



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

EXECUTIVE SUMMARY OF THE THESIS

From business process to Corda R3: enforcing privacy and security of smart contracts

LAUREA MAGISTRALE IN COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA INFORMATICA

Author: ALESSANDRO DI RENZO

Advisor: PROF. MATTIA SALNITRI

Co-advisor: PROF. GIOVANNI MERONI

Academic year: 2022-2023

1. Introduction

In recent years, several studies have highlighted limitations related to contracts edited manually which are characterized by high costs, time-consuming and contentious points subject to personal interpretation. In order to address this aspect, smart contracts have been introduced, i.e., automated computer programs that verify and execute contract terms based on defined conditions. They are faster and more efficient compared to traditional contracts.

Smart contracts define the conditions to be met in an agreement by specifying the activities that participants have to perform to fulfill the agreement. Due to this aspect, contracts can be seen as processes: for instance, a smart contract that regulates the actions a patient must take to book a medical appointment and subsequently the ticket opened by the healthcare provider. In this context, ensuring a certain level of security in the formulation of smart contracts assumes a primary role.

Security properties such as data confidentiality and enforceability of decisions must be considered for reaching secure contracts.

For instance, a contract that regulates an analysis laboratory and a hospital structure where

the results of patients have to be protected from unauthorized access. On the other hand, some decisions taken by specific entities need to be approved by other participants: for instance, a public administration tender where competitors that compete to provide the best offer should have the opportunity to validate the final decision of the winning competitor.

Processes can be represented with several graphical modeling languages as BPMN 2.0 standard. Since contracts can be seen as processes, the same modeling language can be used to model smart contracts. BPMN 2.0 standard needs to be enriched with security properties to represent security concepts. We adopted the extension proposed by Köpke et al. in [2] which defines smart contracts as processes with security notations.

One of the technologies on which smart contracts are implemented and executed is the blockchain which ensures immutability and transparency to the information stored on the ledger. It can be used as a security mechanism for the enforcement of security properties.

In this scenario, the blockchain replicates the context of business processes by ensuring a certain level of security regarding information ex-

changed by design: for these reasons, the choice has landed on the Corda blockchain, a distributed ledger developed by R3 that offers the possibility of realizing a private network and manages the process through a smart contract that regulates single steps of the execution based on the interactions among parties that characterized the process.

This thesis exploits Corda for the enforcement of two security properties related to smart contracts for modeling business processes: privacy of data and enforceability of decisions. The method proposed includes sequential phases:

1. contracts are initially represented with SecBPMN2BC;
2. mapping from BPMN Collaboration to BPMN Choreography and extension of choreography diagram to represent security properties;
3. mapping and transformation from choreography diagram into Corda contracts following the conceptual model and the rules of translation conceived, while guaranteeing the enforcement of security aspects;
4. testing phase, where the effectiveness of the model proposed is evaluated through business process realistic cases.

2. State of the art

We consider a research work that extends standards including security aspects. Vivas et al. in [4] propose a UML-based business process-driven framework for the development of security-critical systems that shows possible threats related to the trade-off between security and functionality when defining security requirements for a system. A similar approach is suggested by the work of Lodderstedt et al. in [3] is aimed to define a modeling language for the development of secure systems with UML through role-based access control. It describes how to use UML to specify security requirements in modern systems that are well-suited only for static design models. This system language is based on an extended model for role-based access control (RBAC), but to overcome lack of methodology they introduced the authorization constraints, a precondition for granting access to an operation, they defined such limit using the Object Constraint Language (OCL). SecureUML is a combination of the main features of these two

systems. As visible in these works, security requirements are defined also using UML as a standard, which is different from the standard that we have used in our work, i.e., BPMN, but it provides a methodology to enrich already existing standards with additional requirements. The enforcement of decisions is discussed in the research work conducted by Haarman et al. in [1], where the approach detailed is aimed at the enforcement of decisions in a collaborative process executed by a smart contract on Ethereum blockchain. The approach is composed of two phases: the operation phase, where the decision is executed locally with an agreement of participants regarding input and data consumed to take the decision; the conflict resolution phase, where the agreement is not reached on the output of the decision and the conflict is solved by the smart contract itself at cost of revealing the logic under that decision. The purpose was related to the overcoming of limitations of exposing data of decisions in public blockchains with a condition, i.e., the conflict on the output, upon which the confidentiality of data is violated. Referring to our work, this kind of solution proposed is not applied, since the enforcement of properties related to decisions and data have to be preserved in all cases.

3. Baseline

3.1. SecBPMN2BC

Our work focuses on enforcing security properties, particularly data confidentiality and decision enforceability, using the model-driven approach proposed by Köpke et al. in [2]. This language extends BPMN 2.0 with security notations for secure smart contracts in blockchain. Smart contracts are depicted as business processes with security requirements.

The modeling language defines privacy spheres for data objects and messages to model read access restrictions: spheres are categorized from strong-dynamic to global which define participants that can access data objects. Moreover, it introduces annotations to specify enforceability levels for decisions, attached to exclusive gateways. Three sets of validators are defined: public, private, and user-defined, determining the required participants for decision verification. The levels of both security properties are re-

ported in Figure 1 and Figure 2 .




Symbol	Level of enforceability of decisions
	Public
	Private
	User-defined

Figure 1: Security property of enforceability of decisions from [2]






Symbol	Level of privacy
	Public
	Private
	Static
	Weak-dynamic
	Strong-dynamic

Figure 2: Security property of privacy from [2]

3.2. Running example

A contract example, expressed with SECBPMN2BC, involves a Company Employee, a Health Care Fund, and a Medical Office. It describes the Company Employee's steps to request an affiliated visit, the Health Care Fund's response and successive actions in case of a positive or negative answer. Initially, the contract specifies the Company

Employee's actions in order to send the request for an affiliated visit. The Health Care Fund's response leads to two scenarios: acceptance or rejection. If accepted, the Medical Office handles data exchange and booking dates. On the other hand, the flow stops. Data objects are related to the pathology documentation of Company Employee, marked with a level of privacy of private type, and available dates of Medical Office to book the visit associated with a strong-dynamic level. Regarding the exclusive gateway, it is related to the compliance of request from Company Employee and the decision is demanded by Health Care Fund. This scenario is the starting point of our method.

3.3. BPMN Choreography

BPMN Choreography is a modeling language that illustrates interactions in business processes among multiple participants through message exchange, emphasizing collaboration to achieve shared goals.

A Choreography diagram includes tasks, messages, and gateways to represent the process. Choreography tasks refer to actions triggered by incoming and outgoing messages from participants. Tasks are connected by sequence flows that define the order of activities. Each task presents an initiator and at least one receiver. Messages represent information exchanged asynchronously between participants and can encompass various data types, requests, or notifications. Gateways make decisions or create alternative or parallel paths in the process. Exclusive gateways rely on data from prior messages, with initiators responsible for decisions. Parallel gateways allow parallel task execution. In this thesis, the focus is on exclusive gateways.

3.4. Corda

Corda is a permissioned blockchain developed by R3 for enterprises. It's a decentralized, open-source ledger tech emphasizing secure and private transactions. Corda exploits smart contracts, i.e., CorDapps, to manage data interactions, prioritizes interoperability, and aims for trust, efficiency, and transparency. The network generated is private, with nodes interacting on a "need-to-know" basis. Participants include individuals, organizations, and automated systems, each with unique cryptographic identities. They

manage shared facts, propose transactions, and validate the shared ledger.

States are shared facts among participants and each node has its own vault of shared states. Transactions modify the ledger by consuming input states and producing output states. Contracts define state behavior, ensuring validity and compliance with predefined rules in transactions. Commands express transactions' intent, specifying required signers and the rules that input and output states have to follow to evolve the state. Notaries validate transactions, ensuring uniqueness and authenticity, marking input states as historic to prevent double-spending. Flows manage contract state evolution, coordinating parties asynchronously.

CorDapps are distributed applications for Corda developed in Java.

4. Method definition

Our work focuses on ensuring security properties like decision enforceability and data privacy by creating contracts in the Corda blockchain. This approach is detailed in Section 5 and Section 6 and represents our primary contribution. The method follows these stages:

1. Initially, we adapt collaboration to choreography, emphasizing interactions between participants. After that, we incorporate security properties and data objects into the diagram. Lastly, we remove security notations and add choreography tasks representing security properties.
2. The second stage involves two phases. First, we define a conceptual mapping between choreography and contracts, specifying how each element in the diagram corresponds to a Corda contract element. In the second phase, we define transformation rules, i.e., algorithmic procedures that automate element creation, names, and attributes.
3. In the final stage we implement in Java what the algorithms produced and enrich them with missing parts in order to reach a deployable CorDapp.

This method doesn't aim for fully automated CorDapp generation and execution. Human input is required to adapt the CorDapp skeleton with information related to possible files or textual messages in the contract.

4.1. Extended BPMN Choreography

BPMN2.0 Choreography lacks representation of data objects and security properties, so we integrated security notations from collaboration using the SecBPMN2BC approach. Regarding the enforceability of decisions, we retained the notation from the collaboration diagram, preserving its original meaning. This was possible because the participants in the choreography diagram remained the same as in collaboration, ensuring the set of validators for decisions. Exceptions to this rule applied when enforcing decisions at a public level, where we transformed them into private decisions involving all contract participants, as Corda doesn't permit external participants.

For data objects, we represented them by using the same graphical notation. We selected tasks to represent data objects while preserving the initiating participant and the order in which other participants access data in the following tasks.

Our goal is to adapt and model the diagram to ensure the representation of security properties, even though not all situations may be replicable due to the characteristics of the contract. An example of the application of the method is reported in Figure 3.

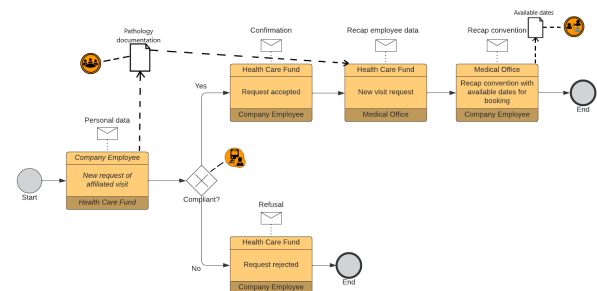


Figure 3: Choreography diagram enriched with security properties

Since the choreography diagram doesn't allow for this type of representation to be valid, we need to include in the choreography tasks the security properties added and the enforcement of the security properties.

From choreography diagram rules, the decision is taken by the initiator of the first task after the gateway. Regarding the enforceability of decisions, participants who act as validators of a specific decision must own the data upon which

the decision is made in order to validate it. Validators can be divided into two categories. The first category comprises receivers of tasks immediately following gateways since they already know the decision-making data. Validators in this category are visible from the choreography diagram. The second category includes validators that are not present in the tasks immediately following the gateway, but they are required to validate the decision from the set of validators specified by the security property associated. There are two subcategories within this group: those who already have the data and can proceed with the validation, i.e., second category without priority, and those who lack it, i.e., second category with priority. Validators of the second category with priority require an additional task before the gateway to own the necessary data: this task is named "Decision_SharingData". Validation for validators of second category are represented with additional tasks before the one of validators of first category, both started by the common initiator who takes the decision.

Data objects can be managed as message content in BPMN Choreography. In order to enforce the privacy property, an additional task containing the data object as part of the message follows the task where the data object first enters the diagram: in this case, the task is named "Privity_SharingData". The participants of this task are based on the data object's associated privacy sphere, so that participants who have access to the data will own it after the execution of the task. To handle both data objects and decision enforcement properties, data objects are enforced immediately. Subsequently, exclusive gateways and decisions enforcement are assessed. An example of enforcement of security properties is reported in Figure 4.

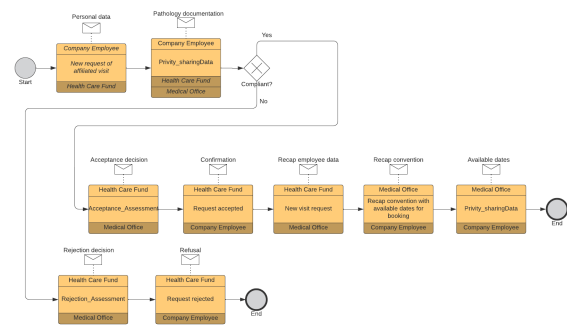


Figure 4: Choreography diagram generated from extended choreography diagram in Figure 3

5. From Collaboration to Choreography

The first mapping is from collaboration to choreography diagram. The objects are mapped based on the following guidelines.

- **Participant identification:** pools and lanes from the collaboration diagram are analyzed to identify all participants in the choreography diagram, considering lanes as distinct participants.
- **Inclusion of start/end events:** the choreography diagram includes start and end events, with intermediate ones added if they are connected to messages in the collaboration diagram.
- **Message events to choreography tasks:** messages exchanged in the collaboration diagram become choreography tasks. Additional tasks may be created to adhere to choreography diagram rules.
- **Initiator and receivers:** connection lines between sender and receiver of messages in the collaboration diagram define task initiators and receivers in the choreography diagram.
- **Tasks not mapped:** some collaboration tasks may not be reported due to the different granularity levels between collaboration and choreography diagram.
- **Exclusive gateways:** exclusive gateways, i.e., gateways related to decisions are certainly mapped with added security property of enforceability of decisions.
- **Other gateways not mapped:** other types of gateways, like parallel gateways and those linked to internal activities, are not mapped unless they are essential for

representing contract constraints.

- **Sequence flow preservation:** flow sequence, indicating task order, is conserved in the choreography diagram, with arrows connecting tasks and giving the sequential order to the entire diagram.
- **Data objects:** data objects present in the collaboration diagram aren't directly represented in the choreography diagram due to limitations and uncertainties.

This transformation raises some implications:

1. There is no unique way to map a collaboration diagram into a choreography one: it depends on the emphasis you want to place on interactions. Flexibility is the key aspect to preserve the contract's meaning.
2. Collaboration diagrams are characterized by a lower level of granularity, detailing individual activities of participants. On the other hand, choreography diagrams emphasize interactions between participants. For this reason, not all collaboration details can be reproduced with the same granularity in the choreography diagram.
3. Gateways not linked to decisions aren't included in the mapping. Since the focus is on enforcing specific security property related to decisions, the other types of gateways are excluded if they aren't of main importance for contract constraints.
4. The choreography diagram resulting after this phase emphasizes the flow and meaning of the contract, rather than focusing on the representation of internal organizational tasks. The focus is on respecting the sequence and general sense of the contract.

An example of the mapping related to the running example is reported in Figure 5.

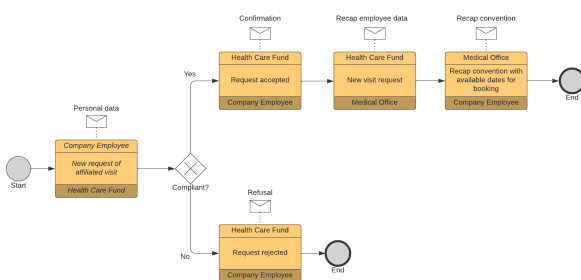


Figure 5: Choreography diagram of affiliated medical visit

Data objects and security properties are then included in the diagram as described in Section 4.1.

6. From Choreography to Corda

The second mapping is from choreography diagram to Corda: the matching between the choreography elements and Corda objects is reported below.

State and contract

States represent the evolution of facts shared among parties in the Corda network, while contracts regulate the behavior and validity of transactions involving states. Choreography diagrams represent the process evolution and the name of the diagram is taken to map to Corda's state and contract pair.

Nodes

Choreography diagram participants may include companies, public infrastructures, and individuals: each participant is represented by a node in the Corda network.

Workflows

Choreography tasks depict interactions among participants, which is strictly related to workflows that create transactions for generating new states and sharing them among participants. For this reason, the mapping is realized between choreography tasks and workflows with related commands that regulate the transaction.

Sequential flow execution

In a choreography diagram, tasks follow a sequential order described by arrows that link consecutive tasks. In order to define the sequential evolution of the state, the input state in a transaction matches the output state of the previous transaction, except for the first task, so that it is possible to provide a sequentiality to the transactions.

Messages representation

Choreography messages can be textual data or files. Corda allows messages to be represented either as attributes of the state, in the case of textual messages, or as attachments to workflow transactions.

Gateways representation

Exclusive gateways in the choreography diagram represent choices related to decisions taken by participants and different paths based on the fi-

nal choice. In order to map this scenario, specific attributes in the state indicate all available paths after the gateway.

The transformation is performed following the Algorithm 1. The lines with the signature "(Exp)" require the action of an expert in order to realize a deployable CorDapp. All the other lines refer to algorithms that automatize the generation of such elements in Corda.

Algorithm 1 Choreography to Corda

- 1: State and contract name definition
 - 2: (Exp) Define basic attributes of state
 - 3: Add nodes in Corda network
 - 4: Workflows and state attributes
 - 5: (Exp) Retrieve Notary information
 - 6: **if** Not First Workflow **then**
 - 7: (Exp) Retrieve input state from the vault
 - 8: **end if**
 - 9: (Exp) Complete transaction requirements
 - 10: Commands name definition
 - 11: Commands rules definition
 - 12: (Exp) Specify complementary rules
 - 13: Messages representation
 - 14: **if** Attachment in transaction **then**
 - 15: (Exp) Add specific methods for the upload of files
 - 16: **end if**
 - 17: Exclusive gateways representation
-

7. Validation

In the validation phase, we evaluated each step of the method proposed through two main phases, which combined validate the method. The first one validates the mapping and transformation from collaboration to choreography diagram and aims to validate the correctness in terms of participants inclusion, interactions representation and presence of security elements. The second one validates the mapping and transformation from choreography diagram to the skeleton of CorDapp by defining nodes, state, contract and workflow elements with associated rules.

In order to carry out the required transformations, we have implemented a software that automates part of the process. Specifically, the software takes as input an XML document containing the interactions and the participants of the collaboration diagram, thus it generates an XML document containing the choreography

tasks and eventual gateways that are part of the choreography diagram. After enriching the XML of the choreography diagram with tags related to security properties, the software takes this XML as input and generates the additional tasks that enable the enforcement of security properties. The XML document representing the choreography diagram inclusive of security properties marks the end of the first validation phase. Regarding the second phase, the software takes as input the XML generated in the previous phase and generates the Java classes that represent the skeleton of the CorDapp. The generated classes include a state that evolves over time with its attributes, a contract that governs its evolution and workflows that create transactions to consume and generate new states."

Examples chosen to test the method refer to realistic cases that describe situations like the electronic emission of a new birth certificate, a hospital televisit performed by an external group or by an internal doctor.

After critically analyzing the results obtained in the two phases, the contracts align with the requirements and guarantee the enforcement of security properties. Despite this, there are some considerations: the mapping from collaboration to choreography diagrams is challenging and errors in this phase may impact the final CorDapp; the number of messages, exclusive gateways, and data objects impact choreography tasks count and CorDapp state attributes; attributes of the output state generated in the transactions are with defaults values of attributes and need to be changed in order to be adapted to the evolution of the state. The scalability limitation may necessitate complementary techniques for representing state evolution in complex contracts.

8. Conclusions

This thesis discusses a method for the enforcement of security properties of smart contracts for modeling processes such as confidentiality of data, i.e., privacy, and enforceability of decisions. The method, after receiving the collaboration diagram modeled with SecBPMN2BC, is composed of the following phases: the first mapping that shows how to transform a collaboration into a choreography diagram based on the interactions among participants and the security properties depicted in the starting diagram; the

second mapping and transformation from choreography diagram to Corda that shows how to realize the skeleton of a CorDapp, with the definition of the main objects such as state, contract and workflows.

We implemented a software that automatizes part of the process: from collaboration to choreography diagram with a semi-automatic procedure and then from choreography to the skeleton of a CorDapp. The software represents the application of the conceptual mappings detailed in the previous sections.

We tested it through three realistic cases. Results showed that the method proposed can be used in order to enforce the security properties mentioned above, with particular attention to the first mapping from collaboration to choreography diagram: possible errors are propagated in the next phase and compromise the effectiveness of the contract. Regarding the second phase, the attention is related to the values of attributes in the output states generated by transactions that may not respect the desired evolution of the state.

Certainly, it needs to be tested further with a greater pool of samples and with external reviewers. Furthermore, some of the activities of mapping and transformation of diagrams, in order to provide an accepted input to the method, have been performed manually with the presence of experts. For future works, the implementation of the method may include fully automated software that, given a contract described with SecBPMN2BC, produces a complete CorDapp as output ready to be applied in real business contexts.

References

- [1] Stephan Haarmann, Kimon Batoulis, Adriatik Nikaj, and Mathias Weske. Executing collaborative decisions confidentially on blockchains. In *Business Process Management: Blockchain and Central and Eastern Europe Forum: BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1–6, 2019, Proceedings 17*, pages 119–135. Springer, 2019.
- [2] Julius Köpke, Giovanni Meroni, and Mattia Salnitri. Designing secure business processes for blockchains with secbpmn2bc. *Future Generation Computer Systems*, 141:382–398, 2023.
- [3] Torsten Lodderstedt, David Basin, and Jürgen Doser. Secureuml: A uml-based modeling language for model-driven security. In *UML 2002—The Unified Modeling Language: Model Engineering, Concepts, and Tools 5th International Conference Dresden, Germany, September 30–October 4, 2002 Proceedings*, pages 426–441. Springer, 2002.
- [4] José L Vivas, José A Montenegro, and Javier López. Towards a business process-driven framework for security engineering with the uml. In *Information Security: 6th International Conference, ISC 2003, Bristol, UK, October 1–3, 2003. Proceedings 6*, pages 381–395. Springer, 2003.