC3-req. 2).

To track updates to the state of the smart negotiable Bill of Lading, either an Unspent Transaction Output (UTXO) or account-based transaction model can be used. UTXO is mostly used in context of cryptocurrencies. However, for more complex smart contracts, account-based is more relevant. It allows for faster and easier updating of the state of the smart contract as the blockchain ledger does not have to be queried each time a new transaction is created to calculate the state. Therefore, the architecture uses an account-based transaction model. The blockchain ledger thus consist of two parts as shown in Table 33. The first part is a blockchain that contains an ordered list of transactions stored in blocks that represent smart negotiable Bill of Lading state changes to either issue a new electronic negotiable Bill of Lading or to transfer ownership to a new holder. The second part is the world state that reflects the latest state of the smart negotiable Bill of Lading which includes the negotiable Bill of Lading document reference, negotiable Bill of Lading reference number and PU of the carrier and pseudo-PU of the current holder.

Table 33: Blockchain ledger architecture design use case III.

Blockchain ledger	
Blockchain	*World state*
Ordered list of transactions (issue or transfer) stored in blocks that represent smart negotiable Bill of Lading state changes.	Smart negotiable Bill of Lading states including negotiable Bill of Lading document reference, negotiable Bill of Lading reference number and identities of carrier and current holder.

Data security
As the negotiable Bill of Lading document contains commercially sensitive data, access should be restricted to ensure confidentiality (see UC3-req. 3). The architecture uses traditional encryption of the pro forma invoice document to ensure confidentiality. Before creating the negotiable Bill of Lading's document reference, the carrier generates a symmetric encryption/decryption key and encrypts the document via E(document, encryption/decryption key) before storing it in IPFS and creates a hash pointer as reference. This enables flexible distribution of the decryption key as it is not linked to a specific party as would be the case if the document was encrypted via E(document, PUparty) which would only provide that party access. To manage distribution of the decryption key, decryption key management is needed.

<u>Decryption key management</u>
The only businesses that should have access to the negotiable Bill of Lading document are the carrier, the current holder and the new holder. This allows for simple decryption key management. The decryption key is encrypted with the pseudo-PU and sent together with an 'issue negotiable Bill of Lading' or 'transfer to new holder' transaction. To that extent, the decryption key is encrypted with the pseudo-PU of the new holder via E(decryption key, pseudo-PUnewholder). This ensures that only authorized businesses have access to the document. As a result, no additional decryption key management is needed.

In conclusion, the data structure and storage of the architecture design stores negotiable Bill of Lading document in an off-chain document storage (IPFS) and stores only a document reference on the blockchain ledger. It uses a multiple blockchain ledger structure with a ledger per tradelane. In addition, an account-based transaction model

is used to store the state of the blockchain ledger. Data encryption is used to provide confidentiality of the negotiable Bill of Lading.

The next sub section presents the consensus mechanism component of the architecture design.

5.3.3 Consensus mechanism

This sub section presents the consensus mechanism component of the architecture design to implement use case III. The component ensures that the nodes in the peer-to-peer network reach consensus on the order of transactions and thus state of the blockchain ledger. First, this sub section discusses which transaction process (see section 4.4) is used. In the execute – order – validate transaction process, the consensus protocol is limited to the ordering of transactions and proposal of new blocks. This allows the use of smart negotiable Bills of Lading that use different programming languages and improves performance as smart contracts do not have to be invoked in sequence. Thus, as the execute – order – validate transaction process provides more flexibility, the architecture uses this process in the consensus mechanism. In the remainder of this sub section, each step is discussed. The first step that is discussed is the execute step.

Execute

The first step of the transaction process is invoking the smart contract and confirmation of the transaction by confirming nodes. An invoking node (current holder) can transfer the electronic negotiable Bill of Lading to a new holder via a transaction. To ensure that the transfer is correct, the read/write set of the transaction has to be confirmed by a confirming node. Which node has to confirm the read/write set is specified for the function in the confirmation policy of the smart negotiable Bill of Lading. In case of transferring ownership to a new holder, the new holder can confirm correctness of the document. The invoking node is always a confirming node as well. To create a transaction, the invoking node creates a preliminary transaction proposal including transaction ID, smart negotiable Bill of Lading ID, contract function, function input (pseudo-PUnewholder) and invoking node pseudo-PU as shown in Table 34. The transaction proposal is signed by the invoking node with pseudo-PRinvokingnode via E(transaction proposal hash, pseudo-PRinvokingnode). The invoking node invokes the negotiable Bill of Lading using the preliminary transaction proposal. The output is a read set that specifies the current holder (pseudo-PU) and state version and a write set that specifies the new holder (pseudo-PU). Next, the invoking node creates a transaction proposal including transaction ID, smart negotiable Bill of Lading ID, contract function, function input, encrypted negotiable Bill of Lading decryption key, read set, write set, confirming node 1 pseudo-PU, confirming node 1 signature and invoking node pseudo-PU as shown in Table 35. The transaction is signed by the invoking node. Next, the invoking node sends the transaction proposal to the confirming node (new holder pseudo-PU) to ensure correctness of the read and write sets. The confirming node retrieves the negotiable Bill of Lading document via the document reference from IPFS, decrypts the document via D(document, decryption key) and checks the document for correctness. Next, the confirming node invokes the smart negotiable Bill of Lading's 'transfer to new holder' function using the transaction proposal. The output is a read set that specifies the current holder (pseudo-PU) and state version and write set specifying new holder pseudo-PU. The confirming node checks if the read set corresponds to the read set of the transaction proposal. Next, the node creates a confirmation message including transaction ID, read set, write set, confirming node 2 pseudo-PU and confirming node 2 signature. The message is signed by the confirming node. The confirming node sends the message to the invoking node. The invoking node creates a final transaction proposal including timestamp, ledger ID, smart negotiable Bill of Lading ID, contract function, encrypted negotiable Bill of Lading decryption key, read set, write set, confirming node 1 pseudo-PU, confirming node 1 signature, confirming node 2 pseudo-PU, confirming node 2 signature and invoking node pseudo-PU. The transaction is signed by the invoking node. The timestamp is required for the correct ordering of transactions. The ledger ID is used to separate transaction of different blockchain ledgers during ordering. The transaction is signed by the invoking node.

Table 34: Transfer to new holder preliminary transaction proposal.

Transfer to new holder preliminary transaction proposal			
Transaction ID	Smart negotiable Bill of Lading ID	Contract function	Function input
Invoking party pseudo-PU	Invoking party message signature		

Table 35: Transfer to new holder transaction proposal.

Transfer to new holder transaction proposal			
Transaction ID	Smart negotiable Bill of Lading ID	Contract function	Function input
Encrypted negotiable Bill of Lading decryption key	Read set	Write set	Confirmation party 1 pseudo-PU
Confirmation party 1 signature	Invoking party pseudo-PU	Invoking party message signature	

Table 36: Transfer to new holder proposal confirmation message.

Transfer to new holder transaction proposal confirmation message			
Transaction ID	Read set	Write set	Confirmation party 2 pseudo-PU
Confirmation party 2 signature	Confirmation party 2 message signature		

Table 37: Transfer to new holder transaction proposal.

Append pro forma invoice transaction			
Timestamp	Ledger ID	Transaction ID	Smart negotiable Bill of Lading ID
Contract function	Negotiable Bill of Lading decryption key	Read set	Write set
Confirmation party 1 pseudo-PU	Confirmation party 1 signature	Confirmation party 2 pseudo-PU	Confirmation party 2 signature
Invoking party pseudo-PU	Invoking party signature		

The execute step can be summarized as follows:

1. The invoking node retrieves the pseudo-PU of the new holder;
2. The invoking node creates a preliminary transaction proposal including transaction ID, smart negotiable Bill of Lading ID, contract function, function input (new holder pseudo-PU) and invoking node pseudo-PU. The transaction proposal is signed by the invoking node;
3. The invoking node invokes the smart negotiable Bill of Lading's 'transfer to new holder' function using the transaction proposal. The output is a read set containing pseudo-PU of the current holder and state version and a write set containing new holder pseudo-PU.

4. The invoking node creates a transaction proposal including transaction ID, smart negotiable Bill of Lading ID, contract function, function input, encrypted negotiable Bill of Lading decryption key, read set, write set, confirming node 1 pseudo-PU, confirming node 1 signature and invoking node pseudo-PU. The transaction is signed by the invoking node;
5. The invoking node sends the transaction proposal to the confirming node (new holder pseudo-PU);
6. The confirming node retrieves the negotiable Bill of Lading document via the document reference from IPFS, decrypts the document via D(document, decryption key) and checks the document for correctness;
7. The confirming node invokes the smart negotiable Bill of Lading's 'transfer to new holder' function using the transaction proposal. The output is a read set containing pseudo-PU of the current holder and state version and write set containing new holder pseudo-PU;
8. The confirming node checks if the read set corresponds to the read set of the transaction proposal;
9. The confirming node creates a confirmation message including transaction ID, read set, write set, confirming node 2 pseudo-PU and confirming node 2 signature. The message is signed by confirming node 2;
10. The confirming node sends the message to the invoking node;
11. The invoking node creates a final transaction proposal including timestamp, ledger ID, smart negotiable Bill of Lading ID, contract function, encrypted negotiable Bill of Lading decryption key, read set, write set, confirming node 1 pseudo-PU, confirming node 1 signature, confirming node 2 pseudo-PU, confirming node 2 signature and invoking node pseudo-PU. The transaction is signed by the invoking node.

This completes the execute step. The next step is the ordering process.

Transaction ordering
As previously discussed in sub section 5.1.1, the architecture must be scalable to handle hundreds of millions of transactions per year in real-time. Proof-of-Work that can only handle between eight to fifteen transactions per second depending on implementation. This makes the protocol infeasible for use in the architecture design. Therefore, a different consensus protocol has to be selected for the architecture design. In the network configuration a permissioned network configuration was already selected to accommodate a different consensus protocol that is not computational heavy. While in a permissioned blockchain network there is no need for a probabilistic consensus protocol, there is still a need to achieve BFT. Honest nodes that participate in the consensus protocol can still be subject to Byzantine failure. For example, a node can go down due to technical issues (Gramoli, 2017). A consensus protocol that provides sufficient scalability in terms of transaction throughput and ensures BFT is thus needed.

The architecture design therefore uses PBFT as consensus protocol. Section 4.4.2 introduced the basic working of the protocol. PBFT has been tested to have a transaction throughput of tens of thousands of transactions per second. A requirement for the use case is scalability of hundreds of millions of transactions per year in real-time (see UC1-req. 6). PBFT thus enables sufficient scalability in terms of transaction throughput with low latency. An issue that has to be addressed in context of PBFT is limited scalability in terms of nodes that can participate in the protocol. Dinh et al. (2018) find that PBFT is scalable up to 32 nodes without major performance issues. For the architecture design, the number of nodes participating in the consensus protocol should therefore be less than 32. Within BFT, the number of nodes subject to Byzantine failure has to be less than one-third of the number of nodes to reach consensus, which in case of 32 nodes is ten nodes. The identities of nodes participating in the network are known, so the assumption can be made that Byzantine faulty nodes are not caused by malicious behavior and only technical of nature. The risk of not reaching consensus due to too many failures is therefore deemed minimal.

To participate, nodes need to have permission to participate. As in PBFT the identities of the nodes participating should be known to all other nodes, the architecture uses a public access control list that provides an overview of the PUs of nodes that participate. To that extent, a decision has to be made who governs the access control list and thus decides which nodes are given permission. The use of PBFT with a limited number of nodes that have permission to participate could in potential conflict with the requirement of having no central authority in control (see UC3-req. 5). The limitation could create the impression that a central authority is in control of the consensus protocol and can influence the ordering of transactions and thus world state of the blockchain ledger. It should however be noted that the only task of nodes participating in the PBFT protocol is the ordering of incoming transactions, propose blocks of ordered transactions and send block proposals to all other nodes in the network for that specific blockchain ledger. No further actions are performed on the transactions. This should provide businesses with sufficient trust that no central authority is in control of the world state of the blockchain ledger. To mitigate the potential conflict in context of who governs the access control list, the list has to be governed by multiple authorities. Within the architecture design the governing consortium would be the ideal choice as it consists of a representation of carriers, P&I and a technology company. No single authority can thus decide which nodes have permission to participate.

To overcome the potential issue of Byzantine failure, the list of nodes participating should be sufficiently diverse. As participating in the consensus protocol requires additional effort regarding technical knowledge and costs, it is expected that the list mainly consists of larger consortium parties like large carriers, P&I and the technology company that have the technological capabilities. The total sum of parties should be around thirty for the best performance of PBFT regarding transaction throughput and BFT. A detailed overview of the PBFT ordering process can be found in section 4.4.2. Transactions are separated based on blockchain ledger.

Validation
The final step of the transaction process is validation of the transaction. Each node that holds a copy of the blockchain ledger acts as validating node. First, the validating node appends the block proposal (N) to the blockchain. To that extent, it adds the hash of the previous block (N-1) and the root hash the transactions to block N and appends the block to the blockchain as shown in Table 28. Next, the validating node checks if the transaction is valid by checking if the confirmation policy for the smart negotiable Bill of Lading's 'transfer to new holder' function has been met. To do so, the confirmation policy is retrieved via the contract ID first. The confirmation policy states that the read and write sets have to be signed by both the current holder and new holder involved in the transfer transaction. The validating node queries the smart negotiable Bill of Lading state to identify which pseudo-PUs is the current holder. If the identity matches with one of the identities that signed the read and write sets and another party (new holder) has signed the read write set, the confirmation policy is fulfilled. Next, validity of the transaction is checked by comparing the read set state version with the current state version of the smart negotiable Bill of Lading. If these match, the transaction is not double spent and thus valid. Final, if valid the write set is used to update the world state. If a block contains an invalid transaction, the block is not dismissed. Instead, an invalid transaction is tagged. The write set of an invalid transaction is not used to update the world state. Finally, after updating the world state the hash of the state is added to the block.

Table 38: Block of transactions.

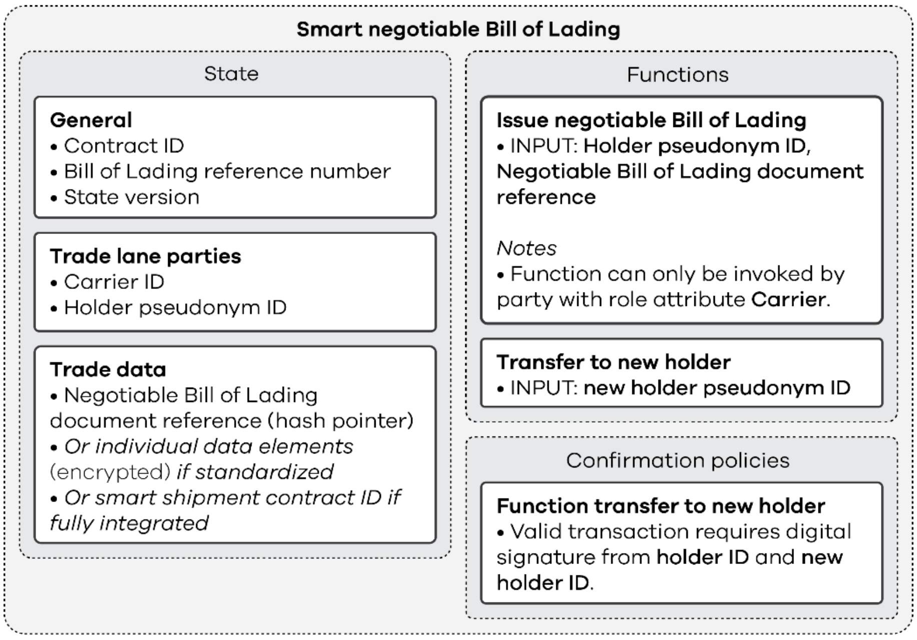
Block N			
Hash block N-1	Root hash transactions	State hash	Transactions (Transaction 1, timestamp 1; Transaction 2, timestamp 2)

In conclusion, the consensus mechanism of the architecture design uses an execute – order – validate transaction process. For the ordering step, Practical Byzantine Fault Tolerance (PBFT) is used as consensus protocol to provide high scalability in terms of transaction throughput. The consortium parties act as ordering nodes within PBFT.

This sub section presented the consensus mechanism of the architecture design. The next sub section presents the application component.

5.3.4 Application

The application component of a blockchain-based architecture can consist of two sub-components: cryptocurrency tokens and smart contracts. The architecture for use case I should be open to businesses (see UC3-req. 4) and does not require an economic incentive. The blockchain-based architecture therefore only uses a smart contract to implement all business logic that provides a connection between the other components. This ensures that all transactions and activities comply with standards as specified by the governing consortium of the architecture. Within the architecture the smart contract is known as the smart negotiable Bill of Lading. Any carrier is free to deploy its own smart negotiable Bill of Lading as long as it complies with the standards specified by the governing consortium. The smart negotiable Bill of Lading consists of three elements: state, functions and confirmation policies. Each shipment is represented by a separate smart negotiable Bill of Lading. Figure 28 provides an overview of the contract.



Smart negotiable Bill of Lading
State
General
• Contract ID
• Bill of Lading reference number
• State version
Trade lane parties
• Carrier ID
• Holder pseudonym ID
Trade data
• Negotiable Bill of Lading document reference (hash pointer)
• Or individual data elements (encrypted) if standardized
• Or smart shipment contract ID if fully integrated
Functions
Issue negotiable Bill of Lading
• INPUT: Holder pseudonym ID, Negotiable Bill of Lading document reference
Notes
• Function can only be invoked by party with role attribute Carrier.
Transfer to new holder
• INPUT: new holder pseudonym ID
Confirmation policies
Function transfer to new holder
• Valid transaction requires digital signature from holder ID and new holder ID.

Figure 28: Overview smart negotiable Bill of Lading.

State

The state of the smart negotiable Bill of Lading is stored on the blockchain ledger and is divided into three parts: general, trade lane parties and trade data. The general state consists of a contract ID, Bill of Lading reference number and state version. Businesses involved in the transfer of title to the goods should be able to identify the smart negotiable Bill of Lading in the blockchain ledger to be invoked by a transaction. Therefore, the general state includes a contract ID. To provide a reference to the actual negotiable Bill of Lading the smart negotiable Bill of Lading belongs to, the state includes a Bill of Lading reference number. As the architecture uses an account-based transaction model that requires state version control to avoid double spending, the state version is included. The state also includes the carrier PU and current holder pseudo-PU. These identities are used in the confirmation policy to ensure exclusive control over issue and transfer and allow the current holder to prove it is in possession of the negotiable Bill of Lading (see UC3-req. 3). To retrieve the negotiable Bill of Lading the smart negotiable Bill of Lading is linked to, the state also includes a document reference in the form of a hash pointer.

Functions
This use case includes two types of transactions: ‘issue negotiable Bill of Lading’ and ‘transfer to new holder’. The smart negotiable Bill of Lading implements these two types of transactions via functions that can be invoked by a carrier to issue a negotiable Bill of Lading or current holder to transfer title to the goods to a new holder. The ‘issue negotiable Bill of Lading’ function includes a check to ensure that only a carrier has write permission and can thus invoke the function. To that extent, it checks if the carrier PU that is used contains the correct role attribute ‘carrier’ as issued by the internal CA. In addition, the contract checks if a negotiable Bill of Lading with the same Bill of Lading reference number already exist in the world state of the blockchain ledger to avoid the same negotiable Bill of Lading being issued twice.

Confirmation policies
In a smart negotiable Bill of Lading, two different confirmation policies are used for the transactions. In case of an ‘issue negotiable Bill of Lading’ the carrier and consignor (first holder) determine if the negotiable Bill of Lading is correct. Therefore, the ‘issue negotiable Bill of Lading’ function requires the write set of the transaction to be signed by both the carrier and consignor. In case of an ‘transfer to new holder’ transaction, two parties can determine if the transfer is valid: the current holder and new holder. The current holder has exclusive control over the smart negotiable Bill of Lading, so any transfer should be authorized by the current holder. The new holder will only accept a transfer as valid if the negotiable Bill of Lading is correct. Therefore, the ‘transfer to new holder’ transaction requires that the read and write sets of the transaction are signed by the current holder and a new holder involved in the transfer. To check if the current holder has signed the transaction, the pseudo-PU of the holder in the state of the smart negotiable Bill of Lading is compared to the pseudo-PU of the confirming party that signed the transaction. If the pseudo-PUs match, the confirmation policy is fulfilled and the transaction is deemed valid.

In conclusion, the application component of the architecture design uses a smart negotiable Bill of Lading to implement all business logic. The contract consists of a state, functions and confirmation policies.

5.3.5 Sub-conclusion blockchain-based architecture of use case III

This section presented the design of the blockchain-based architecture that supports use case III: ‘Transfer title to the goods using an electronic negotiable Bill of Lading ’. The design is shown in Figure 31. The architecture uses a permissioned network configuration topology with identity management based on PKI with support for external and internal PU identify certificates to be open to businesses and internal PU pseudonym identity certificates for confidentially purposes. Permission management on network access permission, blockchain ledger read and write permission and consensus participation permission ensure only correct parties can perform actions within the architecture. Within the data structure and storage component, a multiple blockchain ledger configuration (one for each carrier) is used. An account-based transaction model stores transactions in a blockchain and the state of the contract, including a negotiable Bill of Lading reference (hash pointer) and PU pseudonym identity certificate of the current holder, in the world state. Internal off-chain storage is used to store the actual negotiable Bill of Lading document. Documents are encrypted, and decryption keys are distributed by the current holder. The architecture uses a smart contract to implement the business logic. The smart negotiable Bill of Lading stores the document reference and PU pseudonym identity certificate of the current holder in its state. An ‘transfer to new holder’ function enables the current holder to transfer ownership to a new holder. For the consensus mechanism an execute - order – and validate transaction process is used. Transaction ordering uses PBFT to enable high transaction throughput with the consortium parties acting as ordering nodes. The next section concludes this chapter.

5.4 Conclusion chapter 5

This chapter presented the design of the blockchain-based architecture of each use case to answer sub question 5: *'What does the blockchain-based architecture design look like for each of the critical trade document use cases?'*. The architecture that supports use case I is shown in Figure 29. Figure 30 shows the architecture that supports use case II. The architecture that supports use case III is shown in Figure 31. In the next chapter, for each use case the blockchain-based architecture design is demonstrated.

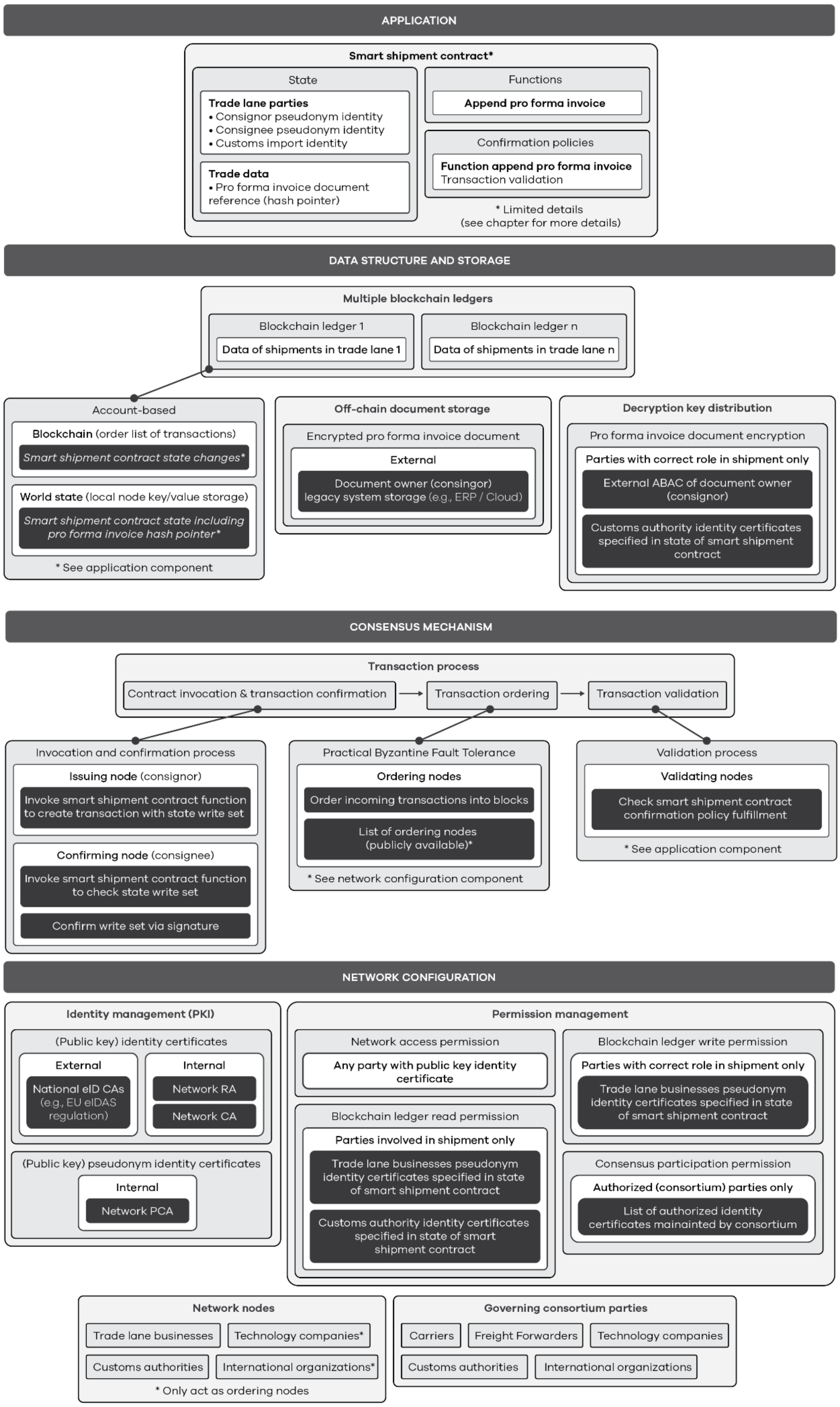
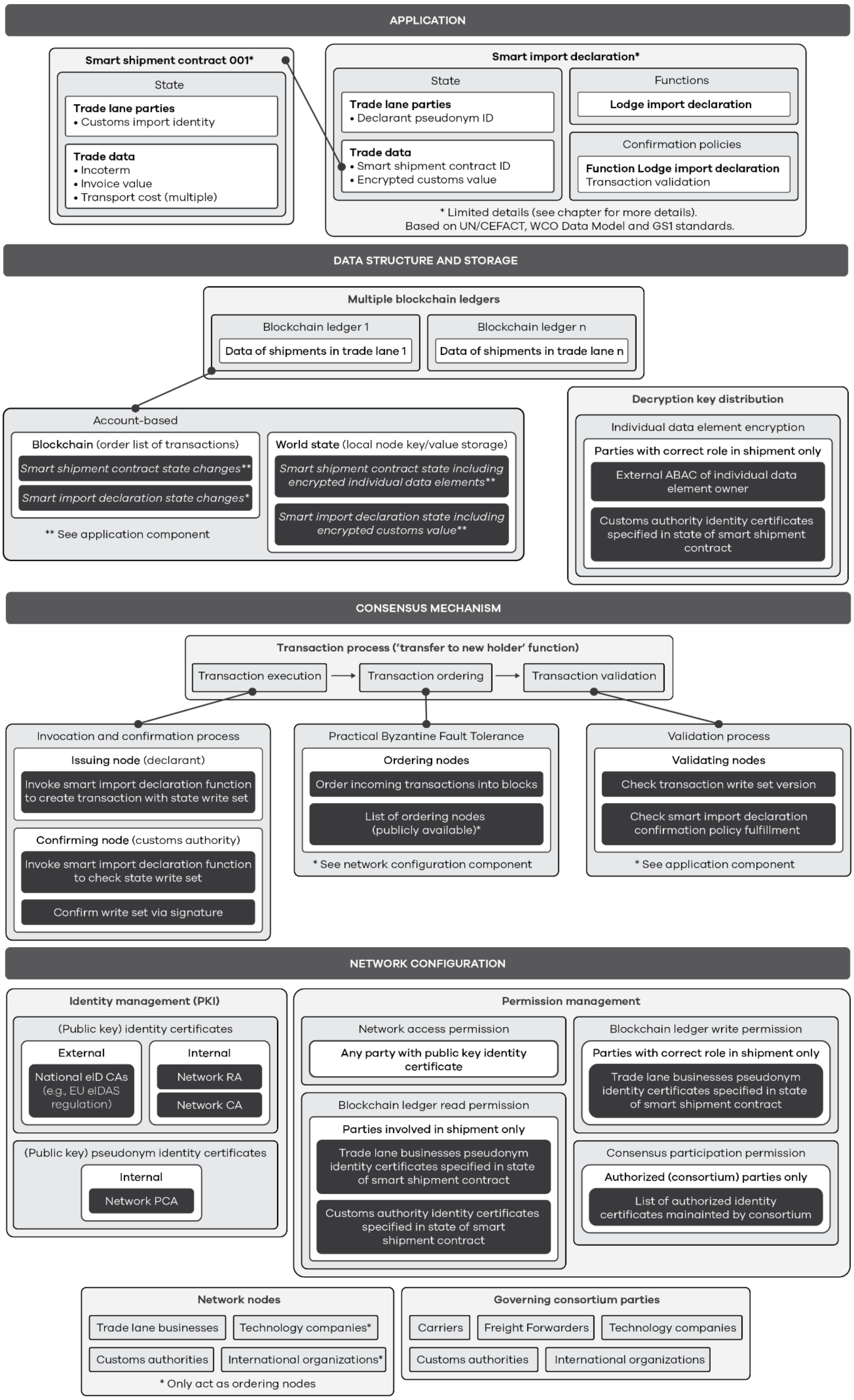

APPLICATION
Smart shipment contract*
State
Trade lane parties
• Consignor pseudonym identity
• Consignee pseudonym identity
• Customs import identity
Trade data
• Pro forma invoice document reference (hash pointer)
Functions
Append pro forma invoice
Confirmation policies
Function append pro forma invoice
Transaction validation
* Limited details (see chapter for more details)
DATA STRUCTURE AND STORAGE
Multiple blockchain ledgers
Blockchain ledger 1
Data of shipments in trade lane 1
Blockchain ledger n
Data of shipments in trade lane n
Account-based
Blockchain (order list of transactions)
Smart shipment contract state changes*
World state (local node key/value storage)
Smart shipment contract state including pro forma invoice hash pointer*
* See application component
Off-chain document storage
Encrypted pro forma invoice document
External
Document owner (consignor) legacy system storage (e.g., ERP / Cloud)
Decryption key distribution
Pro forma invoice document encryption
Parties with correct role in shipment only
External ABAC of document owner (consignor)
Customs authority identity certificates specified in state of smart shipment contract
CONSENSUS MECHANISM
Transaction process
Contract invocation & transaction confirmation
Transaction ordering
Transaction validation
Invocation and confirmation process
Issuing node (consignor)
Invoke smart shipment contract function to create transaction with state write set
Confirming node (consignee)
Invoke smart shipment contract function to check state write set
Confirm write set via signature
Practical Byzantine Fault Tolerance
Ordering nodes
Order incoming transactions into blocks
List of ordering nodes (publicly available)*
* See network configuration component
Validation process
Validating nodes
Check smart shipment contract confirmation policy fulfillment
* See application component
NETWORK CONFIGURATION
Identity management (PKI)
(Public key) identity certificates
External
National eID CAs (e.g., EU eIDAS regulation)
Internal
Network RA
Network CA
(Public key) pseudonym identity certificates
Internal
Network PCA
Permission management
Network access permission
Any party with public key identity certificate
Blockchain ledger read permission
Parties involved in shipment only
Trade lane businesses pseudonym identity certificates specified in state of smart shipment contract
Customs authority identity certificates specified in state of smart shipment contract
Blockchain ledger write permission
Parties with correct role in shipment only
Trade lane businesses pseudonym identity certificates specified in state of smart shipment contract
Consensus participation permission
Authorized (consortium) parties only
List of authorized identity certificates mainainted by consortium
Network nodes
Trade lane businesses
Technology companies*
Customs authorities
International organizations*
* Only act as ordering nodes
Governing consortium parties
Carriers
Freight Forwarders
Technology companies
Customs authorities
International organizations

Figure 29: Blockchain-based architecture design implementation use case I.



APPLICATION
Smart shipment contract 001*
State
Trade lane parties
• Customs import identity
Trade data
• Incoterm
• Invoice value
• Transport cost (multiple)
Smart import declaration*
State
Trade lane parties
• Declarant pseudonym ID
Trade data
• Smart shipment contract ID
• Encrypted customs value
Functions
Lodge import declaration
Confirmation policies
Function Lodge import declaration
Transaction validation
* Limited details (see chapter for more details).
Based on UN/CEFACT, WCO Data Model and GS1 standards.
DATA STRUCTURE AND STORAGE
Multiple blockchain ledgers
Blockchain ledger 1
Data of shipments in trade lane 1
Blockchain ledger n
Data of shipments in trade lane n
Account-based
Blockchain (order list of transactions)
Smart shipment contract state changes**
Smart import declaration state changes*
** See application component
World state (local node key/value storage)
Smart shipment contract state including encrypted individual data elements**
Smart import declaration state including encrypted customs value**
Decryption key distribution
Individual data element encryption
Parties with correct role in shipment only
External ABAC of individual data element owner
Customs authority identity certificates specified in state of smart shipment contract
CONSENSUS MECHANISM
Transaction process ('transfer to new holder' function)
Transaction execution
Transaction ordering
Transaction validation
Invocation and confirmation process
Issuing node (declarant)
Invoke smart import declaration function to create transaction with state write set
Confirming node (customs authority)
Invoke smart import declaration function to check state write set
Confirm write set via signature
Practical Byzantine Fault Tolerance
Ordering nodes
Order incoming transactions into blocks
List of ordering nodes (publicly available)*
* See network configuration component
Validation process
Validating nodes
Check transaction write set version
Check smart import declaration confirmation policy fulfillment
* See application component
NETWORK CONFIGURATION
Identity management (PKI)
(Public key) identity certificates
External
National eID CAs (e.g., EU eIDAS regulation)
Internal
Network RA
Network CA
(Public key) pseudonym identity certificates
Internal
Network PCA
Permission management
Network access permission
Any party with public key identity certificate
Blockchain ledger read permission
Parties involved in shipment only
Trade lane businesses pseudonym identity certificates specified in state of smart shipment contract
Customs authority identity certificates specified in state of smart shipment contract
Blockchain ledger write permission
Parties with correct role in shipment only
Trade lane businesses pseudonym identity certificates specified in state of smart shipment contract
Consensus participation permission
Authorized (consortium) parties only
List of authorized identity certificates maininted by consortium
Network nodes
Trade lane businesses
Technology companies*
Customs authorities
International organizations*
* Only act as ordering nodes
Governing consortium parties
Carriers
Freight Forwarders
Technology companies
Customs authorities
International organizations

Figure 30: Blockchain-based architecture design implementation use case II.

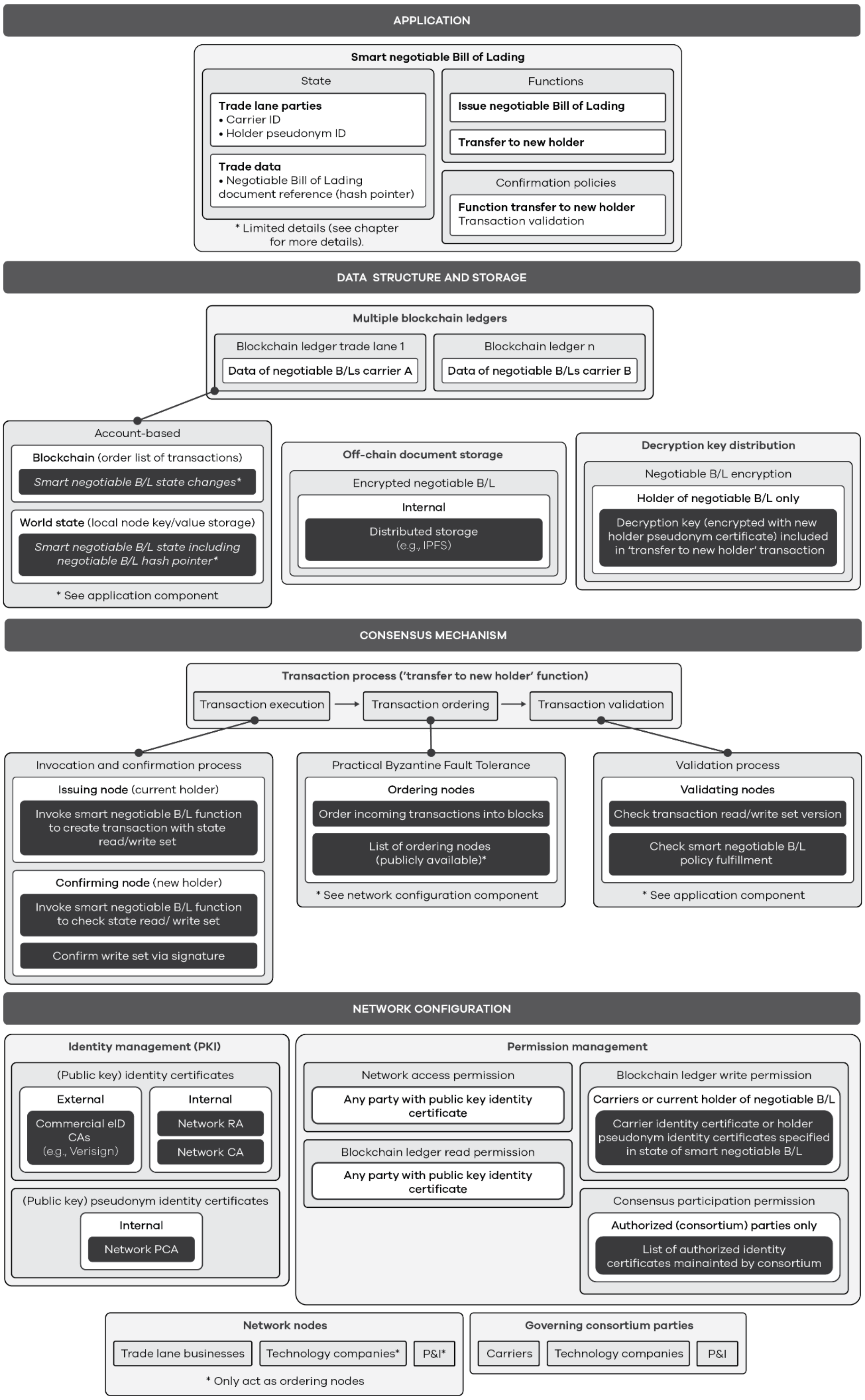

APPLICATION
Smart negotiable Bill of Lading
State
Trade lane parties
• Carrier ID
• Holder pseudonym ID
Trade data
• Negotiable Bill of Lading document reference (hash pointer)
* Limited details (see chapter for more details).
Functions
Issue negotiable Bill of Lading
Transfer to new holder
Confirmation policies
Function transfer to new holder
Transaction validation
DATA STRUCTURE AND STORAGE
Multiple blockchain ledgers
Blockchain ledger trade lane 1
Data of negotiable B/Ls carrier A
Blockchain ledger n
Data of negotiable B/Ls carrier B
Account-based
Blockchain (order list of transactions)
Smart negotiable B/L state changes*
World state (local node key/value storage)
Smart negotiable B/L state including negotiable B/L hash pointer*
* See application component
Off-chain document storage
Encrypted negotiable B/L
Internal
Distributed storage (e.g., IPFS)
Decryption key distribution
Negotiable B/L encryption
Holder of negotiable B/L only
Decryption key (encrypted with new holder pseudonym certificate) included in 'transfer to new holder' transaction
CONSENSUS MECHANISM
Transaction process ('transfer to new holder' function)
Transaction execution
Transaction ordering
Transaction validation
Invocation and confirmation process
Issuing node (current holder)
Invoke smart negotiable B/L function to create transaction with state read/write set
Confirming node (new holder)
Invoke smart negotiable B/L function to check state read/ write set
Confirm write set via signature
Practical Byzantine Fault Tolerance
Ordering nodes
Order incoming transactions into blocks
List of ordering nodes (publicly available)*
* See network configuration component
Validation process
Validating nodes
Check transaction read/write set version
Check smart negotiable B/L policy fulfillment
* See application component
NETWORK CONFIGURATION
Identity management (PKI)
(Public key) identity certificates
External
Commercial eID CAs (e.g., Verisign)
Internal
Network RA
Network CA
(Public key) pseudonym identity certificates
Internal
Network PCA
Permission management
Network access permission
Any party with public key identity certificate
Blockchain ledger read permission
Any party with public key identity certificate
Blockchain ledger write permission
Carriers or current holder of negotiable B/L
Carrier identity certificate or holder pseudonym identity certificates specified in state of smart negotiable B/L
Consensus participation permission
Authorized (consortium) parties only
List of authorized identity certificates mainainted by consortium
Network nodes
Trade lane businesses
Technology companies*
P&I*
* Only act as ordering nodes
Governing consortium parties
Carriers
Technology companies
P&I

Figure 31: Blockchain-based architecture design implementation use case III.

6. Architecture demonstration

After design, the next step is demonstration of the blockchain-based architecture design. The demonstration shows the functioning of the architecture components and its interaction with the problem domain parties (in this case trade lane parties). This chapter performs this demonstration. It answers sub question 5: *'How, for demonstration purposes, can the blockchain-based architecture design be used for the critical trade document use cases'*. Using the three critical trade document use cases as described in chapter 2, how the blockchain-based architecture design from chapter 5 supports the use cases is demonstrated to showcase the functioning of the components and how users (parties) interact with these components. To that extent, a set of typical user activities, based on the functional design requirements, are used. Section 6.1 demonstrates how the architecture supports use case I: *'Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration'*. In section 6.2, support of use case II *'Automated generation of an import declaration by aggregating already exchanged pro forma invoice data'* is demonstrated. Next, section 6.3 demonstrates support of use case III: *'Transfer title to the goods using an electronic negotiable Bill of Lading*. Section 6.4 provides conclusions that answer the sub question.

6.1 Demonstration use case I

This section demonstrates how the blockchain-based architecture design supports use case I. The use case serves two purposes, as defined in requirements UC1-req.1 and 2: 1) exchange of a pro forma invoice document by the consignor via a business-to-business exchange to enable end-to-end visibility and 2) the pull of the exchanged document by the customs authority that can use the document for piggybacking. The next two sub sections demonstrate these purposes for a shipment between consignor A and consignee B with shipment reference (GSIN) 73500538500000019. Customs authority C acts as customs authority on import side. Freight Forwarders D and E and carrier F are part of the trade lane of the shipment. Each party is represented by a node and has a copy of the blockchain ledger.

6.1.1 Exchange of a pro forma invoice document by the consignor

To exchange a pro forma invoice document, consignor A appends the document to a smart shipment contract (with corresponding GSIN) via an 'append pro forma invoice' transaction by invoking the 'append pro forma invoice' function. Input for the function is a hash pointer of the document. Thus, consignor A first has to generate the hash pointer. In addition, only authorized parties should have access to the pro forma invoice document. Therefore, consignor A generates a symmetric encryption/decryption key (see Box 2) and uses the key to encrypt the document. The document is stored in the consignor's document storage (e.g., ERP or cloud system). The key is stored in the key store of the ABAC infrastructure and an access policy which specifies that only the customs authority involved in the shipment and with a valid reason is authorized to access the document is stored in the Policy Access Point (PAP). Next, consignor A generates a hash pointer based on the encrypted pro forma invoice document. The consignor invokes the 'append pro forma invoice' function of the smart shipment contract (001) that belongs to the shipment to create the transaction proposal shown in Table 39. The confirmation policy of the 'append pro forma invoice' function requires the write set of the transaction to be signed by both consignor and consignee (see sub section 5.1.4). Therefore, consignor A sends a confirmation request message to consignee B, similar to the transaction proposal. The decryption key of the encrypted document is sent together with the message, so that consignee B can decrypt the document to check correctness. Consignee B invokes the 'append pro forma invoice' function of the contract to create a confirmation message that includes the signed write set (see Table 40) and sends the message to consignor A. Upon receiving the confirmation message from consignee B, consignor A creates the final 'append pro forma invoice' transaction proposal as depicted in Table 41. Next, the transaction is sent to the ordering nodes.

Table 39: Append pro forma invoice transaction proposal.

'Append pro forma invoice' transaction proposal	
Element	*Value*
Transaction ID	86e344d5434few … (hash)
Smart shipment contract ID	001
Contract function	Append pro forma invoice
Function input	Document reference: 44e3c1384d643476ecf8 …
Write set	Document reference: 44e3c1384d643476ecf8 …
Confirmation party 1 pseudo-PU	Pseudo-PUconsignorA
Confirmation party 1 signature	E(write set hash, pseudo-PRconsignorA)
Invoking party pseudo-PU	Pseudo-PUconsignorA
Invoking party signature	E(transaction proposal hash, pseudo-PRconsignorA)

Table 40: Append pro forma invoice transaction proposal confirmation message.

'Append pro forma invoice' transaction proposal confirmation message	
Element	*Value*
Transaction ID	86e344d5434few … (hash)
Write set	Document reference: 44e3c1384d643476ecf8 …
Confirmation party pseudo-PU	Pseudo-PUconsigneeB
Confirmation party signature	E(write set hash, pseudo-PRconsigneeB)
Confirmation party message signature	E(message hash, pseudo-PRconsigneeB)

Table 41: Final append pro forma invoice transaction proposal.

'Append pro forma invoice' final transaction	
Timestamp	
Ledger ID	
Element	*Value*
Transaction ID	86e344d5434few … (hash)
Smart shipment contract ID	001
Contract function	Append pro forma invoice
Write set	Document reference: 44e3c1384d643476ecf8 …
Confirmation party 1 pseudo-PU	Pseudo-PUconsignorA
Confirmation party 1 signature	E(write set hash, pseudo-PRconsignorA)
Confirmation party 2 pseudo-PU	Pseudo-PUconsigneeB
Confirmation party 2 signature	E(write set hash, pseudo-PRconsigneeB)
Invoking party pseudo-PU	Pseudo-PUconsignorA
Invoking party signature	E(transaction hash, pseudo-PRconsignorA)

The ordering nodes order incoming transactions and create a block of transactions following the Practical Byzantine Fault Tolerance (PBFT) consensus protocol (see sub section 5.1.3 for details on the ordering process). All parties within the trade lane that are represented by a node receive the block and append it to their own copy of the blockchain ledger as shown in Table 43. In the demonstration, Freight Forwarder D is used as example. Upon receiving a new block of final transaction proposals, Freight Forwarder D checks for each transaction, including transaction 86e344d5434few … that exchanges the pro forma invoice, if the confirmation policy is satisfied. The confirmation policy is retrieved via the contract ID. In case of an 'append pro forma invoice' transaction, the write set of the transaction has to be signed by consignor (A) and consignee (B) involved in the shipment (see Figure 23). To that extent, Freight Forwarder D queries the state of the smart shipment contract for the shipment to identify which parties act as consignor and consignee. For this shipment, these are Pseudo-PUconsignorA and Pseudo-PUconsigneeB (see Table 42). These parties correspond to the confirmation parties that signed the write set (see Table 41). The transaction can thus be deemed valid and Freight Forwarder D updates the world state using the write set specified in the transaction. This completes the exchange of the pro forma invoice and enables the Freight Forwarder to pull the exchanged document. The next sub section demonstrates the business-to-government exchange which involves customs authority C.

Table 42: Smart shipment contract state after transaction validation and world state update.

Smart shipment contract state	
Element	*Value*
Contract ID	001
Shipment reference number (GSIN)	73500538500000019
State version	1
Consignor ID	Pseudo-PUconsignorA
Freight Forwarder export ID	Pseudo-PUfreightforwarderD
Carrier ID	Pseudo-PUcarrierF
Freight Forwarder import ID	Pseudo-PUfreightfowarderE
Consignee ID	Pseudo-PUconsigneeB
Customs authority export ID	PUcustomsG
Customs authority import ID	PUcustomsC
Pro forma invoice document reference	44e3c1384d643476ecf8 …

Table 43: Block of transactions.

Block N			
Hash block N-1	Root hash transactions	State hash	Transactions (86e344d5434few …)

6.1.2 Pull of an exchanged pro forma invoice document by the customs authority

The previous sub section demonstrated how consignor A can exchange a pro forma invoice document via the architecture that enables end-to-end visibility of the document. In the second part of the use case the focus is on business-to-government exchange, in which customs authority C pulls the exchanged pro forma invoice document and piggybacks the document for manual-cross validation of the import declaration. Therefore, it is assumed that Freight Forwarder D acts as declarant and has already lodged the import declaration for the shipment with GSIN 73500538500000019 and customs authority C has found inconsistencies that require manual-cross validation using a pro forma invoice.

Customs authority C uses the GSIN specified on the import declaration to query its own copy of the blockchain ledger. The result is the state of smart shipment contract 001 that belongs to the shipment and contains all relevant data (see Table 42). Next, customs authority C uses the pro forma invoice document reference (hash pointer) to retrieve the pro forma invoice document from consignor's A document storage. However, the document is encrypted which prevents unauthorized access. To access the document, customs authority C has to request a decryption key from the document owner. The owner of the pro forma invoice document is consignor A and should therefore be requested to provide access to the encrypted document by providing the decryption key. To that extent, customs authority C sends a decryption key request message to the consignor specified in the state by pseudo-PUconsignorA (consignor A), which includes PUcustomsC to identify the requesting party, reason for access (cross-validation of the import declaration) and the pro forma invoice document reference as specified in the state of the smart shipment contract. The message is signed by customs authority C using PRcustomsC. Table 44 provides an overview of the message. The message arrives at the ABAC infrastructure of consignor A.

Table 44: Decryption key request message customs authority C.

Decryption key request message	
Element	*Value*
Document reference	44e3c1384d643476ecf8 …
Reason	Cross-validation import declaration
Requesting party PU	PUcustomsC
Requesting party signature	E(message hash, PRcustomsC)

Within the ABAC infrastructure, the Policy Enforcement Point (PEP) checks the signature of the message to determine validity. If valid, the message is sent to the Policy Decision Point (PDP). The PDP retrieves the access policy that belongs to the document reference from the PAP. In this case, the policy requires the requesting party to be a customs authority that is part of the trade lane of the shipment (with GSIN 735005385000000019) and has a valid reason to access the document. To that extent, the PDP queries its own copy of the blockchain ledger to identify the customs authority (on import side) that is part of the trade lane of the shipment (via the GSIN). For this shipment, the result would be PUcustomsC. In addition, the piggybacking of the pro forma invoice for manual cross-validation of the import declaration is a valid reason to request. This satisfies the access policy, as the requesting party is indeed the customs authority involved in the shipment and the reason for access is valid. Therefore, the PDP sends a message to the PEP allowing customs authority C access. The PEP retrieves the decryption key for the encrypted pro forma invoice document with reference 44e3c1384d643476ecf8 … from the key storage. A message is created that sends the decryption key to customs authority C. In the message, the decryption key is encrypted using the PU of the requesting party following E(decryption key, PUcustomsC). Table 45 provides an overview of the message.

After receiving the message, customs authority C retrieves the encrypted pro forma invoice document via the document reference from the document storage of consignor A. Next, it decrypts the encrypted document's decryption key following D(decryption key, PRconsignorA). Finally, the encrypted document is decrypted using the decryption key via D(document, decryption key). Customs authority C can then piggyback the decrypted pro forma invoice document for manual-cross validation of the import declaration.

Table 45: Decryption key return message customs authority C.

Decryption key return message	
Element	*Value*
Document reference	44e3c1384d643476ecf8 …
Encrypted decryption key	E(decryption key, PUcustomsC)
Document owner PU	PUconsignorA
Requesting party signature	E(message hash, PRcustomsC)

The next section demonstrates the blockchain-based architecture design that implements use case II.

6.2 Demonstration use case II

This section demonstrates the blockchain-based architecture design that supports use case II. Recall use case II in which a declarant lodges an import declaration that is automatically generated by aggregating already exchanged pro forma invoice data. The demonstration uses a shipment that is represented by smart shipment contract 001 (see section 6.1). The shipment is shipped under DDP incoterm with Freight Forwarder D acting as declarant of the import declaration. Customs authority C acts as customs authority on import side. Each party is represented by a node and has a copy of the blockchain ledger. The assumption is made that a smart shipment contract with relevant pro forma invoice data, including Incoterm, invoice value and transport cost (after entry) elements stored in its state is already available and the smart import declaration is deployed on the blockchain ledger.

To lodge the import declaration for the shipment, Freight Forwarder D invokes the 'lodge import declaration' function of the smart import declaration via a transaction proposal as shown in Table 46. The smart import declaration uses the smart shipment contract ID 001 as specified in the function input to get the state of the smart shipment contract (see Table 47). From the state, the contract retrieves the Incoterm (DDP), Invoice value (10,500) and transport cost after entry (500) data elements required to calculate the customs value. For encrypted data elements, Freight Forwarder D needs to request decryption keys similar to demonstrated in section 6.1.2. The contract selects the algorithm for calculation of the customs value based on the Incoterm. For DDP this means customs value = invoice value – transport cost after entry. For the shipment this results in customs value = 10,500 – 500 = 10,000. The customs value data element is encrypted similar to the pro forma invoice document in sub section 6.1.1. Next, the contract creates the write set consisting of declarant ID = pseudo-PUfreightforwarderD, smart shipment contract ID = 001 and customs value = 10,000 (encrypted). The write set is signed by Freight Forwarder D. The confirmation policy of the smart import declaration requires the write set of the transaction to be signed by both declarant and customs authority (see section 5.2.4). Therefore, Freight Forwarder D creates a transaction proposal confirmation request message as depicted in Table 48 and sends it to the customs authority specified in the state of the smart shipment contract, which for this shipment is customs authority C.

Customs authority C invokes the 'lodge import declaration' function of the smart import declaration via the transaction proposal confirmation request message. Similar to the invoking by Freight Forwarder D, the smart import declaration uses the smart shipment contract ID 001 to get the state of the corresponding smart shipment contract and calculate the customs value. Customs authority C decrypts the customs value decryption key specified in the confirmation request message and uses the key to encrypt the customs value. Next, the contract creates the write set consisting of declarant ID = pseudo-PUfreightforwarderD (invoking and confirming party in message), smart shipment contract ID = 001 and customs value = 10,000 (encrypted). As the write set of customs authority C matches the write set of Freight Forwarder D, the write set is deemed valid and customs authority C signs the write set[19]. Next, customs authority C creates a confirmation message as shown in Table 49. Upon receiving the confirmation message from customs authority C, Freight Forwarder D creates the final 'lodge import declaration' transaction as depicted in Table 50. Next, the transaction is sent to the ordering nodes.

Table 46: Lodge import declaration transaction proposal declarant.

'Lodge import declaration' transaction proposal	
Element	*Value*
Transaction ID	86e3d39fie4dea … (hash)
Smart import declaration ID	002
Contract function	Lodge import declaration
Function input	Smart shipment contract ID: 001
Invoking party pseudo-PU	Pseudo-PUfreightforwarderD
Invoking party signature	E(transaction proposal hash, Pseudo-PRfreightforwarderD)

Table 47: Smart shipment contract state.

Smart shipment contract state	
Element	*Value*
Contract ID	001
Shipment reference number (GSIN)	73500538500000019
State version	1
Consignor ID	Pseudo-PUconsignorA
Consignee ID	Pseudo-PUconsigneeB
Customs authority import ID	PUcustomsC
Incoterm	DDP
Invoice value	10,500
Transport cost (after entry)	500

Table 48: Lodge import declaration transaction proposal confirmation request declarant.

'Lodge import declaration' transaction proposal confirmation request	
Element	*Value*
Transaction ID	86e3d39fie4dea … (hash)
Smart import declaration ID	002
Contract function	Lodge import declaration
Function input	Smart shipment contract ID: 001

[19] The signature of customs is equivalent to the current electronic reply message that customs sends to confirm that a submitted import declaration is OK. This reply message includes the Movement Reference Number (MRN) message a customs authority assigns to a declaration.

Element	*Value*
Write set	Declarant ID: Pseudo-PUfreightforwarderD, Smart shipment contract ID: 001, Customs value: 10,000 (encrypted)
Customs value decryption key	E(decryption key, PUcustomsC)
Confirmation party 1 pseudo-PU	Pseudo-PUfreightforwarderD
Confirmation party 1 signature	E(write set hash, pseudo-PRfreightforwarderD)
Invoking party pseudo-PU	Pseudo-PUfreightforwarderD
Invoking party signature	E(transaction proposal hash, pseudo-PRfreightforwarderD)

Table 49: Lodge import declaration transaction proposal confirmation message customs.

'Lodge import declaration' transaction proposal confirmation message	
Element	*Value*
Transaction ID	86e3d39fie4dea … (hash)
Write set	Declarant: Pseudo-PUfreightforwarderD, Smart shipment contract ID: 001, Customs value: 10,000 (encrypted)
Confirmation party PU	PUcustomsC
Confirmation party signature	E(write set hash, PRcustomsC)
Confirmation party message signature	E(message, PRcustomsC)

Table 50: Lodge import declaration final transaction.

'Lodge import declaration' final transaction	
Timestamp	
Ledger ID	
Element	*Value*
Transaction ID	86e3d39fie4dea … (hash)
Smart import declaration ID	002
Contract function	Lodge import declaration
Write set	Declarant ID: pseudo-PUfreightforwarderD, Smart shipment contract ID: 001, Customs value: 10,000 (encrypted)
Confirmation party 1 pseudo-PU	Pseudo-PUfreightforwarderD
Confirmation party 1 signature	E(write set hash, pseudo-PRfreightforwarderD)
Confirmation party 2 PU	PUcustomsC
Confirmation party 2 signature	E(write set hash, PRcustomsC)
Invoking party pseudo-PU	Pseudo-PUfreightforwarderD
Invoking party signature	E(transaction hash, Pseudo-PRfreightforwarderD)

The ordering nodes order incoming transactions and create a block of transactions using PBFT (see sub section 5.1.3 for details on the ordering process). All parties within the trade lane that are represented by a node receive the block and append it to their own copy of the blockchain ledger. In the demonstration, customs authority C is used as example. Upon receiving a new block of transactions, customs authority C checks for each transaction including transaction 86e3d39fie4dea … that lodges the import declaration if the confirmation policy is satisfied. In case of an ‘lodge import declaration’ transaction, the write set of the transaction has to be signed by declarant (Freight Forwarder D) and customs authority (customs authority C) involved in the shipment (see Figure 23). To that extent, customs authority C queries the state of the smart shipment contract for the shipment (see Table 47) to identify which party acts as customs authority. For this shipment, this is customs authority C. In addition, it checks if the Declarant ID (pseudo-PUfreightforwarderD) in the write set matches the confirmation party (Pseudo-PUfreightforwarderD). Both parties correspond to the confirmation parties that signed the write set. The transaction can thus be deemed valid and customs authority C updates the world state using the write set specified in the transaction as shown in Table 51. The smart shipment contract ID is used to query the state of the corresponding smart shipment contract that contains the identity of both consignor and consignee. This completes the lodging of the import declaration and enables customs authority C to pull the import declaration.

Table 51: Smart import declaration state after transaction validation and world state update.

Smart import declaration state	
Element	*Value*
Contract ID	002
State version	1
Declarant ID	Pseudo-PUfreightforwarderD
Smart shipment contract ID	001
Customs value	10,000 (encrypted)

6.3 Demonstration use case III

In this section, the blockchain-based architecture design of use case III is demonstrated. The use case enables the transfer of title to the goods using an electronic negotiable Bill of Lading by transferring ownership of the Bill of Lading to a new holder. The next sub section demonstrates how business B that is the current holder of the electronic negotiable Bill of Lading with reference ABC1234567 can transfer ownership to new holder business C. The assumption is made that carrier A has already issued the smart negotiable Bill of Lading and a document reference in the form of a hash pointer is stored in the state of the smart negotiable Bill of Lading. Each party is represented by a node and has a copy of the blockchain ledger.

To transfer the electronic negotiable Bill of Lading to business C, business B invokes the ‘transfer to new holder’ function of the smart negotiable Bill of Lading with reference ABC1234567 via a preliminary transaction proposal to create the transaction proposal as shown in Table 52. Input for the function is the identity of the new holder: pseudo-PUbusinessC[20]. The confirmation policy of the ‘transfer to new holder’ function requires the read and write sets of the transaction to be signed by current holder (business B) and a new holder (in this case business C). Therefore, business B sends a confirmation request message to business C, similar to the transaction proposal. The decryption key of the encrypted Bill of Lading is sent together with the transaction, so that business C can decrypt

[20] The pseudo-PUbusinessB and pseudo-PUbusinessC are shared via a separate encrypted message between PUbusinessB and PUbusinessC.

the document to check correctness. Business C invokes the ‘transfer to new holder’ function of the contract to create the read (holder = pseudo-PUbusinessB, state version = 1) and write (holder = pseudo-PUbusinessC) set. As these sets match with the transaction proposal of business B, business C signs the read and write sets. Next it sends a confirmation message as shown in Table 53 to business B. Upon receiving the confirmation message from business C, business B creates the final ‘transfer to new holder’ transaction as depicted in Table 54. Next, the transaction is sent to the ordering nodes.

Table 52: Transfer to new holder transaction proposal.

‘Transfer to new holder’ transaction proposal	
Element	*Value*
Transaction ID	43d324d5843adf … (hash)
Smart negotiable Bill of Lading contract ID	001
Contract function	Transfer to new holder
Function input	Holder: Pseudo-PUbusinessC
Negotiable Bill of Lading decryption key	E(decryption key, PUbusinessC)
Read set	Holder: Pseudo-PUbusinessB, State version: 1
Write set	Holder: Pseudo-PUbusinessC
Confirmation party 1 pseudo-PU	PUbusinessB
Confirmation party 1 signature	E(read/write set hash, PUbusinessB)
Invoking party pseudo-PU	PUbusinessB
Invoking party signature	E(Transaction proposal hash, PRbusinessB)

Table 53: Transfer to new holder transaction proposal confirmation message.

‘Transfer to new holder’ transaction proposal confirmation message	
Element	*Value*
Transaction ID	43d324d5843adf … (hash)
Read set	Holder: pseudo-PUbusinessB, State version: 1
Write set	Holder: pseudo-PUbusinessC
Confirmation party pseudo-PU	Pseudo-PUbusinessC
Confirmation party signature	E(read/write set hash, pseudo-PUbusinessC)
Confirmation party message signature	E(message hash, pseudo-PRbusinessB)

Table 54: Transfer to new holder final transaction.

'Transfer to new holder' final transaction	
Timestamp	
Ledger ID	
Element	*Value*
Transaction ID	86e344d5434few … (hash)
Smart shipment contract ID	001
Contract function	Transfer to new holder
Negotiable Bill of Lading decryption key	E(decryption key, PUbusinessC)
Read set	Holder: pseudo-PUbusinessB, State version: 1
Write set	Holder: pseudo-PUbusinessC
Confirmation party 1 pseudo-PU	Pseudo-PUconsignorA
Confirmation party 1 signature	E(read/write set hash, pseudo-PRconsignorA)
Confirmation party 2 ID	Pseudo-PUconsigneeB
Confirmation party 2 signature	E(read/write set hash, pseudo-PRconsigneeB)
Invoking party ID	Pseudo-PUconsignorA
Invoking party signature	E(transaction hash, pseudo-PRconsignorA)

The ordering nodes order incoming transactions and create a block of transactions using PBFT (see sub section 5.1.3 for details on the ordering process). All parties that are represented by a node receive the block and append it to their own copy of the blockchain ledger. In the demonstration, business D is used as example. Upon receiving a new block of transactions, business D checks for each transaction including transaction 86e344d5434few … that transfers the electronic negotiable Bill of Lading to a new holder if the confirmation policy is satisfied. In case of an 'transfer to new holder' transaction, the read and write sets of the transaction has to be signed by the current holder (business B) and a new holder (in this case business C) involved in the transfer. To that extent, business D queries the state of the smart negotiable Bill of Lading with reference ABC1234567 to identify which party is the current holder. For this negotiable Bill of lading, the current holder is pseudo-PUbusinessB. This party matches one of the confirmation parties in the final transaction shown in Table 54. In addition, another valid confirmation party signature is included in the transaction. Next, business D checks if the read set matches with the current state of the smart negotiable Bill of Lading by comparing the state version. As this is the case, the transaction can therefore be deemed valid and business D updates the world state using the write set which includes an update of the state version to 2. The new state is shown in Table 55. This completes the transfer of ownership of the electronic negotiable Bill of Lading with reference ABC1234567 to business C.

Table 55: Smart shipment contract state after transaction validation and world state update.

Smart negotiable Bill of Lading state	
Element	*Value*
Contract ID	001
Negotiable Bill of Lading reference number	ABC1234567
State version	2
Carrier ID	PUcarrierA
Holder ID	Pseudo-PUbusinessC
Negotiable Bill of Lading document reference	44e3c1384d643476ecf8 …

6.4 Conclusion chapter 6

Using the three use cases from chapter 2 to demonstrate the blockchain-based architecture, this chapter answers sub question 5: '*How, for demonstration purposes, can the blockchain-based architecture design be used for the critical trade document use cases?*'. The demonstration shows that the use cases can be implemented using the blockchain-based architecture design. It demonstrates basic functioning of the architecture using a set of typical user activities.

In use case I, to exchange the pro forma invoice the consignor invokes the smart shipment contract's 'append pro forma invoice' function by providing a document reference as input. Correctness of the pro forma invoice is confirmed by the consignee and the 'append pro forma invoice' transaction is sent to the ordering nodes by the consignor. The ordering nodes create a block containing the transaction and broadcast the block to all other nodes. These nodes validate the transaction and update the world state accordingly. A customs authority uses the document reference specified in the state to retrieve the pro forma invoice document. A decryption key is requested from the consignor in order to access the document.

In use case II, to lodge an import declaration the declarant (e.g., Freight Forwarder) invokes the smart import declaration's 'lodge import declaration' function by providing a smart shipment contract ID as input. The smart import declaration retrieves relevant pro forma invoice data (Incoterm, invoice value, transport cost after entry) from the state of the smart shipment contract. It calculates the customs value via an algorithm based on the Incoterm. Correctness of calculation is confirmed by the customs authority and the 'lodge import declaration' transaction is sent to the ordering nodes by the declarant. The ordering nodes create a block containing the transaction and broadcast the block to all other nodes. These nodes validate the transaction and update the world state of the smart import declaration accordingly.

In use case III, to transfer ownership of an electronic negotiable Bill of Lading to a new holder, the current holder invokes the smart negotiable Bill of Lading's 'transfer to new holder' function by providing the pseudo-PU of the new holder as input. The new holder confirms correctness by checking the negotiable Bill of Lading and comparing the read sets of both current and new holder to see if they match. Next, the 'transfer to new holder' transaction is sent to the ordering nodes by the current holder. The ordering nodes create a block containing the transaction and broadcast the block to all other nodes. These nodes validate the transaction and update the world state of the smart negotiable Bill of Lading to specify the new holder as holder accordingly.

The next chapter evaluates the blockchain-based architecture design.

7. Architecture evaluation

After demonstration, the fifth step in the research approach is evaluation of the blockchain-based architecture design. This chapter evaluates the blockchain-based architecture design to answer sub question 6: *'To what extent, for evaluation purposes, does the blockchain-based architecture design fit with the design requirements and problem domain of the critical trade document use cases?'*. The evaluation is conducted using two methods. First, the blockchain-based architecture design is mapped on the design requirements of each critical trade document use case to evaluate if the defined requirements are fulfilled, which is presented in section 7.1. Next, interviews with experts are used to evaluate the relevance of the use cases and how the blockchain-based architecture design supports these use cases to assess its effectiveness and technical correctness. The expert evaluation is presented in section 7.2. Based on the evaluations, design improvements for the blockchain-based architecture are discussed in section 7.3. Last, 7.4 concludes the chapter by answering the sub question.

7.1 Architecture evaluation based on design requirements

This section maps the blockchain-based architecture design on the design requirements of each critical trade document use case as defined in chapter 3 (see Table 6) to evaluate the fit of the architecture design with the design requirements. For each critical trade document use case, the fit of the design requirements is discussed in the next three sub sections.

7.1.1 Evaluation architecture use case I

In this sub section, the blockchain-based architecture design is evaluated using the requirements for use case I: '*Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross validation of an import declaration*'. Each requirement is discussed separately.

UC1-req. 1: The architecture should provide business-to-business exchange of (a pro forma invoice) document to enable end-to-end visibility

To fulfill the requirement, the consignor should be able to exchange the pro forma invoice document with other businesses within a trade lane to enable end-to-end visibility. In the architecture design, each business in the trade lane is represented by a node in the blockchain network that holds its own copy of the blockchain ledger. Within the blockchain ledger, each shipment is represented as smart shipment contract in which a hash pointer that provides a link reference to the document is stored. The document is stored in the consignor's own off-chain document storage (e.g., ERP or cloud storage). A consignor can exchange a hash pointer by invoking the smart shipment contract's 'append pro forma invoice' function to create a transaction that includes the hash pointer in its write set. Via the consensus mechanism all other nodes (businesses) that hold a copy of the blockchain ledger receive the transaction that contains the hash pointer and use the transaction to update the state of the smart shipment contract. This fulfills the requirement as businesses in a trade lane can access their copy of the blockchain ledger which contains the hash pointer stored in the state of the smart shipment contract and use the hash pointer to retrieve the pro forma invoice document from the consignor's document storage which enables end-to-end visibility.

UC1-req. 2: The architecture should provide business-to-government exchange to enable customs authorities to piggyback an exchanged document for manual cross-validation of an import declaration

In order to enable piggybacking of a business-to-business exchanged pro forma invoice document, the customs authority should be able to pull the encrypted pro forma invoice document and access the encrypted document via a business-to-government exchange. To that extent, a customs authority is represented by a node in the blockchain network and holds a copy of the blockchain ledger of a trade lane for a specific shipment. Similar to businesses, the customs authority thus has access to the smart shipment contract state that contains the hash pointer of the pro forma invoice document. A customs authority can use this pointer to pull the document from the off-chain document storage. However, as the document is encrypted, retrieving the document does not provide access to the actual data within the document. A decryption key is needed to decrypt the document in order to access the data and enable piggybacking. For that reason, the smart shipment contract state also contains the consignor's pseudo-PU. A customs authority can use the pseudo-PU to send a message to the consignor in order to request the decryption key for the encrypted document. Using the decryption key, the customs authority can decrypt the pro forma invoice document and access the required data in the document for piggybacking purposes. Thus, the blockchain-based architecture design fulfills this requirement as it enables customs authorities to pull and piggyback the pro forma invoice document that can then be used for manual-cross validation of the import declaration.

UC1-req. 3: The architecture should ensure confidentiality of both the exchanged document and identities of businesses involved in the exchange transaction and should allow only authorized parties access to the document or identity to overcome legality and liability issues
To ensure confidentiality of the pro forma invoice document and identities only authorized parties should be able to access the document or identity of businesses involved. First fulfillment of confidentiality of the document by the architecture is evaluated. Documents are encrypted before being stored on the consignor's off-chain document storage and can only be accessed by a party in possession of the decryption key. The consignor manages its own decryption key management and can decide who is authorized to access the document and should be provided with a decryption key by specifying access policies. To that extent, two scenarios to showcase the architecture can ensure confidentiality are considered: 1) a customs authority that wants to piggyback the pro forma invoice document for manual cross-validation of an import declaration should be provided with a decryption key in order to access the document and 2) a carrier should not be provided with a decryption key to access the document in order to avoid liability issues by having access to the data within the pro forma invoice. One access policy states that if the requesting customs authority is involved in the trade lane of the shipment, as specified in the smart shipment contract state (PUcustoms), and there is a valid reason (piggyback for manual-cross validation) a decryption key is provided. For a carrier there is no access policy specified. If the carrier would request a decryption key it is not provided with a decryption key as no access policy is fulfilled. A non-authorized party can thus retrieve the pro forma invoice document, but because the document is encrypted it cannot see the actual data within the document. A party still needs the decryption key in order to access the document. This fulfills the requirement of confidentiality of the document as only authorized parties can see the actual data within the pro forma invoice document.

Next, fulfillment of confidentiality of the identities of businesses in the architecture is evaluated. The use of pseudo-PU identity certificates within the architecture enables business to keep their real-world identity confidential while still being involved in document exchanges. A party that is not authorized to see the real-world identity (PU) of the businesses involved in a shipment, will only see the pseudo-PU in the state of the smart shipment contract or in the 'append pro forma invoice' transaction. An authorized party can see the PU as a business can choose to reveal the PU to another party by exchanging the PU. The example presented in section 3.1.2 of a carrier not being authorized to see the PU of the consignor as it could reduce the willingness of the Freight Forwarder to use the architecture is used to illustrate this. In this case the carrier should not be able to see the PU of the consignor in its copy of the blockchain ledger. The difference in perspective between the not-authorized carrier and authorized Freight Forwarder is shown in Table 56 and Table 57. Note that the pseudo-PU is still stored in the transaction and contract state and manual action is required to show the PU by retrieving it from a local (pseudo)-PU storage.

In conclusion, the use of encryption and decryption key management by the document owner (consignor) to ensure confidentiality of the pro forma invoice document and the use of pseudo-PU identity certificates by businesses in the architecture fulfills the requirement.

Table 56: *Transaction perspective carrier and Freight Forwarder.*

Transaction element	**Value**	
	Carrier perspective	*Freight Forwarder perspective*
Invoking party pseudo-PU	04 bb 50 a9 5f 4a 4a … (pseudo-PUconsignorA)	Consignor A (PUconsignorA)

Table 57: *Smart shipment contract state perspective carrier and Freight Forwarder.*

State element	**Value**	
	Carrier perspective	*Freight Forwarder perspective*
Consignor ID	04 bb 50 a9 5f 4a 4a … (pseudo-PUconsignorA)	Consignor A (PUconsignorA)

UC1-req. 4: The architecture should ensure integrity of the exchanged document
To rely on correctness of the data within an exchanged pro forma invoice document, customs authority has to be ensured that once exchanged no alterations can be made to the document. In other words, there should be proof of integrity of the document. In the architecture, proof of integrity is realized in several steps. One of the main concepts of blockchain technology is that a blockchain ledger is (in theory) immutable as it relies on hashing and a consensus mechanism to append new blocks of transactions that can update the state of the ledger. As a result, a document reference stored in a transaction is immutable and thus cannot be altered or modified in any way. If a fraudulent consignor would try to alter a transaction to store a new document reference, the hash of the block that contains the transaction would change, thus breaking the cryptographic link between blocks as block N contains the hash of the previous block N-1. This proofs integrity of the reference stored on the blockchain ledger. However, in the architecture only the document reference is stored on the blockchain ledger. The pro forma invoice document itself is stored in the consignor's off-chain document storage. Storing only a URL on the blockchain ledger as reference to the document would not suffice as proof of integrity. A fraudulent consignor could still alter the document whilst keeping the same URL as reference. To overcome this issue, in the architecture the pro forma invoice document is first hashed to create a hash pointer. The hash pointer is used as document reference and is stored on the blockchain ledger to proof integrity of the document. The two steps the architecture takes, hashing the document and storing the hash pointer on the immutable blockchain ledger fulfill the requirement to ensure integrity of the exchanged document.

UC1-req. 5: The architecture should be open to businesses, without the use of a centralized document storage and without a central authority in control
Any businesses involved in a shipment should in essence have access to the blockchain network without a central authority deciding which businesses can or cannot access. However, as the architecture uses a permissioned network configuration topology, some form of identity and permission management is required. A governing consortium consisting of a representation of trade lane businesses like Freight Forwarders and carriers, customs authorities and international organizations like the WTO or EU DG TAXUD controls the identity and permission management. This ensures non-arbitrary treatment of business that want to use the architecture to exchange trade documents as there is no central authority in control. Within the blockchain-based architecture design, a component that could conflict with the requirement of having no central authority in control is the consensus mechanism. The architecture uses Practical Byzantine Fault Tolerance (PBFT) as consensus protocol that can only handle a limited number of ordering nodes that participate. This limitation could create the impression that a central authority is in control of the consensus protocol and thus the world state of the blockchain ledger. To mitigate this potential conflict, the set of ordering nodes is composed of a representation of the governing consortium parties to ensure non-

arbitrary treatment. In addition, ordering nodes do not have a copy of the blockchain ledger. Their only task is to order incoming transactions and send blocks of ordered transactions to all other nodes. No further actions are performed on the transactions. This should provide businesses with sufficient trust that no central authority is in control. Next to openness and control, another element of this requirement is that the architecture should not use a centralized document storage. However, to not conflict with UC1-req. 1 concerning business-to-business exchange of a pro forma invoice some form of centralized storage is required to enable exchange. Therefore, the architecture stores only a pro forma invoice document reference on the blockchain ledger. The actual document is stored in the consignor's off-chain document storage. While to some extent the storage of a document reference can be seen as centralized storage (as all nodes that have a copy of the blockchain ledger can access it), a reference alone is not sufficient to actually access the document. This fulfills the storage requirement as the document is stored decentralized with the owner (in case of a pro forma invoice the consignor) in control over access to the document. Thus, the architecture design fulfills the requirement as it provides open access to the architecture, uses a decentralized document storage and is governed by a consortium consisting of different types of parties.

UC1-req. 6: The architecture should be able to handle high volumes of document exchanges in real-time
An important requirement of the blockchain-based architecture design is scalability. For the exchange of pro forma invoice documents, the transaction throughput should be at least ten transactions per second for the Netherlands alone as discussed in section 3.1.2. If the architecture should be able to handle transactions outside the Netherlands, for example within the EU customs territory, a multitude of ten transactions per second is needed. In addition, latency should be low to enable real-time exchange. This paragraph therefore evaluates how the designed architecture is scalable compared to blockchain-based architectures that rely on Proof-of-Work as consensus protocol.

In case of the Proof-of-Work, on average each ten minutes a valid nonce for a block proposal is found by a mining node because of the built-in delay. Due to the limited block size, this results in a throughput of eight to fifteen transactions per second. In addition, it would take up to sixty minutes before the 'append pro forma invoice' transaction can be considered valid due to the forking issue in Proof-of-Work. If a pro forma invoice document would be exchanged via transactions in an architecture that uses Proof-of-Work, an additional delay on top of the ten minutes needed for finding a valid nonce for the block proposal would thus occur. Compared to an architecture that uses Proof-of-Work, in the architecture design that uses PBFT there is no delay in appending new blocks of transaction to the blockchain ledger as only the leader can propose blocks and there is no need for validating nodes to wait six blocks before deeming a transaction valid because forking cannot occur. The only delay in terms of network latency that can occur is the sending of messages between ordering nodes which can take up to a few seconds. As a result, PBFT can reach a throughput of tens of thousands of transactions per second which is more than sufficient to handle all 'append pro forma invoice' transactions without delay. Table 58 provides an overview of the comparison.

The comparison shows that the use of Proof-of-Work as consensus protocol in the architecture cannot fulfill the requirement. On the contrary, PBFT can fulfill the requirement. In conclusion, the architecture fulfills the requirement to be able to handle high volumes of document exchanges in near real-time as PBFT which enables high transaction throughput and low latency is used. The next sub section evaluates the blockchain-based architecture design using the requirements for use case II.

Table 58: Comparison consensus protocol scalability requirement use case I.

Required		**Consensus protocol**	
		PBFT	*Proof-of-Work*
Transaction throughput	10+ transactions per second	10,000+ transactions per second	8 – 15 transactions per second
Delay	Real-time	Up to a few seconds (network latency)	60 minutes

7.1.2 Evaluation architecture use case II

This sub section evaluates the blockchain-based architecture design using the requirements for use case II: *'Automated generation of an import declaration by aggregating already exchanged pro forma invoice data'*. Each requirement is discussed separately.

UC2-req. 1: The architecture should enable a declarant to automatically generate an import declaration by retrieving already business-to-business exchanged pro forma invoice data and calculate customs value, and allow business-to-government exchange of the generated import declaration

A declarant is represented by a node in the blockchain network and has a copy of the blockchain ledger that contains the pro forma invoice data in the state of the smart shipment contract. A smart import declaration is deployed on the blockchain ledger that implements all functionality to automatically aggregate individual data elements of the pro forma invoice (e.g., Incoterm, invoice value and transport cost) based on the smart shipment contract ID that is specified by the declarant. This fulfills the retrieval element of the requirement. The calculation procedure is implemented in the smart negotiable Bill of Lading. As the Incoterm is stored in the state of the smart shipment contract, simple retrieval of the Incoterm is sufficient to select the appropriate calculation procedure in the smart import declaration to calculate the customs value. This fulfills the calculation element of the requirement. Via the consensus mechanism all other nodes including the customs authority that hold a copy of the blockchain ledger receive the transaction that contains the smart shipment contract ID and customs value in the write set and use the transaction to update the state of the smart import declaration. To that extent, they can access their own copy of the ledger and read the state of the smart import declaration. This fulfills the requirement that the customs authority should be able to access the import declaration. In conclusion, the architecture fulfills the requirement as a declarant can automatically generate an import declaration and the customs authority can access the generated import declaration.

UC2-req. 2: The architecture should guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value

To fulfill the requirement, there should be a guarantee that pro forma invoice data elements are correctly retrieved and the customs value is calculated correctly. If the architecture lacks this guarantee, customs authorities cannot ensure the smart import declaration is correct. As a result, customs authorities would have to revert back to manual cross-validation of the import declaration and pro forma invoice (see use case I), which would take away the benefits of automation of the import declaration to reduce time and costs for importation of goods. The guarantee of correct retrieval and calculation consists of two elements: 1) correctly implement the retrieval of pro forma invoice data and calculation of the customs value in code and 2) ensure that each time the smart import declaration 'lodge import declaration' function is invoked the same output is generated. The latter relates to the proof of integrity (see UC1-req. 3). Both elements are discussed in the next paragraph.

Smart import declarations are assumed to be developed by Freight Forwarders as they often act as declarant of the import declaration. Customs authorities provide the specifications for correctly implementing the retrieval of pro forma invoice data and customs value calculation. Calculation is standardized. For example, if a shipment is shipped under DDP incoterm, the customs value should be calculated by subtracting the transport cost after point of entry from the invoice value. Before being deployed on the blockchain ledger, a customs authority performs an audit on the code to guarantee correct implementation. A signature is created as sign of approval. Once deployed on the blockchain ledger via a transaction, the smart import declaration becomes immutable. This guarantees that the smart import declaration cannot be altered and thus complies with the specification provided by the customs authority. In addition, the pro forma invoice data is stored in the state of the smart shipment contract on the blockchain ledger and is therefore immutable. This ensures that each time the smart import declaration is invoked, the same pro forma invoice data is retrieved correctly and the customs value is calculated correctly. However, if only the declarant would sign the write set of the transaction, customs authorities cannot be ensured of correctness as no party confirmed that the smart import declaration was correctly invoked. The customs authority therefore acts as confirming node and invokes the function and signs the write set as well. As a result, the declarant cannot alter the write set. Because the confirmation policy specifies that the customs authority involved has to sign the write set in order to create a valid transaction, the declarant cannot use a fraudulent party to sign the write set to create a valid transaction. The architecture thus fulfills the requirement as it guarantees correct retrieval of pro forma invoice data and calculation of the customs value.

UC2-req. 3: The architecture should be able to generate high volumes of import declarations in real-time
For the automated generation of import declarations within the Netherlands, the transaction throughput should be at least ten transactions per second as each transaction represents generation of one import declaration. If the architecture should be able to handle transactions outside the Netherlands, for example within the EU customs territory, a multitude of ten transactions per second is needed. In addition, latency should be low to enable real-time generation. This paragraph therefore evaluates how the designed architecture is scalable compared to blockchain-based architectures that rely on Proof-of-Work as consensus protocol.

In a blockchain-based architecture that uses Proof-of-Work, on average each ten minutes a valid nonce is found by a mining node because of the built-in delay. Due to the limited block size, this results in a throughput of eight to fifteen transactions. In addition, it would take up to sixty minutes before the 'lodge import declaration' transaction can be deemed valid due to the forking issue in Proof-of-Work. If import declaration would be automatically generated within an architecture that uses Proof-of-Work, an additional delay on top of the ten minutes needed for finding a valid nonce for the block proposal would occur. Compared to Proof-of-Work, in the architecture design that uses PBFT there is no delay in appending new blocks of transaction to the blockchain ledger as only the leader can propose blocks and there is no need for validating nodes to wait six blocks before deeming a transaction valid because forking cannot occur. The only delay in terms of latency that can occur is the sending of messages between ordering nodes which can take up to a few seconds. As a result, PBFT can reach a throughput of tens of thousands of transactions per second which is sufficient to handle all 'lodge import declaration' transactions without delay. Table 59 provides an overview of the comparison.

The comparison shows that the use of Proof-of-Work as consensus protocol in the architecture cannot fulfill the requirement. On the contrary, PBFT can fulfill the requirement. In conclusion, the architecture fulfills the requirement to be able to generate high volumes of import declarations in near real-time as PBFT which enables high transaction throughput and low latency is used.

The next sub section evaluates the blockchain-based architecture design using the requirements for use case III.

Table 59: Comparison consensus protocol scalability requirement use case II.

Required		**Consensus protocol**	
		PBFT	*Proof-of-Work*
Transaction throughput	10+ transactions per second	10,000+ transactions per second	8 – 15 transactions per second
Delay	Real-time	Up to a few seconds (network latency)	60 minutes

7.1.3 Evaluation architecture use case III

In this sub section, the blockchain-based architecture design is evaluated using the requirements for use case III: *'Transfer title to the goods using an electronic negotiable Bill of Lading'*. Each requirement is discussed separately. Multiple initiatives for a blockchain-based electronic negotiable Bill of Lading exist (e.g., CargoX). The evaluation therefore includes a comparison of the architecture with the CargoX initiative to show how the architecture overcomes the shortcomings of the initiative. To that extent, evaluation of UC3-req. 5 on scalability includes a comparison with CargoX as the initiative does not fulfill the requirement because it cannot sufficiently scale.

UC3-req. 1: The architecture should provide business-to-business exchange to transfer ownership of the electronic negotiable bill of lading to a new holder
To fulfill the requirement, the current holder should be able to exchange a transaction to transfer ownership of the electronic negotiable Bill of Lading to a new holder. In the architecture design, each business involved (both current and new holder) is represented by a node in the blockchain network that holds its own copy of the blockchain ledger. Within the blockchain ledger, each negotiable Bill of Lading is represented as smart negotiable Bill of Lading in which a hash pointer that provides a link reference to the document and the pseudo-PU of the current holder is stored. The current holder can transfer ownership by invoking the smart negotiable Bill of Lading's 'transfer to new holder' function to create a transaction that includes the pseudo-PU of the new holder in its write set. Via the consensus mechanism all other nodes (businesses) that hold a copy of the blockchain ledger receive the transaction and use the transaction to update the state of the negotiable Bill of Lading. This fulfills the requirement as businesses can access their copy of the blockchain ledger which contains the pseudo-PU of the current holder in the state of the smart negotiable Bill of Lading that reflects the current holder of the negotiable Bill of Lading.

UC3-req. 2: The architecture should guarantee uniqueness of the electronic negotiable Bill of Lading following the principles for functional equivalence
The most critical requirement for use case III is the guarantee of uniqueness of the electronic negotiable Bill of Lading. Therefore, fulfillment of the requirement is discussed using the four principles for functional equivalence an electronic negotiable Bill of Lading should follow: 1) provide exclusive control over issuance and transfer of the electronic negotiable Bill of Lading to the intended holder, 2) assure the electronic negotiable Bill of Lading retains its integrity, 3) enable the holder to demonstrate ownership of the electronic negotiable Bill of Lading and 4) provide confirmation that delivery to the holder has been effected or that the electronic negotiable Bill of Lading has ceased to have any effect or validity. First, exclusive control over issuance and transfer to the new holder is discussed. Two scenarios for issuance of a negotiable Bill of Lading are to be considered: 1) a carrier should be able to create a valid 'issue a new negotiable Bill of Lading' transaction and 2) any other party should not be able to create a valid transaction. To provide exclusive control over issuance, within the blockchain-based architecture the smart negotiable Bill of Lading only allows a business that has role attribute: 'carrier' to invoke the contract's 'issue negotiable Bill of Lading' function in its

PU identity certificate to create a valid transaction. The certificate is issued by the internal CA to ensure that only carriers can be issued such identity certificate. Each 'issue negotiable Bill of Lading' transaction write set is required to be signed by a party with such certificate in order to be deemed valid. This keeps other (fraudulent) parties from issuing false electronic negotiable Bill of Ladings that can be sold to other parties as no valid signature and thus valid transactions can be created.

For exclusive control over transfer to the new holder, again two scenarios are considered: 1) the current holder should be able to create a valid 'transfer to new holder' transaction and 2) any other party should not be able to create a valid transaction. To determine if a transaction is valid the architecture uses a confirmation policy and read set check within the consensus mechanism. The confirmation policy all validating nodes check guarantees that only transactions that fulfill the confirmation policy are deemed valid. For the 'transfer to new holder' function, the confirmation policy requires that the read and write sets of the transactions are signed by the current holder and the new holder. In the state of the smart negotiable Bill of Lading the pseudo-PU of the current holder is stored as shown in Figure 32. A validating node uses this pseudo-PU to check whether the current holder has signed the read set by comparing the pseudo-PUs. If these two match, the confirmation policy is fulfilled. Only the current holder is in possession of the PR that is used to create a valid signature using the pseudo-PU. This guarantees that no other party than the holder can create a valid signature and thus create a valid transaction that transfers ownership. In addition, the read set is compared to the state of the smart negotiable Bill of Lading to check if the state version and current holder are equal to overcome the double spend problem. This guarantees that the current holder has exclusive control over the smart negotiable Bill of Lading because it is the only party that can create a valid transaction to transfer ownership.

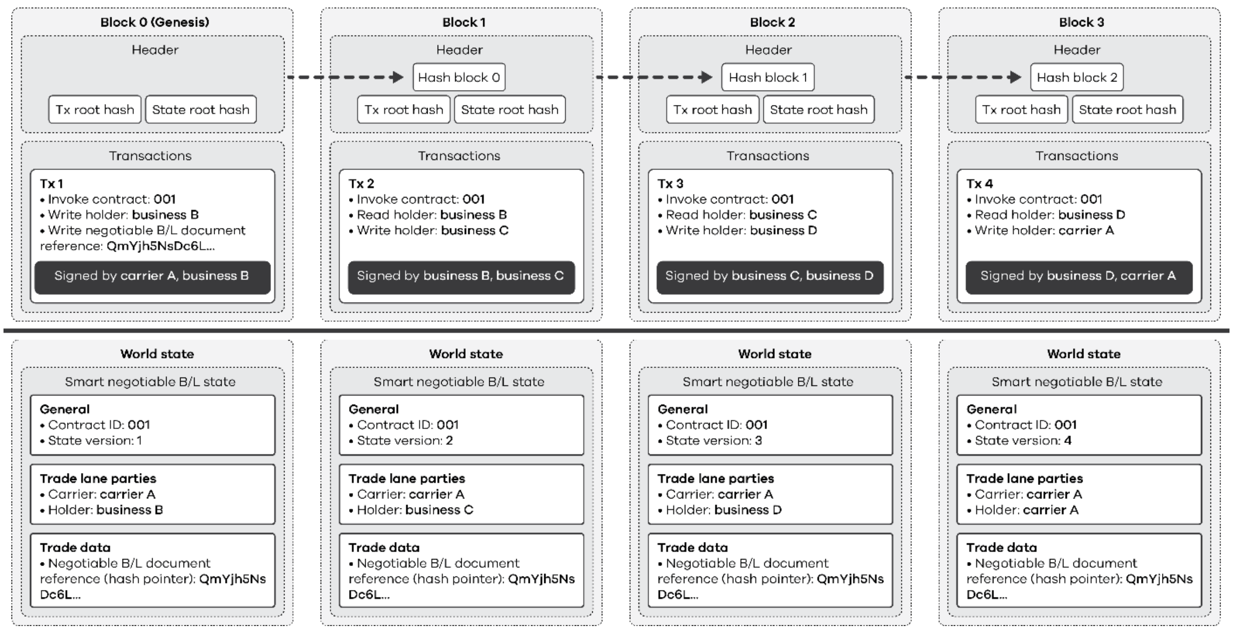


Figure 32: Guarantee of uniqueness of the electronic negotiable Bill of Lading using the blockchain-based architecture.

The second principle for functional equivalence that should be followed by the architecture is the assurance that the electronic negotiable Bill of Lading retains its integrity throughout its lifecycle. To that extent, the document reference to the negotiable Bill of Lading is a hash pointer that is stored on the blockchain ledger via the write set of a transaction as depicted in Figure 32. Transactions stored in a block on the blockchain ledger are immutable. In addition, each node has a copy of the blockchain ledger. Only if all nodes would reach consensus on altering the document integrity fails, which with great certainty will not happen. This ensures that the document cannot be altered by a single party during its lifecycle which completes the principle for retaining integrity.

The third principle is that the holder of the electronic negotiable Bill of Lading should be able to demonstrate ownership. In the architecture, the holder can demonstrate ownership by creating a valid signature using the pseudo-PU specified in the state of the smart negotiable Bill of Lading and the PR that belongs to the pseudo-PU. No other party is able to create a valid signature without the PR which enables only the holder to demonstrate ownership.

The final principle is that confirmation of delivery to the holder, in order words ownership has been transferred to the holder, has been effected or the electronic negotiable Bill of Lading has ceased to have any effect or validity. Ownership is transferred via a 'transfer to new holder' transaction. Once the transaction is added to a block which is appended to the blockchain, the transfer is completed which provides confirmation of delivery to the holder. Confirmation that the electronic negotiable Bill of Lading has ceased to have any effect or validity is realized by transferring ownership of the smart negotiable Bill of Lading back to the carrier. If ownership of the smart negotiable Bill of Lading is transferred to the carrier in order to claim title to the goods, the carrier becomes the holder and thus has exclusive control over transfer. As a consequence, no other party can claim title to the goods which results in the smart negotiable Bill of Lading not having any effect or validity any longer as its lifecycle has ended.

In conclusion, the architecture design fulfills the requirement to guarantee uniqueness as it follows all four principles for functional equivalence that constitute an electronic negotiable Bill of Lading.

UC3-req. 3: The architecture should ensure confidentiality of both the electronic negotiable Bill of Lading and identities of businesses involved in the transfer of ownership and should allow only authorized parties access to the electronic negotiable Bill of Lading or identity
To ensure confidentiality of the negotiable Bill of Lading and identities of businesses, both current holder and new holder, only authorized parties should be able to access the document or identity of business involved. First, fulfilment of confidentiality of the negotiable Bill of Lading is evaluated. The negotiable Bill of Lading is encrypted by the carrier before being stored in the off-chain IPFS storage. The decryption key is sent together with transactions to ensure only authorized parties, for example the new holder, can access the negotiable Bill of Lading. A non-authorized party can thus retrieve the document but cannot see the actual data within as the document is encrypted. This fulfils the requirement of confidentiality of the negotiable Bill of Lading as only authorized parties can see the actual data within the document.

Next, fulfillment of confidentiality of the identities of businesses involved in the architecture is evaluated. The use of pseudo-PU identity certificates within the architecture enables business to keep their real-world identity confidential while still being involved the transfer of ownership of the negotiable Bill of Lading. A party that is not authorized to see the real-world identity (PU) of the businesses involved, will only see the pseudo-PU of the current holder in the state of the smart negotiable Bill of Lading or the pseudo-PUs of the current holder and new holder in the 'transfer to new holder' transaction as shown in Table 60 and Table 61. An authorized party can see the PU as a business can choose to reveal the PU to another party by exchanging the PU with this party.

In conclusion, the use of encryption and decryption key management by the document owner (consignor) to ensure confidentiality of the pro forma invoice document and the use of pseudo-PU identity certificates by businesses in the architecture fulfills the requirement.

Table 60: Transaction perspective unauthorized party.

Transaction element	Value
	Unauthorized party
Confirming party 1 ID	04 bb 50 a9 5f 4a 4a … (pseudo-PUcurrentholder)
Confirming party 2 ID	24 cb 33 b3 zf 4a 2b … (pseudo-PUnewholder)

Table 61: Smart negotiable Bill of Lading state perspective unauthorized party.

State element	Value
	Unauthorized party
Current holder ID	24 cb 33 b3 zf 4a 2b … (pseudo-PUholder)

UC3-req. 4: The architecture should be open to businesses, without the use of a centralized title registry and without a central authority in control
Any business that is either the current holder or new holder of the electronic negotiable Bill of Lading should have access to the blockchain network. A governing consortium consisting of a representation of carriers, P&I and a technology company controls the identity and permission management. This ensures non-arbitrary treatment of business that want to use the architecture to exchange trade documents as there is no central authority in control. A smart negotiable Bill of Lading is used to transfer ownership to a new holder which does not require the use of a centralized title registry. Each business holds its own copy of the blockchain ledger which functions as a decentralized title registry. Within the blockchain-based architecture design, a component that could conflict with the requirement of having no central authority in control is the consensus mechanism. The architecture uses PBFT as consensus protocol that can only handle a limited number of ordering nodes that participate. This limitation could create the impression that a central authority is in control of the consensus protocol and thus the world state of the blockchain ledger. To mitigate this potential conflict, the set of ordering nodes is composed of a representation of the governing consortium parties to ensure non-arbitrary treatment. In addition, ordering nodes do not have a copy of the blockchain ledger. Their only task is to order incoming transactions and send blocks of ordered transactions to all other nodes. No further actions are performed on the transactions. This should provide businesses with sufficient trust that no central authority is in control. Thus, the architecture design fulfills the requirement as it provides open access to the architecture, uses a decentralized title registry and is governed by a consortium consisting of different types of parties.

UC3-req. 5: The architecture should be able to handle high volumes of ownership transfers in real-time
To be relevant to carriers and businesses, the architecture should be able to handle hundreds of millions of 'transfer to new holder' transactions per year with low latency. This paragraph therefore evaluates how the designed architecture is scalable compared to a blockchain-based initiative like CargoX that relies on Proof-of-Work as consensus protocol. In CargoX, the use of Proof-of-Work results in a throughput of eight to fifteen transactions. While exact statistics on the volume of negotiable Bill of Ladings lacks, it is assumed that a throughput higher than fifteen transactions per second is needed. In addition, it takes up to sixty minutes before the 'transfer to new holder' transaction can be deemed valid due to the forking issue in Proof-of-Work. As transfers have to take place in real-time, Proof-of-Work is not suitable. In contrast, in the architecture design that uses PBFT there is no delay in appending new blocks of transaction to the blockchain ledger. The only delay in terms of latency that can occur is the sending of messages between ordering nodes which can take up to a few seconds. As a result, PBFT can reach a throughput of tens of thousands of

transactions per second which is sufficient to handle all ‵transfer to new holder‵ transactions without delay. Table 59 provides an overview of the comparison.

The comparison shows that the use of Proof-of-Work as consensus protocol cannot fulfill the requirement. On the contrary, PBFT can fulfill the requirement. In conclusion, the architecture fulfills the requirement to be able to generate high volumes of ownership transfers in real-time as PBFT which enables high transaction throughput and low latency is used.

Table 62: Comparison consensus protocol scalability requirement use case II.

Required		**Consensus protocol**	
		PBFT	*Proof-of-Work (CargoX)*
Transaction throughput	16+ transactions per second	10,000+ transactions per second	8 – 15 transactions per second
Delay	Real-time	Up to a few seconds (network latency)	60 minutes

7.1.4 Sub-conclusion evaluation based on design requirements

The mapping of the blockchain-based architecture on the design requirements shows that the architecture design fulfills all requirements. The next section evaluates the relevance of the use cases how the blockchain-based architecture design implements these use cases using expert opinion by means of interviews.

Table 63: Overview fulfillment of design requirements.

Design requirements use case I: Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration		
Code	*Definition*	*Fulfilled*
UC1-req. 1	The architecture should provide business-to-business exchange of a (pro forma invoice) document to enable end-to-end visibility	**Yes**, businesses are represented by nodes in the blockchain network and have a copy of the blockchain ledger that contains document references of the shipment-related documents in the trade lane. Consignors can invoke a smart shipment contract to create a transaction that adds a pro forma invoice document reference to the blockchain ledger in order to exchange the document.
UC1-req. 2	The architecture should provide business-to-government exchange to enable customs authorities to piggyback an exchanged document for manual cross-validation of an import declaration	**Yes**, a customs authority is represented by a node in the blockchain network and has a copy of the blockchain ledger that contains document references of the shipment-related documents in the trade lane. The customs authority can use the hash pointer on the blockchain ledger to retrieve the document from the consignor's off-chain document storage as well as request a decryption key for the document.
UC1-req. 3	The architecture should ensure confidentiality of both the exchanged document and identities of businesses involved in the exchange transaction and should allow only authorized parties access to the document or identity to overcome legality and liability issues	**Yes**, the document is encrypted before being stored in the off-chain document storage and the hash pointer being generated. Only authorized parties that receive a decryption key can access the document. Businesses use PU pseudonym identity certificates for transactions to hide their real-world identity. Only authorized parties can see the real-world identity. This overcomes legality and liability issues as confidentiality is ensured.
UC1-req. 4	The architecture should ensure integrity of the exchanged document	**Yes**, the document is hashed and the resulting hash pointer is stored on the immutable blockchain ledger making alterations to the document impossible, thus ensuring integrity of the document.
UC1-req. 5	The architecture should be open to businesses, without the use of a centralized document storage and without a central authority in control	**Yes**, a consortium of different types of parties governs identity and permission management to ensure non-arbitrary treatment of businesses that want to use the architecture thus ensuring openness without a central authority in control. The consortium parties also act as ordering nodes in the consensus mechanism to overcome the potential impression that a central authority is in control. Documents are stored in the consignor's off-chain document storage (decentralized).
UC1-req. 6	The architecture should be able to handle high volumes of document exchanges in real-time	**Yes**, only a limited number of nodes invoke contracts via the execute – order – validate transaction process which improves scalability. The architecture uses PBFT as consensus protocol for transaction ordering which can scale sufficiently in terms of high transaction throughput (10,000+/sec) and low latency in contrast to the Proof-of-Work consensus protocol which cannot scale sufficiently with only throughput of eight to fifteen transactions per second and latency up to sixty minutes.

Design requirements use case II: Automated generation of an import declaration by aggregating already exchanged pro forma invoice data		
Code	*Definition*	*Fulfilled*
UC2-req. 1	The architecture should enable a declarant to automatically generate an import declaration by retrieving already business-to-business exchanged pro forma invoice data and calculate customs value, and allow business-to-government exchange of the generated import declaration	**Yes***, a declarant is represented by a node in the blockchain network that has a copy of the blockchain ledger. The separate smart shipment contract state contains individual data elements of the pro forma invoice that can be queried by other contracts. To generate an import declaration, the declarant can invoke the smart import declaration that automatically retrieves individual data elements by retrieving the state of the smart shipping contract. Based on retrieved data, the customs value is calculated automatically. The result is stored in the state of the smart import declaration via a transaction and stored on the blockchain ledger which enables the customs authority to retrieve the customs value. ** Note: the use case relies on assumption that use case I is implemented. In addition, as current smart contract technology requires data to be stored on-chain, this could conflict with UC1-req. 5 which requires absence of a centralized storage. Required data might therefore not be available.*
UC2-req. 2	The architecture should guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value	**Yes**, customs authorities provide specification of the smart import declaration for development and perform audits on the code to guarantee correct implementation and thus correct retrieval and calculation. Also, smart contract code is deployed on the blockchain ledger which makes the code immutable. This proofs integrity as each time the contract is invoked, the same output is generated. In addition, customs authorities confirm the write set of the transaction to guarantee correctness.
UC2-req. 3	The architecture should be able to generate high volumes of import declarations in real-time	**Yes**, only a limited number of nodes invoke contracts via the execute – order – validate transaction process which improves scalability. The architecture uses PBFT as consensus protocol for transaction ordering which can scale sufficiently in terms of high transaction throughput (10,000+/sec) and low latency in contrast to the Proof-of-Work consensus protocol which cannot scale sufficiently with only throughput of eight to fifteen transactions per second and latency up to sixty minutes.

Design requirements use case III: Transfer title to the goods using an electronic negotiable Bill of Lading		
Code	*Definition*	*Fulfilled*
UC3-req. 1	The architecture should provide business-to-business exchange to transfer ownership of the electronic negotiable bill of lading to a new holder	**Yes**, carriers and businesses involved in the transfer of ownership are represented by nodes in the blockchain network and have a copy of the blockchain ledger that contains a hash pointer to the electronic negotiable Bill of Lading and the PU pseudonym identity certificate of the current holder. The current holder can invoke the smart negotiable Bill of Lading 'transfer to new holder' function to create a transaction that transfers ownership to a new holder that adds the PU pseudonym identity certificate of the new holder in the state of the smart negotiable Bill of Lading.
UC3-req. 2	The architecture should guarantee uniqueness of the electronic negotiable Bill of Lading following the principles for functional equivalence	**Yes**, exclusive control over issuance is realized via a role 'carrier' attribute in the PU identity certificate that are only issued to carriers. Exclusive control over transfer is realized by storing the pseudo-PU of the current holder in the state of the smart negotiable Bill of Lading and use a confirmation policy that requires a 'transfer to new holder' transaction to be signed using the same pseudo-PU. Integrity is retained by storing the hash pointer to the negotiable Bill of Lading on the blockchain ledger as part of the write set of a transaction that is immutable once stored in a block and appended to the ledger. Ownership can be demonstrated by creating a valid transaction using the PR that belongs to the pseudo-PU of the current holder stored in the state of the smart negotiable Bill of Lading. Adding the 'transfer to new holder' transaction to a block that is appended to the blockchain confirms delivery to the holder. Transferring ownership of the smart negotiable Bill of Lading to the carrier confirms that effect or validity has ceased.
UC3-req. 3	The architecture should ensure confidentiality of both the electronic negotiable Bill of Lading and identities of businesses involved in the transfer of ownership and should allow only authorized parties access to the electronic negotiable Bill of Lading or identity	**Yes**, the electronic negotiable Bill of Lading is encrypted by the carrier before being stored in the off-chain IPFS storage. Businesses can use PU pseudonym identify certificates to mask their real-world identity. This ensures confidentiality of the negotiable Bill of Lading and the business identity as only authorized parties can access the document or real-world identity. Decryption keys are encrypted with the PU pseudonym identity certificate of the new holder and added to 'transfer to new holder' transactions to enable decryption key distribution.
UC3-req. 4	The architecture should be open to businesses, without the use of a centralized title registry and without a central authority in control	**Yes****, a consortium of different types of parties governs identity and permission management to ensure non-arbitrary treatment of businesses that want to use the architecture thus ensuring openness without a central authority in control. The consortium parties also act as ordering nodes in the consensus mechanism to overcome the potential impression that a central authority is in control. A smart negotiable Bill of Lading is used to transfer ownership to a new holder which does not require the use of a centralized title registry. *** Note: the assumption was made that there is no need for an additional legal agreement to be signed by businesses in order to access and use the architecture.*
UC3-req. 5	The architecture should be able to handle high volumes of ownership transfers of electronic negotiable Bill of Ladings in real-time	**Yes**, only a limited number of nodes invoke contracts via the execute – order – validate transaction process which improves scalability. the architecture uses PBFT as consensus protocol which can scale sufficiently in terms of high transaction throughput (10,000+/sec) and low latency in contrast to the Proof-of-Work consensus mechanism which cannot scale sufficiently with only throughput of eight to fifteen transactions per second and latency up to sixty minutes. This allows the architecture to handle hundreds of millions of ownership transfers per year which outperforms current initiatives.

7.2 Architecture evaluation using expert opinion

This section presents the evaluation of the relevance of the use cases and how the blockchain-based architecture design supports the use cases to assess its effectiveness and technical correctness using expert opinion by means of interviews. The relevance of use case I is not evaluated as this use case can be seen as a base case for the data pipeline concept and has therefore been extensively evaluated with experts from both businesses and customs authorities in previous research part of the EU CORE project. Therefore, evaluation of the relevance of use case I does not provide added value to the research. First, sub section 7.2.1 presents the interview strategy this research takes to determine the purpose of the interviews and to select interviewees. Next, sub section 7.2.2 evaluates the relevance of use case II. Third, sub section 7.2.3 evaluates the relevance of use case III. Last, sub section 7.2.4 evaluates the technical correctness of the blockchain-based architecture design.

7.2.1 Interview strategy

To evaluate the relevance of the use cases and how the blockchain-based architecture design supports the use cases to assess its effectiveness and technical correctness, the use cases and blockchain-based architecture design were discussed with several experts in the field of exchanging trade data using interviews. According to Johannesson and Perjons (2014), interviews are: 'effective instruments for gathering stakeholder opinions and perceptions about the use and value of an artifact' (Johannesson & Perjons, 2014, p. 144). A criterium for the selection of interviewees was that they are involved in the development of blockchain-based systems to exchange trade data. Four interviewees that match the criterium were selected. Three out of four interviewees are involved in different roles in the development of the Global Trade Digitization (GTD) platform. As introduced in chapter , this platform is one of the most prominent initiatives that aims at implementing blockchain technology to exchange trade data. The GTD platform is also the result of previous research on the data pipeline concept in the EU CORE project, which makes it closely related to this research.

Four interviews were held. Table 64 provides an overview of the interviews and interviewees. The interviews were held either face-to-face, via e-mail or via teleconference. Minutes of the interviews can be found in appendix I. The use case descriptions and blockchain-based architecture design were sent to the interviewees beforehand to allow the interviewees to prepare for the interview. Each interview has a different goal. The goal of the first interview with the Dutch Tax and Customs Administration is evaluation of the relevance of use case II: 'Automated generation of an import declaration using already exchanged pro forma invoice data' as the focus is on customs import procedures. The goal of the second interview is evaluation of use case III: 'Transfer title to the goods using an electronic negotiable Bill of Lading'. As container carriers are most closely involved in the process of issuing a negotiable Bill of Lading and releasing goods to businesses claiming title to the goods with a negotiable Bill of Lading, a representative from Maersk Line, one of the world's largest container shipping companies and partner in the GTD platform, was interviewed to evaluate the relevance of use case III. The third interview focusses on how the blockchain-based architecture design supports the use cases to evaluate the effectiveness of the architecture from a technical perspective. This includes evaluation of the architecture components as demonstrated in chapter 6. Two technical experts from technology company International Business Machine Corporation (IBM) were interviewed together to evaluate the design. As the IBM representatives are closely involved in the development of the GTD platform and implementation of use cases, some notes on the relevance of the use cases were also made during the interview. The next three sub sections discuss the outcomes of the expert interviews.

Table 64: Overview of interviews.

#	Goal	Date	Type	Interviewee[21]	
1	Evaluation use case II	19-07	E-mail	1	Representative Dutch Tax and Customs Administration
2	Evaluation use case III	25-06	Teleconference	2	Representative Maersk Line, involved in development GTD platform
3	Evaluation architecture design	03-07	Face-to-Face	3	Representative IBM, involved in development GTD platform
3	Evaluation architecture design	03-07	Face-to-Face	4	Representative IBM, part of BeNeLux Blockchain Practice

7.2.2 Evaluation relevance use case II

In this sub section, the evaluation of the relevance of II: 'Automated generation of an import declaration using already exchanged pro forma invoice data' by a representative from the Dutch Tax and Customs Administration (customs authority) is discussed. As the use case is explorative of nature, evaluation is on a high-level abstraction. Automated generation of an import declaration by aggregating already exchanged trade data (in this case from the pro forma invoice) is highly relevant for customs authorities. As the representative states: 'it can fundamentally change the way in which customs authorities function'. Customs procedures can be simplified, resulting in less bureaucracy and thus reduce time delay and costs. The move towards this use case is realistic. However, from a legal perspective the use case is still far away as legislation is not ready.

7.2.3 Evaluation relevance use case III

This sub section discusses the evaluation by a Maersk Line representative on the relevance of use case III: 'Transfer title to the goods using an electronic negotiable Bill of Lading. The current issues of paper-based negotiable Bill of Ladings regarding fraud, time delay and high costs are indeed very present. The Maersk Line representative referred to the current process of paper-based negotiable Bill of Ladings as being 'a nightmare'. The negotiable Bill of Lading is used less and less because of the difficult process, while the ability to transfer title to the goods is very useful. The move towards an electronic negotiable Bill of Lading to electronically transfer title to the goods is therefore very relevant. Currently, blockchain technology seems indeed the way to go.

Guarantee of uniqueness and legal agreements

In the use case, the guarantee of uniqueness of an electronic negotiable Bill of Lading and the required additional legal agreements are seen as requirement for and challenge of the use case. The challenge stems from the idea that businesses might be reluctant to adopt the electronic negotiable Bill of Lading if additional legal agreements are required. Current maritime law does not support electronic negotiable Bills of Lading. This requires an additional legal agreement to be endorsed by the International Group of Protection & Indemnity Clubs (P&I) and accepted by businesses that obligates a business to treat an electronic negotiable Bill of Lading as the functional equivalence of a negotiable Bill of Lading. The interviewee states that indeed such agreement (similar to the Bolero Rulebook) is currently needed. Maersk Line is currently working together with P&I to explore the legal agreements that would be required for a blockchain-based electronic negotiable Bill of Lading (in GTD), but the interviewee does not know the details of these agreements.

This research does not dive into the specifics of such legal agreements as the scope tends more towards exploring the technical aspects of guarantee of uniqueness and scalability of a blockchain-

[21] Names of interviewees are known by the researcher.

based system for an electronic negotiable Bill of Lading. Therefore, the use of the principles for functional equivalence, specified in the Rotterdam Rules and UNCITRAL Model Law, as requirement to guarantee uniqueness of the electronic negotiable Bill of Lading was proposed to the interviewee. The interviewee agreed that the use of these principles to base an electronic negotiable Bill of Lading, in this case a smart negotiable Bill of Lading, on would be a good approach. However, a note should be made that the interviewee is not an expert on maritime law and answers are based upon information provided by the legal department of Maersk Line. Further research into current and future maritime law would be required.

Use of a smart negotiable Bill of Lading
In the interview, two options for an electronic negotiable Bill of Lading without a centralized title registry were discussed. One option is to store the centralized title registry of existing initiatives on the blockchain ledger to create a form of decentralized title registry by distributing the title registry to improve transparency of the registry. A second option is the use of smart contract to represent an electronic negotiable Bill of Lading and transfer ownership of the contract to a new holder via transactions. The use case opts for the smart contract option.

The use of a smart contract (smart negotiable Bill of Lading) to represent an electronic negotiable Bill of Lading led to a discussion on the buy-in challenge of trade lane businesses. The Maersk Line representative states that Maersk Line is not yet convinced of using smart contracts (like in the CargoX initiative or this architecture design) to transfer ownership of an electronic negotiable Bill of Lading to a new holder. There is uncertainty if the use of smart contracts gets sufficient buy-in from trade lane businesses as they need to put in more trust in the system. Maersk opts for a hybrid approach: provide both the option of using a distributed title registry or smart contract solution. The distributed title registry could be as simple as exchanging a document that contains the registry. The advantage over current centralized title registries would be that the updates to the registry are transparent.

The previous two sub sections discussed the evaluation of use cases II and III. The next sub section evaluates the effectiveness and technical correctness of the blockchain-based architecture.

7.2.4 Evaluation effectiveness and technical correctness blockchain-based architecture design

In this section the evaluation of the effectiveness and technical correctness of the blockchain-based architecture is presented. The technical correctness was evaluated with two IBM representatives. Due to time constraint, not all architecture component were evaluated explicitly. Instead, the demonstration of implementation of use cases was mainly used to evaluate the effectiveness and correctness of the blockchain-based architecture design. If demonstration is successful, this shows the effectiveness and proper functioning of the blockchain-based architecture. The blockchain-based architecture design and demonstration were sent to the IBM interviewees beforehand to allow for technical feedback during the interview.

In the interview, general validity of the blockchain-based architecture was confirmed. For example, the concept of the execute – order – validate transaction process as used in the blockchain-based architecture's consensus mechanism component is based on the Hyperledger platform on which the GTD platform is built. Also, the use of PBFT was validated as Hyperledger proposes this as a pluggable consensus protocol option. Therefore, the interview focused on elements of the blockchain-based architecture design that are more specific toward the use cases to evaluate the effectiveness of the architecture.

Encryption
An important requirement for all three use cases is confidentiality of the document or individual data elements. Therefore, the architecture uses encryption of documents and individual data elements to

ensure confidentiality. Decryption keys are distributed by the document or individual data element owner to allow the owner to control which parties are authorized to access the document or individual data elements. The IBM interviewees agree that the use of encryption is a good solution to ensure confidentiality.

Blockchain ledger per trade lane
For use cases I and II, the blockchain-based architecture proposes the use of a separate blockchain ledger for each trade lane to store documents or trade data of shipments within that trade lane. The IBM interviewees see a potential scalability issue. Current blockchain-based architectures that use multiple blockchain ledgers within a blockchain network, like the GTD platform, have limited scalability of 10,000 blockchain ledgers. A separate blockchain ledger for each trade lane could quickly reach this limit. Therefore, the GTD platform uses a blockchain ledger per carrier instead of trade lane to reduce the number of ledgers.

Public key pseudonym identity certificates
The requirement of confidentiality extents to hiding of the identity of a business involved in a transaction. The blockchain-based architecture proposes the use of public key (PU) pseudonym identity certificates for businesses issued by a Pseudonym Certificate Authority for use within shipments on a transactional-level. A short discussion on these certificates was held. The IBM interviewees agree that this proposal enables masking of identities and thus provides confidentiality.

Electronic negotiable Bill of Lading as smart negotiable Bill of Lading
During the interview, additional attention was paid to use case III: ‘Transfer title to the goods using an electronic Negotiable Bill of Lading’. IBM and Maersk Line work together in the GTD platform, which makes this use case highly relevant from the perspective of a carrier. Even further, the use case can enable future use cases in context of trade financing as the negotiable Bill of Lading is often used as collateral when placed under a Letter of Credit arrangement. In addition, the requirement to guarantee uniqueness makes the electronic negotiable Bill of Lading a special type of document. The functioning of the smart negotiable Bill of Lading was demonstrated in detail to the interviewees using the demonstration in chapter 6 which also covered the principles of functional equivalence to demonstrate the guarantee of uniqueness. The ‘transfer ownership to a new holder’ functionality was used to demonstrate typical user activities. The four principles of functional equivalence were discussed as part of the demonstration. The IBM interviewees acknowledge that all four principles are correctly implemented. The reasoning is similar to the evaluation based on design requirements in section 7.1.3.

The IBM representatives found the design of the smart contract to represent the electronic negotiable Bill of Lading very interesting. The smart negotiable Bill of Lading contains all required functionality as well as guarantees uniqueness as the principles for functional equivalence are fulfilled. One of the IBM representatives expressed confidence that the current design could actually be implemented in a real-world application. This sub section discussed the evaluation of implementation of the use cases in the blockchain-based architecture design to determine its effectiveness. The next section presents a set of design improvements to the blockchain-based architecture design that were derived based on the evaluation.

7.3 Design improvements

In the previous sub sections, the outcomes of the expert interviews to evaluate the relevance of the use cases and effectiveness and technical correctness of the blockchain-based architecture design were discussed. In this section, two design improvements are derived based on this evaluation. First, the blockchain-based architecture design proposes the use of a separate blockchain ledger per trade lane (for use cases I and II) under the assumption that these lanes are relatively fixed. Evaluation of the blockchain-based architecture by the IBM representatives raised a potential scalability issue with the use of separate blockchain ledgers per trade lane. The current limit of 10,000 ledgers within a blockchain network could quickly be reached, causing a scalability issue. Based on their own experience with development of the GTD platform, they proposed the use of a blockchain ledger per carrier instead of trade lane. Use case III already uses the idea of a blockchain ledger per carrier. The underlying reason is that the most important business in case of an electronic negotiable Bill of Lading is the carrier. Also, a negotiable Bill of Lading can be transferred to businesses beyond the trade lane.

A downside of using a separate blockchain ledger per carrier in the architecture that supports use case I and II is that it could potentially create a conflict with the confidentiality requirement (see UC1-req. 4). Documents or individual data elements and (pseudonym) identities of businesses on many shipments in many trade lanes in which the carrier is involved would become available to a large group of businesses. Even though encryption and pseudonym identities are used to provide confidentiality, business might be reluctant to business-to-business exchange documents or individual data elements in this setting. A business could perform extensive data analysis on the blockchain ledger to identity patterns of shipments that can be used for competitive gain. For example, it might be possible to derive the real-world identities of businesses based on these patterns. This conflicts with requirement UC1-req. 4 and can reduce the willingness of businesses to exchange documents or individual data elements (Van Engelenburg & Janssen, 2018).

Also, especially small-medium enterprises (SMEs) might not be interested in maintaining a large blockchain ledger with an enormous volume of documents or individual data elements of shipments they are not involved in. A design improvement that makes a trade-off between the limitation of a blockchain ledger per trade lane and potential lack of confidentiality per carrier is therefore proposed. In use cases I and II, besides the carrier the Freight Forwarder also plays a prominent role. The Freight Forwarder often takes care of preparation and processing of documentation pertaining to a shipment. For example, for a shipment under DDP Incoterm, the Freight Forwarder can act as declarant of the import declaration on behalf of the consignor. Therefore, the proposed design improvement is to use a separate blockchain ledger based on the carrier and Freight Forwarder(s) involved in a trade lane. Figure 33 depicts the improvement. For each combination of carrier and Freight Forwarder(s) a separate ledger is used. Documents or individual data elements of shipments that involve the same combination of carrier and Freight Forwarders are stored on the same blockchain ledger. The improvement overcomes the limitation in terms of number of blockchain ledgers within the blockchain network, while providing better confidentiality of shipment documents or individual data elements and identities of businesses compared to a solution with a separate blockchain ledger per carrier.

A second design improvement refers to the notion made by the Maersk Line representative that Maersk Line is not yet convinced of the use of a smart contract to represent an electronic negotiable Bill of Lading as it requires buy-in from businesses. Maersk line therefore opts for a hybrid approach that includes both a smart contract and a distributed title registry file. In essence, this distributed title registry is the exchange of a document (e.g., an Excel file specifying who is the current holder of an electronic negotiable Bill of Lading). An external title registry system keeps track of transfer of ownership transactions. After a new transaction is added, the distributed title registry document is stored on the blockchain ledger via a transaction similar to the exchange of a pro forma invoice document in use case I. In this case, the central authority in control of the title registry system is responsible for exchanging the title registry after each transaction. Storing the title registry on the blockchain ledger improves transparency as the central authority cannot alter previous transactions. Businesses can easily view the title registry by accessing their copy of the blockchain ledger. Different from the architecture design of use case III, a blockchain ledger per title registry system would be used instead of per carrier. All elements of the electronic negotiable Bill of Lading regarding functional equivalence and transfer of ownership to a new holder would be kept at the title registry system, taking away the need for businesses to buy-in to the blockchain-based architecture to transfer ownership to a new holder using a smart negotiable Bill of Lading. The architecture is therefore extended with the possibility to exchange a title registry, similar to the exchange of a pro forma invoice in use case I.

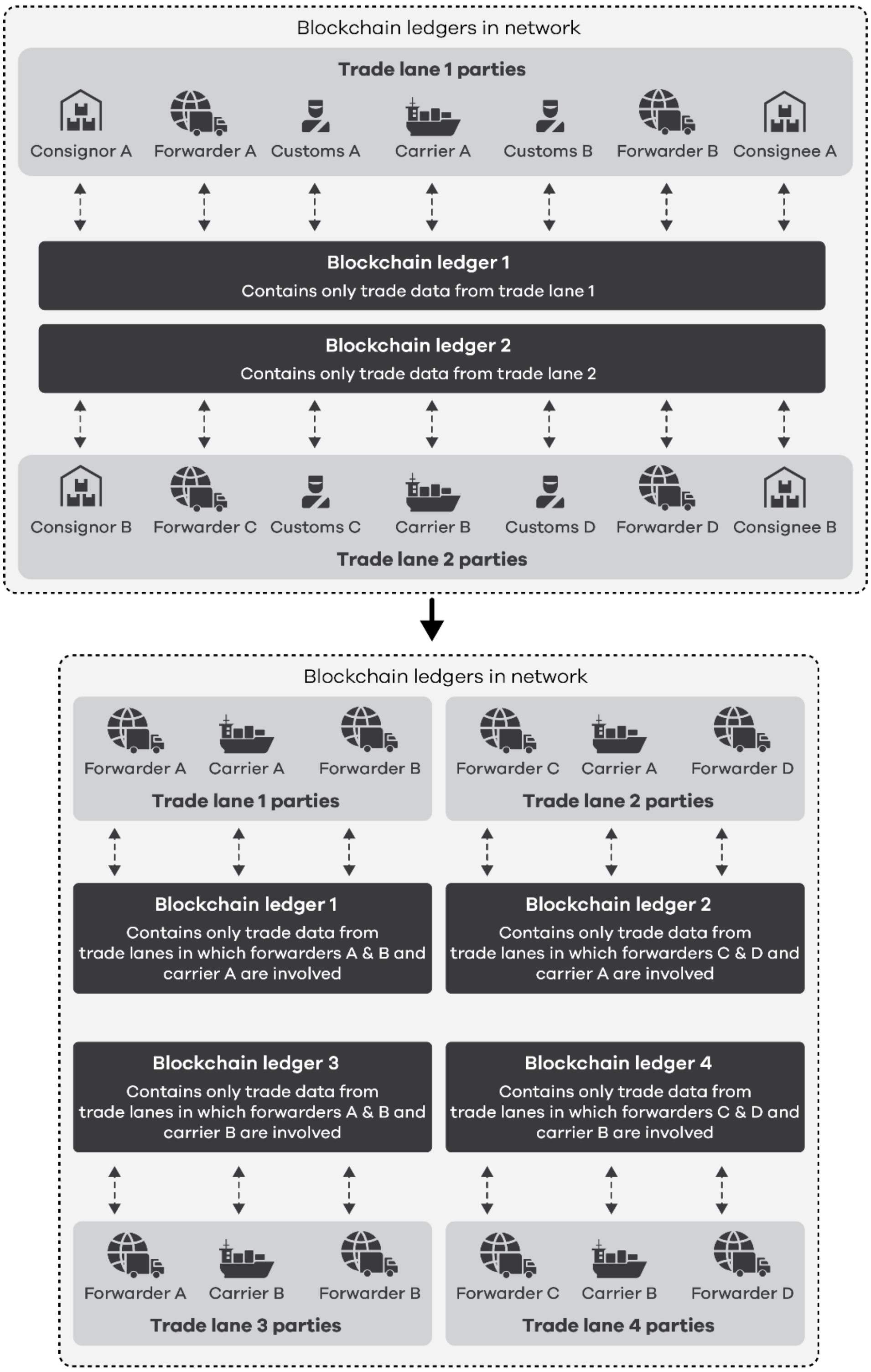

Blockchain ledgers in network
Trade lane 1 parties
Consignor A
Forwarder A
Customs A
Carrier A
Customs B
Forwarder B
Consignee A
Blockchain ledger 1
Contains only trade data from trade lane 1
Blockchain ledger 2
Contains only trade data from trade lane 2
Consignor B
Forwarder C
Customs C
Carrier B
Customs D
Forwarder D
Consignee B
Trade lane 2 parties
Blockchain ledgers in network
Forwarder A
Carrier A
Forwarder B
Trade lane 1 parties
Forwarder C
Carrier A
Forwarder D
Trade lane 2 parties
Blockchain ledger 1
Contains only trade data from trade lanes in which forwarders A & B and carrier A are involved
Blockchain ledger 2
Contains only trade data from trade lanes in which forwarders C & D and carrier A are involved
Blockchain ledger 3
Contains only trade data from trade lanes in which forwarders A & B and carrier B are involved
Blockchain ledger 4
Contains only trade data from trade lanes in which forwarders C & D and carrier B are involved
Forwarder A
Carrier B
Forwarder B
Trade lane 3 parties
Forwarder C
Carrier B
Forwarder D
Trade lane 4 parties

Figure 33: Current multiple blockchain ledger sub component (top) and design improvement blockchain ledger sub component (bottom) of the blockchain-based architecture of use cases I and II.

This section discussed two design improvements for the blockchain-based architecture based on the outcomes of the expert interviews that evaluated the relevance of the use cases and implementation of the use cases in the blockchain-based architecture. The next section concludes this chapter.

7.4 Conclusion chapter 7

Chapter 7 mapped the blockchain-based architecture design on the design requirements to evaluate its fit with the requirements and used expert opinion (gathered via interviews) to evaluate the relevance of the use cases and evaluate effectiveness and technical correctness of the blockchain-based architecture design. This answers sub question 7: *'To what extent, for evaluation purposes, does the blockchain-based architecture design fit with the design requirements and problem domain of the critical trade document use cases?*'. The evaluation of the blockchain based architecture design using the design requirements for each use case showed that the architecture fulfills all requirements. Table 63 provides an overview of how each requirement is fulfilled. Two important notes are made for UC2-req. 1 and UC3-req. 4. The argument that requirement UC2-req. 1 is fulfilled relies on the assumption that use case I is already implemented and the required on-chain storage (which is some form of centralized storage) does not conflict with UC1-req. 5 which requires that no centralized storage is used for the exchange of documents. Fulfillment of UC3-req. 4 relies on the assumption that there is no need for an additional legal agreement (like the Bolero Rulebook) to be signed by businesses in order to access and use the blockchain-based architecture. Evaluation of the relevance of use cases II and III by expert opinion showed that the automated generation of an import declaration using already exchanged pro forma invoice data and transfer title to the goods using an electronic negotiable Bill of Lading are very relevant to reduce bureaucracy, time delay and costs. Evaluation of the blockchain-based architecture confirmed effectiveness and technical correctness of the design. Based on the outcomes of the expert interviews, two design improvements were considered. The multiple blockchain ledger structure as used for use cases I and II was changed to be based on carrier and Freight Forwarders involved in a shipment instead of all trade lane parties involved to avoid the scalability issue of 10,000 ledgers. Also, the architecture that supports use case III was extended to enable the exchange of a title registry similar to the exchange of a pro forma invoice document in use case I. The next chapter presents the final step of his research and derives conclusions in order to answer the main research question.

8. Conclusions

The final part of the research approach is to derive conclusions in order to answer the main research question. In this chapter, conclusions are derived to answer the main research question: '*Which design of a blockchain-based architecture can be developed to support exchange of trade documents in trade lanes that takes into account the requirements of openness, security and scalability?*'. First section 8.1 presents conclusions for each of the sub questions. Next, based on these conclusions section 8.2 answers the main research question. Third, section 8.3 discusses the scientific and societal relevance. Finally, section 8.5 reflects on this research by discussing the research process, identifies limitations and proposes potential future research to address these limitations. Also, a link between the thesis and CoSEM programme is made.

8.1 Conclusions sub questions

In this thesis, six sub questions following a Design Science Research (DSR) approach were used to accumulate knowledge in order to answer the main research question. In this section, conclusions for each of the sub questions are presented.

8.1.1 Answering sub question 1

A literature analysis was conducted to answer sub question 1: *'What do the three identified critical trade document use cases that can be enabled by blockchain technology look like?'*. The three use cases that were developed are: 1) exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration, 2) automated generation of an import declaration by aggregating already exchanged pro forma invoice data and 3) transfer title to the goods using an electronic negotiable Bill of Lading. Use case I finds that blockchain technology can be used for the business-to-business exchange of a pro forma invoice by storing the invoice on the blockchain ledger via transactions. It also enables business-to-government exchanges as a customs authority can use its own copy of the blockchain ledger to pull the pro forma invoice and piggyback the document for manual cross-validation. Use case II finds that blockchain technology can be used to automatically generate an import declaration by aggregating exchanged pro forma invoice data as enabled by use case I as use case II is an extension of use case I. Pro forma invoice data is stored on the blockchain ledger. Using smart contracts, data can be retrieved to automatically generate an import declaration. This overcomes the need for manual-cross validation. Use case III finds that blockchain technology can be used to transfer title to the goods of an electronic negotiable Bill of Lading via transactions that are stored on the blockchain ledger. The immutability of the blockchain ledger can guarantee uniqueness of the electronic negotiable Bill of Lading as it overcomes the double spend problem.

8.1.2 Answering sub question 2

To answer sub question 2:'*What are the design requirements for a blockchain-based architecture design that supports the critical trade document use cases?*', the findings of sub question 1 and a literature review on the data pipeline concept regarding data security and governance as well as literature on the electronic negotiable Bill of Lading was used to define a set of design requirements for the blockchain-based architecture for each critical trade document use case. For use case I, exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration, a set of six design requirements was defined as shown in Table 65. For use case II, automated generation of an import declaration by aggregating already exchanged pro forma invoice data, a set of three design requirements was defined as shown in Table 66. For use case III, transfer title to the goods using an electronic negotiable Bill of Lading, a set of five design requirements was defined as shown in Table 67.

Table 65: Overview design requirements use case I.

Design requirements use case I	
Code	*Definition*
UC1-req. 1	The architecture should provide business-to-business exchange of (a pro forma invoice) document to enable end-to-end visibility
UC1-req. 2	The architecture should provide business-to-government exchange to enable customs authorities to piggyback an exchanged document for manual cross-validation of an import declaration
UC1-req. 3	The architecture should ensure confidentiality of both the exchanged document and identities of businesses involved in the exchange transaction and should allow only authorized parties access to the document or identity to overcome legality and liability issues
UC1-req. 4	The architecture should ensure integrity of the exchanged document
UC1-req. 5	The architecture should be open to businesses, without the use of a centralized document storage and without a central authority in control
UC1-req. 6	The architecture should be able to handle high volumes of document exchanges in real-time

Table 66: Overview design requirements use case II.

Design requirements use case II	
Code	*Definition*
UC2-req. 1	The architecture should enable a declarant to automatically generate an import declaration by retrieving already business-to-business exchanged pro forma invoice data and calculate customs value, and allow business-to-government exchange of the generated import declaration
UC2-req. 2	The architecture should guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value
UC2-req. 3	The architecture should be able to generate high volumes of import declarations in real-time

Table 67: Overview design requirements use case III.

Design requirements use case III	
Code	*Definition*
UC3-req. 1	The architecture should provide business-to-business exchange to transfer ownership of the electronic negotiable bill of lading to a new holder
UC3-req. 2	The architecture should guarantee uniqueness of the electronic negotiable Bill of Lading following the principles for functional equivalence
UC3-req. 3	The architecture should ensure confidentiality of both the electronic negotiable Bill of Lading and identities of businesses involved in the transfer of ownership and should allow only authorized parties access to the electronic negotiable Bill of Lading or identity
UC3-req. 4	The architecture should be open to businesses, without the use of a centralized title registry and without a central authority in control
UC3-req. 5	The architecture should be able to handle high volumes of ownership transfers of electronic negotiable Bill of Ladings in real-time

8.1.3 Answering sub question 3

A literature analysis on blockchain technology architectures to identify the core components of such architecture was conducted in order to answer sub question 3: '*What are the core blockchain technology architecture components?*'. Four core components for a blockchain architecture were identified: 1) network configuration, 2) data structure and storage, 3) consensus mechanism and 4) application. The network configuration component is used to determine which parties in which role can participate in the peer-to-peer blockchain network by controlling permissions. Two types of network configuration topologies can be distinguished: 1) permissionless, in which there is no control over permissions for read or write permissions and participation in consensus mechanism and 2) permissioned in which read or write permissions and permission to participate in the consensus mechanism are managed via additional identity and permission management components.

The data structure and storage component determines how data is stored, either on-chain on the blockchain ledger or off-chain in a separate data storage with a reference to the data stored on-chain. It also determines whether al data is stored on a single or multiple blockchain ledgers. In addition, it also determines the way in which transactions are stored and update the state of the blockchain ledger. To that extent, two types of transaction models are identified: 1) UTXO which is stateless and 2) account-based which explicitly stores a state in accounts. Data security can be realized via either data encryption or Zero-Knowledge proof.

The third component identified component is the consensus mechanism that is used to determine the way in which parties reach agreement on the order of transactions in the blockchain ledger and thus state of the blockchain ledger. Two types of transaction processes can be distinguished: 1) order – executes that requires smart contracts to be deterministic which reduces flexibility as all nodes have to invoke the same smart contract for each transaction to confirm validity and thus reach consensus and 2) execute – order – validate which allows smart contracts to be non-deterministic as not all nodes have to invoke the same smart contract for each transaction. This also improves scalability. In addition, several consensus protocols are identified to be used in the consensus mechanism, of which Proof-of-Work and Practical Byzantine Fault Tolerance (PBFT) are currently the most prominent and well-developed. An important trade-off between Proof-of-Work and PBFT is that Proof-of-Work has high scalability in terms of nodes, but limited scalability in terms of transactions (maximum throughput of eight to fifteen transactions per second) due to a built-in delay of on average ten minutes in case of Bitcoin for generation of new blocks using a cryptographic mining challenge. This to avoid double spending by forking of the blockchain ledger. PBFT has high scalability in terms of transaction throughput as it does not use mining but voting for generation of blocks. However, as it is subject to Sybil attacks, it is only applicable in a permissioned network configuration where the nodes that are allowed to participate in the consensus mechanism are regulated. In addition, PBFT is subject to low scalability in terms of nodes participating.

The fourth core component is the application. This component implements the business logic and provides the functionality used to alter the state of the blockchain ledger by creating output transactions based on input provided by another transaction that is created by a node that wants to add a new transaction to the blockchain ledger. The contract can be used to represent the state of a physical asset (e.g., via a smart contract) or a native digital asset (e.g., cryptocurrency token).

8.1.4 Answering sub question 4

Using the outcomes of sub questions two and three, the actual architecture design was developed for each use case in order to answer sub question 4: '*What does the blockchain-based architecture design look like for each of the critical trade document use cases?*'. For support of use case I, the architecture design uses a permissioned network configuration with identity management based on PKI managed by a consortium with support for external and internal PU identify certificates to be open to businesses and internal PU identity pseudonym certificates for confidentially purposes.

Permission management on network access permission, blockchain ledger read and write permission and consensus participation permission ensure only authorized parties can perform actions within the architecture. Within the data structure and storage component, a multiple blockchain ledger structure (one ledger per trade lane) is used. An account-based transaction model stores transactions in a blockchain and the state of the contract, including a pro forma invoice document reference (hash pointer), in the world state. External off-chain storage of the data owner (consignor) is used to store the actual pro forma invoice document. Documents are encrypted and external ABAC managed by the data owner enables decryption key distribution. The architecture uses a smart contract to implement the business logic. The smart shipment contract stores the pro forma invoice document reference and PU and pseudo-PU identity certificates of the trade lane parties in its state. An 'append pro forma invoice' function to create a transaction enables a consignor to exchange a pro forma invoice document. The trade lane party identities are used for permission management and decryption key distribution. For the consensus mechanism an execute - order – validate transaction process is used. Transaction ordering uses PBFT as consensus protocol to enable high transaction throughput with consortium nodes acting as ordering nodes.

The architecture design to support use case II is an extension of the architecture of use case I and thus uses the same permissioned network configuration. Within the data structure and storage component, a multiple blockchain ledger structure (one for each trade lane) is used. An account-based transaction model stores transactions in a blockchain and the contract state including individual pro forma invoice data elements and customs value data element in the world state. No off-chain storage is used. Individual data elements are encrypted and external ABAC managed by the data owner (consignor) enables decryption key distribution. The architecture uses two smart contracts to implement the business logic. A smart shipment contract stores pro forma invoice data elements in its state. A declarant can use the 'lodge import declaration' function of a separate smart import declaration to retrieve these data elements and calculate the customs value based on the retrieved data elements. The same consensus mechanism as use case I is used.

To support of use case III, the architecture uses a permissioned network configuration topology with identity management based on PKI with support for external and internal PU identify certificates to be open to businesses and internal PU pseudonym identity certificates for confidentially purposes. Permission management on network access permission, blockchain ledger read and write permission and consensus participation permission ensure only correct parties can perform actions within the architecture. Within the data structure and storage component, a multiple blockchain ledger structure (one for each carrier) is used. An account-based transaction model stores transactions in a blockchain and the state of the contract, including a negotiable Bill of Lading reference (hash pointer) and PU pseudonym identity certificate of the current holder, in the world state. Internal off-chain storage is used to store the actual negotiable Bill of Lading document. Documents are encrypted and decryption keys are distributed by the current holder. The architecture uses a smart contract to implement the business logic. The smart negotiable Bill of Lading stores the document reference and PU pseudonym identity certificate of the current holder in its state. An 'transfer to new holder' function enables the current holder to transfer ownership to a new holder. For the consensus mechanism an execute - order – and validate transaction process is used. Transaction ordering uses PBFT to enable high transaction throughput with the consortium parties acting as ordering nodes.

8.1.5 Answering sub question 5

A set of typical user activities, based on the functional requirements, were used to demonstrate the architecture design for each use case in order to answer sub question 5: *'How, for demonstration purposes, can the blockchain-based architecture design be used for the critical trade document use cases?'*. In use case I, to exchange the pro forma invoice the consignor invokes the smart shipment contract's 'append pro forma invoice' function by providing a document reference as input. Correctness of the pro forma invoice is confirmed by the consignee and the 'append pro forma invoice' transaction is sent to the ordering nodes by the consignor. The ordering nodes create a block containing the transaction and broadcast the block to all other nodes. These nodes validate the transaction and update the world state accordingly. A customs authority uses the document reference specified in the state to retrieve the pro forma invoice document. A decryption key is requested from the consignor in order to access the document.

In use case II, to lodge an import declaration the declarant (e.g., Freight Forwarder) invokes the smart import declaration's 'lodge import declaration' function by providing a smart shipment contract ID as input. The smart import declaration retrieves relevant pro forma invoice data (Incoterm, invoice value, transport cost after entry) from the state of the smart shipment contract. It calculates the customs value via an algorithm based on the Incoterm. Correctness of calculation is confirmed by the customs authority and the 'lodge import declaration' transaction is sent to the ordering nodes by the declarant. The ordering nodes create a block containing the transaction and broadcast the block to all other nodes. These nodes validate the transaction and update the world state of the smart import declaration accordingly.

In use case III, to transfer title to the goods by transferring ownership of an electronic negotiable Bill of Lading to a new holder, the current holder invokes the smart negotiable Bill of Lading's 'transfer to new holder' function by providing the pseudo-PU of the new holder as input. The new holder confirms correctness by checking the negotiable Bill of Lading and comparing the read sets of both current and new holder to see if they match. Next, the 'transfer to new holder' transaction is sent to the ordering nodes by the current holder. The ordering nodes create a block containing the transaction and broadcast the block to all other nodes. These nodes validate the transaction and update the world state of the smart negotiable Bill of Lading to specify the new holder as holder accordingly.

8.1.6 Answering sub question 6

To answer sub question 6: *'To what extent, for evaluation purposes, does the blockchain-based architecture design fit with the design requirements and problem domain of the critical trade document use cases?'*, the blockchain-based architecture design of each use case was mapped on the design requirements to evaluate fulfillment. The evaluation using the design requirements for each use case showed that the architecture fulfills all requirements. Fulfillment of the requirements for each use case is shown in Table 68. Next, evaluation of the relevance of use cases II and III by expert opinion showed that the automated generation of an import declaration using already exchanged pro forma invoice data and transfer title to the goods using an electronic negotiable Bill of Lading are very relevant to reduce bureaucracy, time delay and costs which fits perfectly with the problem domain. Evaluation of the blockchain-based architecture confirmed effectiveness and technical correctness of the design.

Table 68: Overview fulfillment of design requirements.

Design requirements use case I: Exchange of a pro forma invoice document that customs authorities can piggyback for manual cross-validation of an import declaration		
Code	*Definition*	*Fulfilled*
UC1-req. 1	The architecture should provide business-to-business exchange of a (pro forma invoice) document to enable end-to-end visibility	**Yes**
UC1-req. 2	The architecture should provide business-to-government exchange to enable customs authorities to piggyback an exchanged document for manual cross-validation of an import declaration	**Yes**
UC1-req. 3	The architecture should ensure confidentiality of both the exchanged document and identities of businesses involved in the exchange transaction and should allow only authorized parties access to the document or identity to overcome legality and liability issues	**Yes**
UC1-req. 4	The architecture should ensure integrity of the exchanged document	**Yes**
UC1-req. 5	The architecture should be open to businesses, without the use of a centralized document storage and without a central authority in control	**Yes**
UC1-req. 6	The architecture should be able to handle high volumes of document exchanges in real-time	**Yes**
Design requirements use case II: Automated generation of an import declaration by aggregating already exchanged pro forma invoice data		
Code	*Definition*	*Fulfilled*
UC2-req. 1	The architecture should enable a declarant to automatically generate an import declaration by retrieving already business-to-business exchanged pro forma invoice data and calculate customs value, and allow business-to-government exchange of the generated import declaration	**Yes***
UC2-req. 2	The architecture should guarantee correct retrieval of pro forma invoice data elements and calculation of the customs value	**Yes**
UC2-req. 3	The architecture should be able to generate high volumes of import declarations in real-time	**Yes**
Design requirements use case III: Transfer title to the goods using an electronic negotiable Bill of Lading		
Code	*Definition*	*Fulfilled*
UC3-req. 1	The architecture should provide business-to-business exchange to transfer ownership of the electronic negotiable bill of lading to a new holder	**Yes**
UC3-req. 2	The architecture should guarantee uniqueness of the electronic negotiable Bill of Lading following the principles for functional equivalence	**Yes**
UC3-req. 3	The architecture should ensure confidentiality of both the electronic negotiable Bill of Lading and identities of businesses involved in the transfer of ownership and should allow only authorized parties access to the electronic negotiable Bill of Lading or identity	**Yes**
UC3-req. 4	The architecture should be open to businesses, without the use of a centralized title registry and without a central authority in control	**Yes****
UC3-req. 5	The architecture should be able to handle high volumes of ownership transfers of electronic negotiable Bill of Ladings in real-time	**Yes**

** Note: the use case relies on assumption that use case I is implemented. In addition, as current smart contract technology requires data to be stored on-chain, this could conflict with UC1-req. 5 which requires absence of a centralized storage. Required data might therefore not be available.*
*** Note: the assumption was made that there is no need for an additional legal agreement to be signed by businesses in order to use the architecture.*

8.2 Answering the main research question

Based on the conclusions of the sub questions, this section answers the main research question: '*Which design of a blockchain-based architecture can be developed to support exchange of trade documents in trade lanes that takes into account the requirements of openness, security and scalability?*'. This research demonstrates that a blockchain-based architecture can be developed that is open, secure and scalable to support each of the critical trade document use cases to exchange trade documents. To that extent, for each use case, the architecture is governed by a consortium of parties to guarantee openness. Also, the architecture uses a permissioned network configuration and data encryption for security purposes, a data structure and storage to store transactions including references to trade documents, a consensus mechanism with PBFT as consensus protocol to provide scalability and an application component uses smart contracts to implement all business logic to support exchange. The blockchain-based architecture design of each critical trade document use case differs per case. The underlying reason is that the specificity of the use case affects design choices. In the architecture, the data structure and storage and application components are affected most. Depending on the use case, these components require a different design to ensure security and to handle all required business logic that is needed to support the use case.

The differences – even though subtle - in design of the blockchain-based architecture for each critical trade document use case shows that generalization of a blockchain-based architecture design to support exchange of any trade document in trade lanes is not evident. A blockchain-based architecture is not a black box that can simply be used to support all kinds of use cases. Differences between use cases influence the design of the blockchain-based architecture, especially regarding data structure and storage and the application. In-depth analysis of individual use cases is therefore needed to come up with effective designs for a blockchain-based architecture. However, because the architecture supports all three critical trade documents some generalization can be made. The exchange of a pro forma invoice is the base case of the data pipeline concept. It contains the most relevant data and other documents are based upon this data. The electronic negotiable Bill of Lading is the most difficult document to exchange due to its requirement for guarantee of uniqueness. Exchange of other documents will be easier. A generalized blockchain-based architecture design that is positioned between the architecture designs that supports the use cases will therefore probably support the exchange of other trade documents as well.

Whether the blockchain-based architecture supports the exchange of trade documents in real-world application remains uncertain. The architecture design is based on requirements that stem from academic literature on the data pipeline concept or electronic negotiable Bill of Lading. The data pipeline concept literature is the result of previous research efforts that involved both public authorities and businesses (e.g., EU CORE project). In addition, the architecture design is evaluated by experts that are involved in the development of a blockchain-based platform to exchange trade data. This to some extent provides confidence that the architecture design provides a good basis to support exchange of trade documents as it is well-embedded in the trade lane domain. Therefore, the blockchain-based architecture design is a good first step in the implementation process of an architecture to support actual exchange of trade documents that can benefit both customs authorities and businesses in trade lanes.

8.3 Scientific relevance

This research fits within wider research on e-government that focusses on the re-use of commercial (trade) data that can be used to gain public value (Jarman & Luna-Reyes, 2016). This re-use can for example enable customs authorities to perform better risk assessment (Klievink et al., 2016; Rukanova et al., 2017). Also, the research fits within more recent research on the use of blockchain technology in e-government with many potential benefits to create public value (Ølnes et al., 2017). More specifically, the research contributes to the knowledge base on the data pipeline concept which is a specific research topic within e-government research that aims at improving the exchange of trade data in trade lanes (Klievink et al., 2012). The contribution is two-fold. First, the research proposes an open, secure and scalable blockchain-based architecture that enables a novel approach to the business-to-business and business-to-government exchange of trade documents that can be used for piggybacking by customs authorities for risk assessment and cross-validation purposes. Second, it proposes an extension of the exchange of trade documents by enabling automatic generation of an import declaration that takes away the need for manual-cross validation. Another scientific contribution of this research is that it adds to research on the GTD platform regarding the scalability issue of blockchain technology in terms of transaction throughput. To that extent, the research proposes a blockchain-based architecture design that uses PBFT as consensus protocol in which network consortium parties act as ordering nodes to overcome the node scalability issue of PBFT. This design can overcome the limited transaction throughput which was not proposed before in literature in the context of exchange of trade documents and specifically the critical trade document use cases. In addition, this research contributes to the knowledge base on the electronic negotiable Bill of Lading by proposing an open, secure and scalable blockchain-based architecture to support the electronic negotiable Bill of Lading which overcomes the knowledge gap.

The findings of this research have demonstrated that, tough subtle, there are differences in design of a blockchain-based architecture not only between domains, but also within a single domain for different use cases. Generalization of blockchain-based architecture designs for different use cases is thus not straightforward. Use cases should therefore be analyzed separately. To that extent, the research performs an in-depth analysis for three use cases. This provides a contribution to literature as in the current knowledge base of blockchain technology the architecture is often seen as a black box with exaggerated benefits (Ølnes et al., 2017). The research followed a structured design process to develop a blockchain-based architecture design. The identified core components and sub components can be used in future research to explore how a blockchain-based architecture can support use cases to exchange trade documents. The design process therefore provides a contribution to academic literature. The governance of the architecture is not explicitly mentioned in academic literature on blockchain-based architecture design. Underlying reason is that literature is mostly technical-oriented. However, this research has shown that for architecture design, addressing governance of the architecture is important. Especially regarding openness, governance of the architecture needs to be addressed. For example, identity and permission management needs to be governed to ensure all parties can use the architecture. Also, governance is needed to decide who can participate in the consensus protocol. In addition, development of standards which is crucial to exchange trade documents should be governed. Therefore, the need to explicitly address governance of a blockchain-based architecture is a contribution to academic literature on architecture design.

8.4 Societal relevance

Besides scientific relevance, the blockchain-based architecture design also has societal relevance. Customs authorities need novel ways to enable exchange of trade data including documents that can be used to improve risk assessment and cross-validation. Improvement can reduce tax fraud and increase security of goods that enter a customs territory. This research provides a new approach to the exchange of trade documents that can potentially realize these improvements. Customs authorities can more easily pull a pro forma invoice document and piggyback the document for manual cross-validation using the architecture design. Using the architecture design, customs authorities can benefit even further as manual-cross validation is no longer needed which reduces time-consuming bureaucracy. In context of the electronic negotiable Bill of Lading, businesses can make the required shift from paper-based to an electronic negotiable Bill of Lading. Evaluation of the effectiveness and technical correctness of the blockchain-based architecture design and relevance of the use cases with experts from the field in chapter 7 shows that parties - both customs authority and businesses - involved in the development of blockchain-based solutions related to the data pipeline concept can benefit from the design that supports exchange of critical trade documents in an open, secure and scalable way.

The next section reflects on the research by discussing limitations of the research and identifying potential future work. Section 8.6 links the thesis with the CoSEM programme.

8.5 Reflection

The final section of this thesis reflects on the research. First, section 8.5.1 the limitations of the research outcomes. Next, future research to address these limitations is discussed in section 8.5.2.

8.5.1 Limitations

This section discusses limitations of this research and puts the research results into perspective.

Only design and evaluation of support of exchange of trade documents

Within information systems there is often a difference between design and practical real-world implementation. This research only provided a design for a blockchain-based architecture that supports the exchange of trade documents (illustrated by the use cases) and evaluated the design with experts. A valid question that therefore arises is: will the design actually work in practice when implementing the architecture to exchange trade documents? The architecture is evaluated by experts that are involved in the development of a blockchain-based platform, the Global Trade Digitization (GTD) platform, that aims at improving data exchange within trade lanes. Their practical expertise on implementing architectures to exchange trade documents provides confidence that the architecture design forms a good basis for future research into implementation of the architecture to exchange trade documents. This future research could include the development of a Proof of Concept (PoC) based on the architecture design to validate the architecture from a practical point of view.

Use case selection

Within trade lanes many trade documents are exchanged. The research focused only on use cases for three trade documents: pro forma invoice, import declaration and electronic negotiable Bill of Lading. The differences in the blockchain-based architecture design for these three documents shows that generalization of the architecture for other trade documents cannot easily be made. This potentially limits the applicability of the architecture design for the exchange of other trade documents and thus general conclusions in context of exchange of trade documents within international trade lanes. However, as mentioned by Klievink, Stijn and Hesketh et al. (2012) the use of a selected number of cases does not have to be a limitation. In this research, the three use cases where explicitly chosen due to their critical importance. In use case I, the exchange of a pro forma invoice document is the 'base' case of the exchange of trade documents as illustrated by the data pipeline concept. Almost all other trade documents, like an import declaration are aggregated (partially) from this document. If exchange of the pro forma invoice document would be infeasible, exchange of any other trade document would as a result also become difficult to realize. As evaluation of the architecture design shows that it supports exchange of the pro forma invoice for piggybacking to perform cross-validation of the import declaration, in essence any other document (except for the negotiable Bill of Lading) can be exchanged using the same architecture. It would only require some rules (e.g., confirmation policy in the smart shipment contract) to be adapted to the specific document. This, to an extent, allows for generalization of the research outcomes as the architecture is applicable to any other trade document that can be used for cross-validation of the import declaration. In addition, the import declaration in use case II plays an important role in cross-border activities. Without declaration, goods cannot be shipped. It is therefore one of the most crucial documents to exchange. Simplifying the bureaucratic and cumbersome process of lodging an import declaration and cross-validation by automation can have major benefits. In addition, use cases I and II involve all relevant trade lane parties including customs authorities and businesses which adds to the relevance within the domain.

Due to its special nature, the selection of the electronic negotiable Bill of Lading in use case III is critical as it is the most difficult document to exchange: it requires the guarantee of uniqueness that other documents do not need. If use case III can be realized, the exchange of other documents should in theory be possible. The evaluation step (see chapter 7) also included validation of the relevance of the use cases by representatives from Maersk Line and the Customs administration of the Netherlands which adds to the validity of the research as it well embedded it in the problem domain. In conclusion, realization of these use cases can be a major step towards exchange of documents within international trade lanes and can therefore be a basis for future research. The use case selection of these three critical trade documents is therefore not a limitation to the research. The design process of this research that designed an architecture based on the critical trade document use cases can be applied to other use cases that exchange trade documents in the future.

Scalability of blockchain-based architecture design only proved theoretically and between only two consensus protocols
The notion of sufficient scalability of the blockchain-based architecture design is only argued for theoretically and based on expert opinions. However, this does not prove that the blockchain-based architecture design is actually sufficiently scalable in real-world application. Therefore, a working PoC based on the blockchain-based architecture design in this research should be developed to create scalability benchmarking of the proposed design by measuring transaction throughput. From the interview with IBM representatives, use case III on the electronic negotiable Bill of Lading seems most suitable for a first version of the PoC as it would only require a simple smart contract with the ability to transfer ownership.

In addition, practical knowledge on the consensus protocols is very limited due to their novelty. Proof-of-Work and PBFT are the only two protocols that are analyzed extensively in research and currently used in practice. Therefore, in the consensus mechanism protocol component of the blockchain-based architecture only Proof-of-Work and PBFT were considered as design options. Not considering other protocols could affect validity of the architecture as a different protocol might perform significantly better. However, the current blockchain-based architecture is modular in the sense that only the 'transaction ordering' in the consensus mechanism component would change if another consensus protocol was chosen as the protocol is only used for ordering of transactions and block proposal. This would not affect the rest of the architecture. Thus, the limited comparison of consensus protocols does not affect validity of the rest of the architecture.

Evaluation limited to only few expert interviews
For the evaluation of the blockchain-based architecture design, only interviews with representatives of the Customs Administration of the Netherlands, IBM and Maersk Line were held. All interviewees are involved in the development of the GTD platform. This can lead to a potential positive bias towards not only the perceived benefits of using blockchain technology to exchange trade data, but also regarding the blockchain-based architecture design. The design shows similarities to the Hyperledger Platform the GTD platform is based on. For example, the transaction process execute – order – validate is the result of research on Hyperledger. The interviewees from IBM were aware of potential bias as they clearly stated this during the interviews. To mitigate this potential bias, more extensive evaluation of the blockchain-based architecture design in which a broader spectrum of experts in the field that either have a different role within international shipping (e.g., representatives from Freight Forwarders) or are not involved in the GTD platform are interviewed could strengthen the evaluation and thus design of the blockchain-based architecture.

Assumption no need for additional legal framework electronic negotiable Bill of Lading
In use case III on the electronic negotiable Bill of Lading the assumption was made that there is no need for additional legal agreements (e.g., like a Bolero Rulebook) as the use case is more explorative of nature and takes a technical-oriented approach which assumes legislation to be in

place. The focus is on the aspect of guarantee of uniqueness and scalability of an architecture to implement an electronic negotiable Bill of Lading. However, real-world implementation of the blockchain-based architecture design could lead to issues as it would not fit with current legislation, thus affecting validity. To overcome this potential issue, the blockchain-based architecture design fits with the guiding principles for functional equivalence as defined by the Rotterdam Rules and UNCITRAL Model Law on Electronic Transferable Records. Any future (inter)national (maritime) legislation will be based upon these guiding principles. Therefore, with regard to the validity of the blockchain-based architecture design for use case III it can be concluded that the approach taken does not negatively affect the validity of the design once legislation is implemented.

8.5.2 Future research

Based on the limitations discussed in the previous section, this section identifies potential future research.

Research to develop Proof of Concept based on architecture design

To close the gap between design and practical real-world implementation to determine if the architecture supports exchange of trade documents, development of a PoC based on the architecture design is proposed. This PoC can be used to determine if exchange of trade documents is actually supported in real-world application. From the interview with IBM representatives, use case III on the electronic negotiable Bill of Lading seems most suitable for a first version of the PoC as it would only require a simple smart contract with the ability to transfer ownership.

Research to develop Proof of Concept for scalability benchmarking

The notion of sufficient scalability of the blockchain-based architecture design is only evaluated theoretically and based on expert opinions. However, this does not prove that the blockchain-based architecture design is actually sufficiently scalable in real-world implementation. Therefore, a working PoC based on the blockchain-based architecture design in this research should be developed to enable scalability benchmarking of the proposed design by measuring transaction throughput.

Research into data element standardization

In use case II on the automated import declaration the assumption was made that required individual data elements (e.g., Incoterm, goods value, different costs) are available within the blockchain ledger. However, standards to realize this do not yet exist in context of blockchain technology. Therefore, use case II should be seen as an explorative use case showcasing the potential of blockchain technology for automating cross-border customs activities. Research on the applicability of existing standards to exchange trade data, such as UN/CEFACT, the WCO data model and GS1, within blockchain-based systems is therefore proposed.

Research on applicability of use case III within trade finance domain

Use case III in this research only focused on the transfer of ownership an electronic negotiable Bill of Lading to a new holder. However, the negotiable Bill of Lading is also often used in trade financing as collateral. If the consignor wants to be paid immediately for a shipment as soon as the goods are shipped, the negotiable Bill of Lading is placed under a Letter of Credit arrangement by a financial institution (e.g. a bank). As soon as the consignor transfers ownership of the negotiable Bill of Lading to the bank, the consignor will receive payment for the goods. The bank coordinates with the consignee’s bank to transfer ownership to the consignee upon payment by the consignee to its bank. As this research has shown that the transfer of title to the goods using an electronic negotiable Bill of Lading can be supported by a blockchain-based architecture, an extension of the use case could include placing the negotiable Bill of Lading under a Letter of Credit arrangement to digitize the process. Therefore, future research on extension of this use case should be performed.

Research on effect additional legal agreement on adoption of blockchain-based electronic negotiable Bill of Lading
Previous initiatives that implement an electronic negotiable Bill of Lading lack, partially due to the cumbersome required registration and additional legal agreements, adoption. Whilst the architecture design overcomes some of the other adoption barriers regarding transparency of and control over the centralized title registry, it does not research the effect of the additional required legal agreements in this context. The assumption was made that the use case in this research is explorative of nature which takes a technical-oriented approach and assumes that upcoming legislation for an electronic negotiable Bill of Lading is implemented. This might however not fit with the current reality of lacking legislation. Therefore, additional research is needed to identity if the additional legal agreement is still a barrier for adoption if a blockchain-based architecture is used for an electronic negotiable bill of lading that overcomes all other adoption barriers.

Business model use cases
In the research, financing of the blockchain-based architecture was left outside the scope as the focus was more technical-oriented concerning exchange of trade documents (illustrated by the critical trade document use cases) using blockchain technology that addresses openness, security and scalability challenges. However, actual implementation of the use cases also requires more insight into financing. Some early efforts into research have been made by research on the GTD platform. For example, the current approach of the GTD platform is to make trade data freely available (e.g., the exchange of a pro forma invoice document in use case I) and allow businesses to offer paid services on top (e.g., the automated generation of an import declaration in use case II) of this data. This would relate to a platform-as-a-service business model. Further research into the potential of using the blockchain-based architecture design as platform-as-a-service, specifically for the critical trade document use cases is therefore proposed.

Research into relevance use case I and II from business perspective
Use cases I and II on were only evaluated from a customs perspective. However, end-to-end visibility of trade documents or reduced bureaucracy by automating the lodging of an import declaration might also have major benefits for businesses. Therefore, it is proposed that future research focusses on the business perspective to identify the relevance of the use cases. This future research closely relates to the previous proposal on the business model of the use cases.

8.6 Link with CoSEM programme
This thesis is part of the Complex Systems and Engineering (CoSEM) Master's programme. Therefore, this section discusses the link between the thesis and CoSEM programme. The CoSEM programme focusses on the design of complex socio-technical systems. These systems are characterized by the integration of multi-actor complexities (e.g., between public and private organizations) and technical complexities. This thesis focuses on the design of such complex socio-technical system (in the form of a blockchain-based architecture) by integrating the complexity of data exchange within trade lanes, consisting of both public and private organizations with different intentions, and highly complex and innovative blockchain technology. To provide more detail on the link with the CoSEM programme, for a typical CoSEM thesis a set of criteria should be fulfilled (as laid out in the Course Guide). These criteria are: 1) the work has clear design and/or engineering components, 2) the design has a clear technology component and technical issues are addressed, 3) both process management strategies and system engineering approaches are addressed, 4) complex design/engineering issues are dealt with in a systematic and creative way, 5) CoSEM methods, tools and techniques for creatively designing and assessing the impact of technical solutions in organizations are used, 6) the subject covers values originating from both the public and private domains. The fulfilment of each criterium is addressed in the next paragraph.

Criterium one is fulfilled as the thesis focusses on the design of a blockchain-based architecture. The use of blockchain technology and addressing technical challenges such as limited scalability fulfills criterium two on incorporating technological complexity. The design is structured by using a Design Science Research (DSR) approach, which fulfills criterium three and four as it systematically approaches the engineering of a system (the blockchain-based architecture) via several phases to design and evaluate the system. Courses on socio-technical system and software engineering as well as information architecture design within the CoSEM programme provide tools and theory for the design of the blockchain-based architecture. Examples of tools included in the thesis are requirement analysis and BPMN modeling. The thesis uses several use cases that incorporate the views of both public and private organizations (e.g., customs authorities, consignors, Freight Forwarders and carriers). The blockchain-based architecture design is based on requirements that addresses the needs of both types of organizations. This fits criterium six. Thus, the thesis fulfills all criteria for a typical CoSEM thesis and therefore clearly fits within the CoSEM programme.

Literature

Abeyratne, S. A., & Monfared, R. P. (2016). Blockchain ready manufacturing supply chain using distributed ledger. *International Journal of Research in Engineering and Technology*, *05*(09), 1–10. https://doi.org/10.15623/ijret.2016.0509001

Ainsworth, R. T., & Alwohaibi, M. (2015). *BLOCKCHAIN, BITCOIN, AND VAT IN THE GCC: THE MISSING TRADER EXAMPLE*. Retrieved from http://www.bu.edu/law/faculty-scholarship/working-paper-series/

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., … Yellick, J. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *EuroSys*. https://doi.org/10.1145/3190508.3190538

Apte, S., & Petrovsky, N. (2016). Will Blockchain Technology Revolutionize Supply Chain Applications? *Journal of Excipients and Food Chemicals*, *7*(3), 76–78. Retrieved from https://jefc.scholasticahq.com/article/910-will-blockchain-technology-revolutionize-excipient-supply-chain-management

Badzar, A. (2016). *Blockchain for securing sustainable transport contracts and supply chain transparency - An explorative study of blockchain technology in logistics*. Retrieved from https://lup.lub.lu.se/student-papers/search/publication/8880383

Benjumea, V., Lopez, J., Montenegro, J., & Troya, J. (2004). A first approach to provide anonymity in attribute certificates. https://doi.org/10.1007/978-3-540-24632-9_29

Bolero International Ltd. (1999). Bolero Rulebook.

Botton, N. (2018). *Blockchain and Trade: Not a Fix for Brexit, but Could Revolutionise Global Value Chains (If Governments Let It)*. Retrieved from https://www.econstor.eu/bitstream/10419/174812/1/ecipe-pb-2018-01.pdf

Buchmann, J. A., Karatsiolis, E., & Wiesmaier, A. (2013). *Introduction to Public Key Infrastructures*. https://doi.org/10.1007/978-3-642-40657-7

BusinessDictionary.com. (n.d.). Definition of a shipment. Retrieved August 10, 2018, from http://www.businessdictionary.com/definition/shipment.html

CargoX. (2018). *CargoX Whitepaper*.

Castro, M., & Liskov, B. (1999). Practical Byzantine fault tolerance. In *Proceedings of the third symposium on Operating systems design and implementation* (pp. 173–186). https://doi.org/10.1.1.17.7523

CGI. (2004). *Public Key Encryption and Digital Signature: How do they work?* Retrieved from http://www.cgi.com/files/white-papers/cgi_whpr_35_pki_e.pdf

Chalaemwongwan, N., & Kurutach, W. (2018). State of the art and challenges facing consensus protocols on blockchain. *International Conference on Information Networking*, 957–962. https://doi.org/10.1109/ICOIN.2018.8343266

Curwen, N. (2007). The bill of lading as a document of title at common law. In *THE BILL OF LADING* (pp. 139–162). Retrieved from http://ssudl.solent.ac.uk/66/

Customs Administration of the Netherlands. (2017). *Facts & Figures in brief*.

Dhillon, V., Metcalf, D., & Hooper, M. (2017). Foundations of Blockchain. In *Blockchain Enabled Applications* (pp. 15–24). Retrieved from https://link.springer.com/chapter/10.1007/978-1-4842-3081-7_3

Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., & Wang, J. (2018). Untangling Blockchain: A Data Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering*, *30*(7), 1366–1385. https://doi.org/10.1109/TKDE.2017.2781227

Dresch, A., Pacheco Lacerda, D., & Valle Antunes, J. A. (2015). *Design Science Research: A method for Science and Technology Advancement (book)*. Retrieved from https://link.springer.com/content/pdf/10.1007%2F978-3-319-07374-3.pdf

Drescher, D. (2017). *Blockchain basics: A non-technical introduction in 25 steps*. *Blockchain Basics: A Non-Technical Introduction in 25 Steps*. https://doi.org/10.1007/978-1-4842-2604-9

Dubovec, M. (2006). The Problems and Possibilities for using Electronic Bills of Lading as collateral. *Arizona Journal of International & Comparative Law*, *23*(2), 437–466. https://doi.org/10.3868/s050-004-015-0003-8

e-title. (n.d.). What does the Electronic Title User Group do? Retrieved June 27, 2018, from http://www.e-title.net/etug_what.php

EU CORE project. (2018). The Core Project - Vision. Retrieved September 20, 2018, from http://www.coreproject.eu/

EU TAXUD. (2017). Blockchain's potential: what governments need to consider.

European Commission. (2018). Documents for customs clearance | Trade Helpdesk. Retrieved April 17, 2018, from http://trade.ec.europa.eu/tradehelp/documents-customs-clearance

European Parliament, & European Council. Union Customs Code 952/2013 (2013).

Eyal, I., & Sirer, E. G. (2014). Majority is not enough: Bitcoin mining is vulnerable. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 8437, pp. 436–454). https://doi.org/10.1007/978-3-662-45472-

5_28
Fawcett, S. E., Osterhaus, P., Magnan, G. M., Brau, J. C., & McCarter, M. W. (2007). Information sharing and supply chain performance: The role of connectivity and willingness. *Supply Chain Management*, *12*(5), 358–368. https://doi.org/10.1108/13598540710776935
Forrester, I., & Odarda, O. E. (2005). THE AGREEMENT ON CUSTOMS VALUATION. In *The World Trade Organization: Legal, Economic and Political Analysis* (pp. 532–572). Retrieved from https://link-springer-com.tudelft.idm.oclc.org/content/pdf/10.1007%2F0-387-22688-5_12.pdf
Francisco, K., & Swanson, D. (2018). The Supply Chain Has No Clothes: Technology Adoption of Blockchain for Supply Chain Transparency. *Logistics*, *2*(1), 2. https://doi.org/10.3390/logistics2010002
Francisconi, M. (2017). *An explorative study on blockchain technology in application to port logistics*.
Gao, Z., Xu, L., Chen, L., Zhao, X., Lu, Y., & Shi, W. (2018). CoC: A Unified Distributed Ledger Based Supply Chain Management System. *Journal of Computer Science and Technology*, *33*(2), 237–248. https://doi.org/10.1007/s11390-018-1816-5
Garay, J., Kiayias, A., & Leonardos, N. (2017). The Bitcoin Backbone Protocol: Analysis and Applications. Retrieved from https://eprint.iacr.org/2014/765.pdf
Goldby, M. (2008). Electronic bills of lading and central registries: what is holding back progress? *Information & Communications Technology Law*, *17*(2), 125–149. https://doi.org/10.1080/13600830802239381
Governatori, G., Idelberger, F., Milosevic, Z., Riveret, R., Sartor, G., & Xu, X. (2018). On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artificial Intelligence and Law*, *26*(2), 377–409. https://doi.org/10.1007/s10506-018-9223-3
Gramoli, V. (2017). From blockchain consensus back to Byzantine consensus. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2017.09.023
GS1. (2013). GS1 Identification Keys in Transport & Logistics GS1; GS1 Guideline. Retrieved from http://www.gs1.org/docs/tl/T_L_Keys_Implementation_Guideline.pdf
Hamida, E. Ben, Brousmiche, K. L., Levard, H., & Thea, E. (2017). Blockchain for Enterprise : Overview , Opportunities and Challenges. In *The Thirteenth International Conference on Wireless and Mobile Communications (ICWMC 2017)*.
Han, C.-R., & McGauran, R. (2014). Tracing trails: implications of tax information exchange programs for customs administrations. *World Customs Journal*, *8*(2), 3–14. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.739.854&rep=rep1&type=pdf#page=82
Hesketh, D. (2009). Seamless Electronic Data and Logistics Pipelines Shift Focus From Import Declarations To Start of Commercial Transaction. *World Customs Journal*, *3*(1), 27–32. Retrieved from http://worldcustomsjournal.org/Archives/Volume 3%2C Number 1 (Apr 2009)/04 Hesketh.pdf
Hesketh, D. (2010). Weaknesses in the supply chain : who packed the box ? *World Customs Journal*, *4*(2), 3–20.
Hevner, A. (2007). A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, *19*(2), 87–92. https://doi.org/http://aisel.aisnet.org/sjis/vol19/iss2/4
Hevner, A., & Chatterjee, S. (2010). *Design Science Research in Information Systems; Theory and Practice* (Vol. 22). https://doi.org/10.1007/978-1-4419-5653-8
Hevner, A., March, S., Park, J., & Ram, S. (2004). DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH. *MIS Quarterly*, *28*(1), 75–105. Retrieved from https://pdfs.semanticscholar.org/fa72/91f2073cb6fdbdd7c2213bf6d776d0ab411c.pdf
Hofman, W., & Bastiaansen, H. (2014). *Towards an IT infrastructure for compliance management by data interoperability – the changing role of authorities*. *International Journal of Advanced Logistics*.
Hong, G. (2012). Electronic bill of lading's title transfer in international e-commerce. In *Proceedings of the 2012 2nd International Conference on Business Computing and Global Informatization, BCGIN 2012* (pp. 354–356). https://doi.org/10.1109/BCGIN.2012.98
Hu, V. C. V., Ferraiolo, D. F., & Kuhn, D. R. (2006). *Assessment of access control systems*. *NIST Interagency Report 7316*. https://doi.org/10.6028/NIST.IR.7316
Hulstijn, J., Overbeek, S., Aldewereld, H., & Christiaanse, R. (2012). Integrity of supply chain visibility: Linking information to the physical world. In *Lecture Notes in Business Information Processing* (Vol. 112, pp. 351–365). https://doi.org/10.1007/978-3-642-31069-0_29
ICC. (2010). Incoterms® rules 2010. Retrieved May 23, 2018, from https://iccwbo.org/resources-for-business/incoterms-rules/incoterms-rules-2010/
IHS Markit. (2017). PIERS: The complete US Import/Export Bill of Lading data. Retrieved August 10, 2018, from https://ihsmarkit.com/products/piers.html
IMO. (2018). International Maritime Organization profile. Retrieved September 19, 2018, from https://business.un.org/en/entities/13
ISO. (2011). Systems and software engineering - Architecture description. Retrieved from http://cabibbo.dia.uniroma3.it/asw/altrui/iso-iec-ieee-42010-2011.pdf

Jarman, H., & Luna-Reyes, L. (2016). *Private Data and Public Value; Governance, Green Consumption, and Sustainable Supply Chains*. https://doi.org/10.1007/978-3-319-27823-0

Jensen, T. (2017). *SHIPPING INFORMATION PIPELINE; AN INFORMATION INFRASTRUCTURE TO IMPROVE INTERNATIONAL CONTAINERIZED SHIPPING*.

Jensen, T., Vatrapu, R., & Bjørn-Andersen, N. (2018). Avocados crossing borders: The problem of runaway objects and the solution of a shipping information pipeline for improving international trade. *Information Systems Journal*, *28*(2), 408–438. https://doi.org/10.1111/isj.12146

Jiang, J., Aldewereld, H., Dignum, V., Wang, S., & Baida, Z. (2015). Regulatory compliance of business processes. *AI & SOCIETY*, *30*(3), 393–402. https://doi.org/10.1007/s00146-014-0536-9

Johannesson, P., & Perjons, E. (2014). *An Introduction to Design Science*. *Springer International Publishing Switzerland*. https://doi.org/10.1007/978-3-319-10632-8

Karame, G. O., Androulaki, E., & Capkun, S. (2012). Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin. *IACR Cryptology EPrint*. https://doi.org/10.1145/2382196.2382292

Karan, H. (2011). Transport Documents in the Light of the Rotterdam Rules. In *The United Nations Convention on Contracts for the International Carriage of Goods Wholly or Partly by Sea* (pp. 229–248). https://doi.org/10.1007/978-3-642-19650-8_9

Karp, A. H., Haury, H., & Davis, M. H. (2009). From ABAC to ZBAC : The Evolution of Access Control Models.

Kim, H. M., & Laskowski, M. (2018). Towards an ontology-driven blockchain design for supply-chain provenance. *Intelligent Systems in Accounting, Finance and Management*, *25*(1), 18–27. https://doi.org/10.1002/isaf.1424

Klievink, B., Bharosa, N., & Tan, Y.-H. (2016). The collaborative realization of public values and business goals: Governance and infrastructure of public–private information platforms. *Government Information Quarterly*, *33*, 67–79. https://doi.org/10.1016/j.giq.2015.12.002

Klievink, B., & Lucassen, I. (2013). Facilitating adoption of international information infrastructures: a Living Labs approach. In *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-642-40358-3_21

Klievink, B., van Stijn, E., Hesketh, D., Aldewereld, H., Overbeek, S., Heijmann, F., & Tan, Y.-H. (2012). Enhancing Visibility in International Supply Chains. *International Journal of Electronic Government Research*, *8*(4), 14–33. https://doi.org/10.4018/jegr.2012100102

Klievink, B., & Zomer, G. (2015). IT-Enabled Resilient, Seamless and Secure Global Supply Chains: Introduction, Overview and Research Topics. In *I3E 2015, LNCS 9373* (pp. 443–453). https://doi.org/10.1007/978-3-319-25013-7_36

Knol, A., Klievink, B., & Tan, Y.-H. (2014). Data Sharing Issues and Potential Solutions for Adoption of Information Infrastructures: Evidence from a Data Pipeline Project in the Global Supply Chain over Sea. In *BLED 2014 Proceedings* (Vol. 40). Retrieved from http://aisel.aisnet.org/bled2014

Korpela, K., Hallikas, J., & Dahlberg, T. (2017). Digital Supply Chain Transformation toward Blockchain Integration. *Proceedings of the 50th Hawaii International Conference on System Sciences*, 4182–4191. https://doi.org/http://hdl.handle.net/10125/41666

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. Retrieved from http://www-inst.eecs.berkeley.edu/~cs162/fa12/hand-outs/Original_Byzantine.pdf

Lee, E. (2012). International Transportation. In *Management of International Trade* (pp. 187–217). https://doi.org/10.1007/978-3-642-30403-3_6

Lee, R., Nguyen, V., & Pagnoni, A. (2008). Securing Uniqueness of Rights e-Documents: A Deontic Process Perspective. *Journal of Theoretical and Applied Electronic Commerce Research*, *3*(3), 83–102. https://doi.org/10.4067/S0718-18762008000200007

Lemieux, V. L. (2016). Trusting records: is Blockchain technology the answer? *Records Management Journal*, *26*(2), 110–139. https://doi.org/10.1108/RMJ-12-2015-0042

Lin, I.-C., & Liao, T.-C. (2017). A Survey of Blockchain Security Issues and Challenges. *International Journal of Network Security*, *1919*(55), 653–659. https://doi.org/10.6633/IJNS.201709.19(5).01)

Martincus, C. V., Carballo, J., & Graziano, A. (2015). Customs. *Journal of International Economics*, *96*(1), 119–137. https://doi.org/10.1016/j.jinteco.2015.01.011

Mcdermott, B. C., Nagle, J., Horowitz, M., Johnson, S., & Nagle, J. (2017). From Bills Of Lading To Blockchain Structures : Part 1.

Mendling, J., Weber, I., van der Aalst, W., Brocke, J. vom, Cabanillas, C., Daniel, F., … Zhu, L. (2017). Blockchains for Business Process Management - Challenges and Opportunities. *ACM Transactions on Management Information Systems*, 1–20. https://doi.org/10.1145/3183367

Milani, F., García-Bañuelos, L., & Dumas, M. (2016). Blockchain and Business Process Improvement. *BPTrends*, 4. Retrieved from www.bptrends.com

Morabito, V. (2017). *Business Innovation Through Blockchain*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-48478-5_2

Nærland, K., Müller-Bloch, C., Beck, R., & Palmund, S. (2017). Blockchain to Rule the Waves-

Nascent Design Principles for Reducing Risk and Uncertainty in Decentralized Environments. *Thirty-Eighth International Conference on Information Systems (ICIS)*, 1–16.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. https://doi.org/10.1007/s10838-008-9062-0

Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies. *Journal of Information Technology*, *21*(4), 284–298. https://doi.org/10.1057/palgrave.jit.2000080

Nguyen, G. T., & Kim, K. (2018). A survey about consensus algorithms used in Blockchain. *Journal of Information Processing Systems*, *14*(1), 101–128. https://doi.org/10.3745/JIPS.01.0024

Ølnes, S. (2016). Beyond Bitcoin enabling smart government using blockchain technology. In *Lecture Notes in Computer Science* (Vol. 9820, pp. 253–264). https://doi.org/10.1007/978-3-319-44421-5_20

Ølnes, S., Ubacht, J., & Janssen, M. (2017). Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, *34*(3), 355–364. https://doi.org/10.1016/j.giq.2017.09.007

Overbeek, S., Klievink, B., Hesketh, D., Heijmann, F., & Tan, Y. H. (2011). A Web-based data pipeline for compliance in international trade. In *CEUR Workshop Proceedings* (Vol. 769, pp. 32–48).

P&I. (n.d.). About the Group. Retrieved June 27, 2018, from https://www.igpandi.org/about

Pagnoni, A., & Visconti, A. (2010). Secure electronic bills of lading: blind counts and digital signatures. *Electron Commer Res*, *10*, 363–388. https://doi.org/10.1007/s10660-010-9060-2

Park, J., Sandhu, R., & Ahn, G.-J. (2001). Role-based access control on the web. *ACM Transactions on Information and System Security*, *4*(1), 37–71. https://doi.org/10.1145/383775.383777

Pass, R., Seeman, L., & Shelat, A. (2017). Analysis of the Blockchain Protocol in Asynchronous Networks. In *EUROCRYPT 2017, part II, LNCS* (Vol. 10210, pp. 643–673). https://doi.org/10.1007/978-3-319-56620-7

Pilkington, M. (2015). *Blockchain Technology: Principles and Applications*. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2662660&rec=1&srcabs=2749514&alg=1&pos=2

Pruksasri, P., Berg, J. Van Den, Hofman, W., & Tan, Y. H. (2014). Data Concealing of Supply Chain Transactions using the Distributed Trust Backbone. In *The 9th International Conference for Internet Technology and Secured Transactions, ICITST* (pp. 151–156). https://doi.org/10.1109/ICITST.2014.7038796

Pruksasri, P., Van Den Berg, J., & Hofman, W. (2012). THREE PROTOCOLS FOR SECURING THE DATA PIPELINE OF THE INTERNATIONAL SUPPLY CHAIN. In *IADIS International Conference e-Commerce* (pp. 27–34).

Pruksasri, P., Van Den Berg, J., Hofman, W., & Daskapan, S. (2013). Multi-level Access Control in the Data Pipeline of the International Supply Chain System. *Innovation in the High-Tech Economy*, 79–90. https://doi.org/10.1007/978-3-642-41585-2_7

Ramachandran, M., & Mahmood, Z. (2017). *Requirements Engineering for Service and Cloud Computing*. https://doi.org/10.1007/978-3-319-51310-2

Ramberg, J. (2011). *ICC Guide to Incoterms 2010*. Retrieved from http://store.iccwbo.org/content/uploaded/pdf/ICC-Guide-To-Incoterms®-2010.pdf

Rizzo, P. (2015). Wave Brings Blockchain Trade Finance Trial to Barclays. Retrieved June 22, 2018, from https://www.coindesk.com/wave-blockchain-trade-finance-barclays/

Rukanova, B., Henriksen, H. Z., Henningsson, S., & Tan, Y.-H. (2016). *The anatomy of digital trade infrastructures*. https://doi.org/10.1007/978-3-319-64930-6_14

Rukanova, B., Huiden, R., & Tan, Y.-H. (2017). Coordinated border management through digital trade infrastructures and trans-national government cooperation: The FloraHolland case. In *EGOV 2017, Lecture Notes in Computer Science* (Vol. 10428, pp. 240–252). https://doi.org/10.1007/978-3-319-64677-0_20

Rukanova, B., Wigand, R. T., van Stijn, E., & Tan, Y.-H. (2015). Understanding transnational information systems with supranational governance: A multi-level conflict management perspective. *Government Information Quarterly*, *32*(2), 182–197. https://doi.org/10.1016/j.giq.2014.12.003

Rüsch, S. (2018). High-Performance Consensus Mechanisms for Blockchains. In *EuroDW'18*. Retrieved from http://conferences.inf.ed.ac.uk/EuroDW2018/papers/eurodw18-Rusch.pdf

Sato, T., & Himura, Y. (2018). Smart-Contract Based System Operations for Permissioned Blockchain. In *9th IFIP International Conference on New Technologies, Mobility and Security, NTMS* (pp. 1–6). https://doi.org/10.1109/NTMS.2018.8328745

Scherer, M. (2017). *Performance and Scalability of Blockchain Networks and Smart Contracts*. Retrieved from https://umu.diva-portal.org/smash/get/diva2:1111497/FULLTEXT01.pdf

Schläger, C., Priebe, T., Liewald, M., & Pernul, G. (2007). Enabling Attribute-based Access Control in Authentication and Authorisation Infrastructures. In *20th Bled eConference eMergence* (pp. 814–826). Retrieved from http://epub.uni-regensburg.de/6467/1/bled2007-

enablingABAC_AAIs_HPifs.pdf
Sillaber, C., & Waltl, B. (2017). Life Cycle of Smart Contracts in Blockchain Ecosystems. *Datenschutz Und Datensicherheit - DuD*, *41*(8), 497–500. https://doi.org/10.1007/s11623-017-0819-7
Single Window for Logistics Luxembourg. (2017). Calculation of customs value; practical guide.
Smolander, K., Rossi, M., & Pekkola, S. (2017). Infrastructures, Integration and Architecting during and after Digital Transformation. In *Proceedings - 2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems, JSOS* (pp. 23–26). https://doi.org/10.1109/JSOS.2017.1
Sparka, F. (2010). Bills of Lading and other Maritime Transport Documents. In *Jurisdiction and Arbitration Clauses in Maritime Transport Documents: A Comparative Analysis* (Vol. 19). https://doi.org/10.1007/978-3-642-10222-6_3
Sukhodolskiy, I. A., & Zapechnikov, S. V. (2017). An Access Control Model for Cloud Storage Using Attribute-Based Encryption. In *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)* (pp. 578–581). https://doi.org/10.1109/EIConRus.2017.7910620
Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. *First Monday*, *2*(9), 9–13. https://doi.org/http://dx.doi.org/10.5210/fm.v2i9.548
Takahashi, K. (2016). Blockchain technology and electronic bills of lading. *THE JOURNAL OF INTERNATIONAL MARITIME LAW*, *22*, 202–211.
Tan, Y., Kouwenhoven, N., & Buhmann, N. (2018). Presentation Global Trade Digitization (GTD) Platform.
Tasca, P., & Tessone, C. (2018). *Taxonomy of Blockchain Technologies. Principles of Identification and Classification*. Retrieved from https://arxiv.org/pdf/1708.04872.pdf
Thomas, J., & Tan, Y. H. (2015). Key design properties for shipping information pipeline. In *I3E 2015, Lecture Notes in Computer Science* (Vol. 9373, pp. 491–502). https://doi.org/10.1007/978-3-319-25013-7_40
Triepels, R., Daniels, H., & Feelders, A. (2018). Data-driven fraud detection in international shipping. *Expert Systems with Applications*, *99*, 193–202. https://doi.org/10.1016/j.eswa.2018.01.007
Triepels, R., Feelders, A., & Daniels, H. (2015). Uncovering Document Fraud in Maritime Freight Transport Based on Probabilistic Classification. In *14th IFIP TC 8 International Conference, CISIM 2015, proceedings* (pp. 282–293). https://doi.org/10.1007/978-3-319-24369-6
Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys & Tutorials*, *18*(3), 2084–2123. https://doi.org/10.1109/COMST.2016.2535718
U.S. Customs and Border Protection. (2017). Importer- What is a Pro Forma invoice? Retrieved May 23, 2018, from https://help.cbp.gov/app/answers/detail/a_id/1285/~/importer--what-is-a-pro-forma-invoice%3F
UNCITRAL. (2017). *UNCITRAL Model Law on Electronic Transferable Records*. Retrieved from http://www.uncitral.org/pdf/english/texts/electcom/MLETR_ebook.pdf
United Nations. United Nations Convention on the International Carriage of goods wholly or partly by sea (2008). Retrieved from http://www.rotterdamrules.com/sites/default/files/pdf/convention.pdf
United Nations Economic and Social Council. (2018). *White Paper on technical application of Blockchain to United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) deliverables*.
UPS. (2014). To do it right, rely on UPS. Retrieved from https://www.ups.com/media/en/UPSInternationalShippingHowToGuide.pdf
Urciuoli, L., Hintsa, J., & Ahokas, J. (2013). Drivers and barriers affecting usage of e-Customs - A global survey with customs administrations using multivariate analysis techniques. *Government Information Quarterly*, *30*(4), 473–485. https://doi.org/10.1016/j.giq.2013.06.001
Van Boom, W. H. (1997). Certain legal aspects of electronic bills of lading. *European Transport Law*, *32*(1), 9–24. Retrieved from http://www.professorvanboom.eu/pdf_files/1997_electronic bills of lading.pdf
van Engelenburg, S., & Janssen, M. (2018). A Blockchain Architecture for Reducing the Bullwhip Effect. In *BMSD 2018, LNBIP* (Vol. 319, pp. 69–82). https://doi.org/10.1007/978-3-319-94214-8
van Engelenburg, S., Janssen, M., & Klievink, B. (2017). Design of a software architecture supporting business-to-government information sharing to improve public safety and security Combining business rules, Events and blockchain technology. *J Intell Inf Syst*. https://doi.org/10.1007/s10844-017-0478-z
van Engelenburg, S., Janssen, M., Klievink, B., Engelenburg, S. Van, Janssen, M., Klievink, B., … Klievink, B. (2015). Design of a Business-to-Government Information Sharing Architecture Using Business Rules. In *Software Engineering and Formal Methods 2015 Workshops, LNCS* (Vol. 9509, pp. 124–138). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-319-

15201-1
van Engelenburg, S., Janssen, M., Klievink, B., & Tan, Y.-H. (2017). Comparing a Shipping Information Pipeline with a Thick Flow and a Thin Flow. In *EGOV 2017, LNCS* (pp. 228–239). Springer, Cham. https://doi.org/10.1007/978-3-319-64677-0_19
Viryasitavat, W., Xu, L. Da, Zhuming, B., & Sapsomboon, A. (2018). Blockchain-based business process management (BPM) framework for service composition in industry 4.0. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-018-1422-y
Vukolić, M. (2016). The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *Lecture Notes in Computer Science* (Vol. 9591, pp. 112–125). https://doi.org/10.1007/978-3-319-39028-4_9
Vukolić, M. (2017). Rethinking Permissioned Blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts - BCC '17* (pp. 3–7). https://doi.org/10.1145/3055518.3055526
WCO. (2005). WCO Guide To Customs Valuation and Transfer Picing. Retrieved from http://www.wcoomd.org/en/topics/key-issues/revenue-package/~/media/36DE1A4DC54B47109514FFCD0AAE6B0A.ashx
Weigand, H., & Bukhsh, F. A. (2011). Supporting Customs Controls by Means of Service-Oriented Auditing. In *IFIP Advances in Information and Communication Technology* (Vol. 353, pp. 28–43). https://doi.org/10.1007/978-3-642-27260-8_3
Wieringa, R. (2014). *Design Science Methodology; for Information Systems and Software Engineering*.
Wolffgang, H.-M., & Dallimore, C. (2013). The valuation of goods for customs purposes. In *European Yearbook of International Economic Law (EYIEL)* (pp. 391–411). https://doi.org/10.1007/978-3-642-33917-2_15
WTO. (n.d.). Customs Valuation - Technical Information. Retrieved May 22, 2018, from https://www.wto.org/english/tratop_e/cusval_e/cusval_info_e.htm
Wu, C., Starr, L., & Tan, J. (2017). *Electronic Bills of Lading*. *UK P&I Club: Legal Briefing*. Retrieved from https://www.ukpandi.com/fileadmin/uploads/uk-pi/Documents/2017/Legal_Briefing_e_bill_of_Lading_WEB.pdf
Wüst, K., & Gervais, A. (2017). Do you need a Blockchain? *IACR Cryptology EPrint Archive*, 1–7.
Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., & Chen, S. (2016). The blockchain as a software connector. In *Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA* (pp. 182–191). https://doi.org/10.1109/WICSA.2016.21
Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., … Rimba, P. (2017). A Taxonomy of Blockchain-Based Systems for Architecture Design. In *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA* (pp. 243–252). https://doi.org/10.1109/ICSA.2017.33
Zhang, K., & Jacobsen, H. (2018). Towards Dependable , Scalable , and Pervasive Distributed Ledgers with Blockchains. In *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)* (pp. 3808–3813).
Zheng, Z., Xie, S., Dai, H. N., & Wang, H. (2016). Blockchain challenges and opportunities: A survey. *Int. J. Web and Grid Services Blockchain*, 9. Retrieved from http://inpluslab.sysu.edu.cn/files/blockchain/blockchain.pdf

Appendix I – Minutes expert interviews

Appendix I contains the minutes of the interviews held with experts that were used to evaluate the relevance of the use cases and technical correctness of the blockchain-based architecture design. A more detailed summary of the interviews is available upon request at the author of the thesis.

Minutes interview 2: relevance use case III
Type: Teleconference
Date: 25th June 2018
Interviewee: Maersk Line representative involved in Global Trade Digitization (GTD) Platform
Other participants: Yao-Hua Tan

The interview starts off with some discussion on electronic negotiable B/L initiatives, both current and blockchain-based. The most prominent current initiative is Bolero. Bolero requires legal buy-in from trade parties and works closely together with banks. The main trust assumption is that a trade party gives the B/L to Bolero and deletes its own B/L. Bolero then creates a B/L on the Bolero system.

Next, the blockchain-based initiative Wave is mentioned. The goal of the Wave initiative is to integrate directly into a data pipeline. The CargoX initiative follows a similar approach to Bolero which is to aim specifically at electronic negotiable B/Ls. The current line of thinking is that tokens representing B/Ls are not needed in case of an integrated initiative like the GTD platform.

The question is posed that if a B/L is not represented as token in the blockchain, does this require some form of central registry on the GTD platform? The final technical structure of B/Ls on the GTD platform is still subject to discussion. As the Hyperledger platform is used, the registry will in principal be distributed of nature. A suggestion is made to create a centralized registry which is distributed to all other nodes.

The GTD platform aims to provide full digitization and integration of all processes and trade data in a trade lane. For example, Maersk Line information systems can be connected to GTD to allow for direct carrier-related data (e.g., to create a B/L) sharing within the GTD platform. This integration can support buy-in from trade parties to use standardized B/Ls within GTD. For parties that do not buy-in to the GTD platform for all trade data and processes, some form of P2P B/L transfer is required.

The GTD platform follows a dual/hybrid approach. The first step is to digitize current processes without changing these processes. This includes digitization of existing trade documents. For example, an electronic B/L. A second step is to change processes. To fulfill this step, it is important to know what parties really need to manage title to the goods. For example, do they really need a traditional document, or can other means suffice. This can lead to new insights how title management and claim procedures can be simplified.

A question is posed regarding the role of standards. Negotiable B/Ls are used less and less in international trade because paper-based negotiable B/Ls are a 'nightmare'. As long as an electronic negotiable B/L is cheap and has endorsement from P&I, this will lead to buy-in from trade parties.

A question is posed how the GTD platform interprets legislation regarding electronic negotiable B/Ls during development. Current international law does not prohibit the use of electronic negotiable B/Ls. The law provides sufficient flexibility. However, current law has low legal threshold for parties to buy-in. Therefore, some legal framework similar to the Bolero Rule Book is required. P&I has to endorse this framework. As long as P&I endorses, this should convince trade parties to buy into the GTD platform. The Rotterdam Rules and UNCITRAL Model Law provides a good basis for the development of a solution for an electronic negotiable B/L.

Minutes interview III: Evaluation blockchain-based architecture design
Type: Meeting face-to-face
Date: 2nd July 2018
Interviewees: IBM representative involved in GTD platform,
IBM representative part of BeNeLux Blockchain Practice
Other participants: Yao-hua Tan, Boriana Rukanova

Interview conducted in Dutch, minutes translated into English

The interview starts off with a general introduction of the electronic negotiable B/L use case and demonstration of a smart contract B/L (s-B/L) within the blockchain-based architecture design. Interviewees agree that adding the current owner to the state of the s-B/L should be sufficient to determine current ownership and provide exclusive control over transfer. This is similar to other documents that are shared using blockchain technology.

The use of a distributed title registry is discussed shortly. The main benefit of this solution is that no central party manages the registry. The added value in terms of automation is unclear. The advantage of a s-B/L is that transfer of ownership and payment (potentially under L/C) can be automated. If sufficient funds are sent to the contract, the contract automatically transfers ownership to the new holder that sent the funds.

A question is posed how a carrier can be guaranteed uniqueness of a presented B/L if multiple blockchain platforms are used. Since the carrier issues the B/L on a specific platform, it knows which platform is used for which B/L. When a B/L is presented, the carrier would check its own registry to verify that the presented B/L is on the correct platform, under the assumption that there is no interoperability between platforms.

An example case in Peru in which a paper-based B/L disturbs the process of issuing a L/C is discussed to stress the need for paperless B/Ls. A lengthy discussion is held on how a s-B/L and s-L/C can be used to simplify the process. An issue that arises is how banks that issue L/Cs can know if a s-B/L is already used as collateral for another L/C. One potential option would be to add to the s-B/L state if the contract is currently used as collateral under a L/C if assumed that the L/C procedure is not integrated into GTD.

To enable the use of pseudonym identities, in Hyperledger (and thus GTD) trade parties first register to be issued a public key certificate. Using this certificate, a party can request one-time use transaction certificates for each transaction. These certificates are derived from the public key certificate, but only the certificate authority can link both certificates.

Currently, GTD does not aim at implementing all use cases itself. Instead, it seeks partnership with existing initiatives. For example, essDOCS provides a paperless trade solution that from a technical perspective works fine. However, the difficulty is the legal aspect. It can take up to two years for a bank to be 'on-board' with essDOCS. Since international law on electronic trade documents is non-existing, legal agreements are needed. This is one of the reasons for partnerships as these agreements can be used. It is however unclear what the legal implications for GTD are if for example the essDOCS system would be integrated.

Automation (via smart contracts) uses event triggers. This requires standardization of events, especially regarding legal status. Some event triggers already exist implicitly. For example, Incoterms already specifies from a legal perspective what actions should be taken at which moment. Event triggers can be based on these Incoterms.

Channels in Hyperledger can be used to represent trade lanes, only providing access to parties involved in a trade lane. Each channel has its own blockchain ledger. Currently, there is a limit of 10,000 channels. GTD provides templates to configure channels. These templates are used to specify which parties can execute which actions (upload documents, add events) to a channel. Incoterms that are agreed upon at the start of a shipment can be the basis for these templates. Within a channel, encryption on individual data elements ensures that only authorized trade parties can access data.