Towards trustworthy, flexible, and privacy-preserving peer-to-peer business process management systems

Tiphaine Henry

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAS024

Thèse de doctorat

Towards trustworthy, flexible, and privacy-preserving peer-to-peer business process management systems

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 16/11/2022, par

TIPHAINE HENRY

Composition du Jury :

Professor Chirine Ghedira-Guegan (PhD) Univ. Lyon, Université Jean Moulin Lyon 3, LIRIS, UMR5205, IAE Lyon School of Management, France	Présidente
Assoc. Professor Alexander Norta (PhD) Department of Software Science, School of Information Technologies, Tallinn University of Technology, Estonia	Rapporteur
Professor Thomas Hildebrandt (PhD) Computer Science Department, University of Copenhagen, Denmark	Rapporteur
Sara Tucci Pergiovanni (PhD) Head of Laboratory, CEA, France	Examinateur
Professor Joaquin Garcia (PhD) Télécom SudParis, Institut Polytechnique de Paris, Samovar, France	Examinateur
Emmanuel Bertin (PhD) Research Scientist, Orange Innovation	Examinateur
Professor Walid Gaaloul (PhD) Télécom SudParis, Institut Polytechnique de Paris, Samovar, France	Directeur de thèse
Nassim Laga (PhD) Research Project Leader, Orange Innovation, France	Encadrant
Julien Hatin (PhD) Research Project Leader, Orange Innovation, France	Invité

Acknowledgment

Mes premiers remerciements vont à mes encadrants : les docteurs Nassim Laga et Julien Hatin, et le professeur Walid Gaaloul. Le sujet de thèse proposé est à la fois passionnant sur le plan théorique et précurseur d'applications industrielles. Je les remercie de leur bienveillance, de leurs encouragements, et de la veille qu'ils ont mise en œuvre pour que cette thèse soit une réussite.

Je remercie chaleureusement le professeur Alexander Norta, le professeur Thomas Hildebrandt, le professeur Chirine Ghedira-Guegan, le docteur Sara Tucci-Pergiovanni, le professeur Joaquin Garcia, et le docteur Emmanuel Bertin d'avoir accepté de participer à mon jury de thèse.

Je remercie également Orange de m'avoir donné l'opportunité de réaliser ce travail, ainsi que tous les moyens nécessaires pour son bon déroulement. Je remercie Nassim Laga et Philippe Legay, de m'avoir fait confiance et de m'avoir permis de réaliser cette thèse. Je remercie également Damien Lannelonque et Xavier Loir de m'avoir accueilli dans leurs équipes durant la thèse.

Ma gratitude va aussi à tous les gens qui ont participé de près ou de loin à la réalisation de ce travail, que ce soit de façon informelle dans les couloirs et les pauses café, ou formelle au travers de travaux de recherches, de séminaires et de stages. Je remercie plus particulièrement Amina Brahem, Denis Perrin, Paul Amsellem, Drissa Houatra, Oumaima Alaoui Ismaili, Rémy Scholler, Jérôme Dupont, Shenle Pan, Roman Beck, Blaise Carnevillier, Léo Kazmierczak, et Eloi Besnard, pour les nombreux échanges fructueux que nous avons eus. Je remercie également les membres de l'équipe SAMOVAR de l'Institut Telecom Sudparis. Leurs commentaires, suggestions, ou tout simplement sympathie ont largement contribué au succès de cette thèse.

Enfin, je remercie Sara Tucci-Pergiovanni, et tous les membres de ma nouvelle équipe pour leur sympathie et leur accueil dans mes nouvelles fonctions au sein du CEA List.

Merci enfin à mes proches pour leur soutien au long de ces trois ans : mes parents et ma soeur Astrid, qui ont été non seulement patients mais également les meilleurs supporters tout au long de ces années d'études, mon compagnon Antoine, pour son humour et sa confiance, sans qui je ne serais pas là où je suis aujourd'hui, ainsi que ma famille et mes amis, pour leur présence bienveillante et constante. Cette thèse leur est dédiée.

Abstract

Blockchain technology has been introduced as a trustworthy disintermediation tool for managing cross-organizational business processes in the past decade. It ensures activities' execution traceability while enforcing the control flow agreed upon at design time with other partners.

However, ensuring trust in the deployment and execution protocol remain challenging in this environment. Indeed, a separation of concerns should be ensured among participants: the latter should only have access to the status of the public tasks they are concerned with. Additionally, cross-organizational processes comprise public and private activities, and a mechanism ensuring the connection between the these two views should be defined. Another challenge concerns business process management systems flexibility. Cross-organizational processes are dynamic by nature, hence both control-flow and runtime resources allocation flexibility should be made accessible in a blockchain-based process management system. Finally, we consider the data privacy challenge. Data processed in the blockchain environment can be accessed by blockchain participants. Hence, the trustworthy auditability of the blockchain-based process management systems should be backed with a confidential treatment of sensitive business data.

To address these challenges, we propose the three following contributions in this manuscript. First, we design and implement an on/off-chain deployment and execution strategy for on/off-chain choreographies, which enforces a trustworthy separation of concern between participants at each step of the deployment and execution. Second, we propose to bring control-flow flexibility to the blockchain-based business process management system through change management: a change impacting other partners is propagated to affected processes using a smart contract. We also leverage smart contracts for a dynamic selection of service providers. Finally, we propose two mechanisms to reconcile privacy imperatives with the benefits of blockchain. The first mechanism leverages fully homomorphic encryption for blockchain-based calculations such as sealed-bid auctions. The second mechanism leverages banks as trustworthy intermediaries while secreting the payment value. We demonstrate the feasibility of each contribution through an implemented prototype and its effectiveness via experiments anchored in the logistics domain.

Contents

List of Figures

List of Tables

List of Symbols

DCR graphs symbols

R	Set of roles
r, r'	Roles
G	DCR graph
G_r	DCR choreography private view (projection) of role r
G_γ	DCR choreography public view
Φ	A DCR choreography
E	Set of labeled events
e, e'	Two events
id_e	Index of activity e in the relation matrix
ϵ	Set of internal events in G
i	An interaction
I, γ	Set of interactions
M	Set of markings
m_e	Marking of event e
in	Included event state
pe	Pending event state
ex	Executed event state
In	Set of currently included events
Pe	Set of currently pending responses
Ex	Set of previously executed events
f	Labelling function
L	Set of labels
Γ	Set of relations of the graph
l	A relation between two events
$\longrightarrow \bullet$	Condition relation
$\bullet \longrightarrow$	Response relation
$\longrightarrow \diamond$	Milestone relation
$\longrightarrow +$	Include relation
$\longrightarrow \%$	Exclude relation
σ	counter keeping track of the number of projections realized

$\gamma_{approval}$	List recording whether a participant has generated its local projection
γ_{fetch}	List recording whether the participant fetched the public view

Business Process Model change symbols

ν	A change
$R_{endorsers}$	Set of change endorsers
G_{Ref}	Change element (also called refinement element)
G'_{Ref}	Updated change element
Q	A business process model fragment
$t1$	the deadline timestamp for change endorsement
$t2$	the deadline timestamp for change propagation
δ	the endorser response $\in \{0, 1\}$
$\gamma_{requests}$	List of change requests
$\gamma_{endorsement}$	List of change endorsement responses
γ_{propag}	List of partners' change propagation states

Encryption symbols

ψ_x	Hash of x
c, c_{max}	Ciphered numbers
c_x, c'_x	Ciphers of x
C_, D_	Ciphering function of type _
k_	Ciphering key for the symmetric encryption of type _
k'_	Ciphered Ciphering key for the symmetric encryption of type _
$(s_\_, p_\_)$	Secret and public keys for the asymmetric encryption algorithm of type _
O_{enc}	Array of numbers ciphered with an algorithm of type _
L	List of ciphered aggregated offers
L_{dec}	Deciphered argmax vector
Ψ	A hash function
θ_1, θ_2, θ_3	Values to be hashed
ρ_l, ρ_r	Left and right branches of Merkle tree

Blockchain/IPFS symbols

V_{token}	Token smart contract
V_{DCR}	Smart contract managing the DCR choreography

a_V	public address of smart contract V
a_{role}	Blockchain address of partner *role*
a_{sender}	Address of the sender of the blockchain transaction
$a_{endorser}$	the endorser address
$A_{endorsers}$	the list of endorser addresses
ψ_G	the current IPFS workflow hash
$\psi_{G'}$	the IPFS hash of the requested change description
q	Token price for a service
t	Target block to call a smart contract
N	Total Number of tokens to generate
h	Payment channel
H	List of payment channels
α	Payment token conversion rate
m	Number of tokens

Miscellaneous

A_1, A_2, A_3, A_4	Four carriers
D_1, D_2	Availability dates
O_{A1}, O_{A2}	Vector of offers of A_1 and A_2
ϕ	A toy example location
x, y, n	Natural numbers
p	a prime number
g	an algebraic operation
β	penalty factor
d	Claim time
P_F	List of filtering criteria
P_O	List of QoS optimization criteria
W	List of optimization weights where each weight $\in$ [0,1]
S	Service request
U	Candidates matrix
η_r	Quality of service rating of role r
$P_{F_{availability}}$	List of candidates' availability dates
ω	A boolean variable assessing candidates' matching status
X	Binary filter vector
B	binary argmax vector
$\Pi_{O_{A1}}, \Pi_{O_{A2}}$	interchanged offers

List of Acronyms

RSA	Rivest Shamir Adleman
AES	Advanced Encryption Standard
FHE	Fully Homomorphic Encryption
T	Token unit
SP	Service Provider
SC	Smart Contract
IPFS	Inter Planetary File System
BPMN	Business Process Model and Notation
DCR	Dynamic-Condition-Response
QoS	Quality of Service
FTSP	Freight Transportation Procurement Process
CMR (or CMR Convention)	Convention on the Contract for the International Carriage of Goods by Road
AR	Allowed Change Rule
DR	Denied Change Rule
API	Application Programming Interface
DoS	Denial of Service
DSS	Digital Signature Scheme
ECDSA	Elliptic Curve Digital Signature Algorithm
ECDH	Elliptic Curve Diffie-Hellman
DSA	Digital Signature Algorithm
PBFT	Practical Byzantine Fault Tolerance consensus mechanism
UTXO	Unspent transaction output
TEE	Trusted execution environment
VDF	Verifiable delay function
EVM	Ethereum Virtual Machine
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ASIC	Application Specific Integrated Circuit
DAG	Directed Acyclic Graph

List of Publications

This Ph.D. thesis is based on the following publications that are referred to in the text by Roman numbers.

I HICSS 2021: Hawaii, USA, *Cross-collaboration processes based on blockchain and IoT: a survey*, T.Henry, N.Laga, J.Hatin, W.Gaaloul, I.Boughzala

II ICSOC 2021, Dubai, UAE, *Trustworthy Decentralized Execution of Declarative Business Process Choreographies*, T.Henry, A.Brahem, N.Laga, J.Hatin, W.Gaaloul, B.Benatallah

III IEEE SCC 2021, online session, *Hire me fairly: Towards dynamic resource-binding with smart contracts*, T.Henry, N.Laga, R.Beck, W.Gaaloul

IV HICSS 2022, Hawaii, USA, *Decentralized procurement mechanisms for efficient logistics services mapping - a design science research approach*, T.Henry, R.Beck, N.Laga, W.Gaaloul, S.Pan

V BPM 2022, Münster, Germany, *A trustworthy decentralized change propagation mechanism for declarative choreographies*, A.Brahem, T.Henry, S.Bhiri, T.Devogele, N. Laga, N. Messai, Y.Sam, W. Gaaloul, B. Benatallah

VI ICSOC 2022, Bozen Bolzano, Italy *Random-value tokens for privacy-preserving decentralized payments*, T.Henry, L.Kazmierczak, J.Hatin, E.Bertin, N.Laga, W.Gaaloul

VII *Towards trustworthy and privacy-preserving decentralized auctions*, T.Henry, J.Hatin, N.Laga, W.Gaaloul, submitted to the Journal of Banking and Financial Technology

Two patents were derived from publications IV and VII.

- Patent: n° FR2100122 "*Procédé de fourniture de service mis en œuvre par ordinateur dans une chaîne de blocs, nœud d'un réseau de chaîne de blocs et programme d'ordinateur correspondants.*". T.Henry, N.Laga, 2021

- Patent: n°FR2109204 "*Procédé de fourniture de service mis en œuvre par ordinateur dans une chaîne de blocs, nœud d'un réseau de chaîne de blocs et programme d'ordinateur correspondants*". T.Henry, B. Carnevillier, J.Hatin, N.Laga

Chapter 1

Introduction

Contents

1.1 Research context

A trend toward content decentralization, distribution, and disintermediation can be noted since the early 2000s [174]: service and resource providers communicate and collaborate with clients more directly, with fewer intermediaries. Such habits can be particularly noticed in the industry. Indeed, the digitization and automation of industrial processes, backed by servitization and decentralization trends, has encouraged the development of more flexible and cross-organizational partnerships [134, 137]. Intermediation platforms such as Amazon or Uber are examples of cross-collaboration processes: customers and service providers (e.g., manufacturers or delivery

carriers) interact at different stages of the process to lead to value creation [59]. Some examples of added-value activities are producing items, proceeding to checkout, proceeding to the delivery, etc.

In this cross-organizational setting, contractual trust- the subjective belief that a set of agreements will be fulfilled while respecting a set of constraints (e.g., resource, time, etc.)- is paramount. To better communicate and coordinate, business process models are used: the activities of each partner are specified as a shared diagram, and activities are linked together to be orchestrated in a standardized fashion [221, 92, 128, 197]. The status of each activity is specified and monitored on the business process model instance. Hence, each partner oversees the whole process pipeline and verifies the correct execution of activities at the right time and by the right partner.

Activities managed in a cross-organizational setting can be public (e.g., involving several partners), or private (e.g., part of one partner's internal process). Communication activities aim at better communication and coordination between partners. Nonetheless, public activities take place physically in a decentralized fashion. Hence, challenges arise regarding the multi-party management of such decentralized process.

First, a need for a trusted communication environment arises [157]. A third-party platform often carries out in a centralized silo fashion the management of shared tasks aiming at communication and coordination [21, 228]. The platform embodies trust: for example, it can act as a mediator in the case of a dispute resolution, collect the ratings of the clients, or propose optional insurance. However, these third-party platforms represent added costs for the involved partners. They also create an imbalanced sharing of resources, impede contractual flexibility, and constitute single points of failure that can cause security leakages. Power imbalances regarding access to information may rise as in traditional settings, distrust into provided logs may arise as execution logs, and process execution is provided by a trusted third-party [83, 33]. Hence, a need for leveraging trust in a trustless multiparty environment arises.

Additionally, a need for more efficient execution procedures can be cited, backed by the automation of paperwork activities. Indeed, legal or security-related guidelines can generate delays for document processing activities (cf AML procedures or import-export regulations). Similarly, human factors can also induce activity execution delays (e.g., a collaborator forgets to confirm the reception of a package). Hence, a need to reduce process execution delays through the lens of process automation raises.

Thirdly, a need for more flexible process management can be pointed at. Indeed, business process management systems may prove to be rigid with respect to control-flow executions. Nonetheless, adapting processes to evolving regulation laws, internal process changes, or partnering choices can be cumbersome but remains necessary for enterprises to stay competitive.

Finally, a need for bridging the gap between business and IT is at

stake. The tools and proof-of-concepts currently developed are often too development-oriented. Their manipulation requires coding skills. Consequently, from a business analyst's perspective, three difficulties might arise: a steep learning gap, the threat of committing implementation errors, and security concerns.

In the past decade, blockchain technology has been introduced as a trustworthy disintermediation tool for managing information, and payment [105, 135].

Blockchain can be defined as a decentralized peer-to-peer ledger that keeps track of transactions between users. This is particularly helpful for providing trust, traceability, and security in systems that exchange data or assets." [213]. Validated transactions are stored in blocks, and blocks are linked with the former block's hash following a Merkle tree structure [136]. Consensus algorithms such as proof-of-work, or proof-of-stake, regulate the chain's growth by verifying transactions and discarding invalid transactions. Additionally, cryptography rules ensure the pseudo-anonymity of users. Blockchains today start to be adopted by the banking system and the supply chain [29], smart shipment monitoring [44, 10], land registry implementations in Sweden and Georgia [178, 199], corruption fight in Ukraine [191], or food traceability such as the IBM food trust initiative [97], a consortium of food partners such as Carrefour and the Mousseline mash potatoes traceability in 2019, or the Liebig soup in 2020 [68].

The maturation of blockchain has followed an iterative process. Bitcoin was inspired by early attempts to create an anonymous transaction system, namely Digicash, Hashcash, and B-Money [146]. Satoshi Nakamoto published the Bitcoin whitepaper in October 2008 to summarize these early attempt proposals. The whitepaper proposes an electronic payment system based on cryptographic proof instead of trust in central banks. It proposes proof of work consensus as a trustworthy distributed consensus as one CPU (Central Unit Processing) is worth one vote. Building on this whitepaper, Bitcoin is proposed as a trustworthy pseudonymous cryptocurrency. The genesis block was mined on Jan 3rd, 2009.

The second generation blockchain, released in 2015, is Ethereum, which proposes, on top of the blockchain, smart contracts [31]. Ethereum history begins with the need for a platform that can support a Turing complete code. Ethereum has been developed to offer more than currency exchanges using Turing-complete scripts executed on-chain named smart contracts. With Ethereum smart contracts, any algorithm can theoretically be coded and executed as a smart contract. Additionally, decentralized applications (dApps) can be developed using Ethereum's blockchain as a backend. Users access dApps functionalities by calling smart contract functions instead of traditional API (Application Programming Interface) calls. The smart contract, publicly stored in the ledger, autonomously executes contractual terms without calling any third party. Smart contracts are deployed in the

blockchain ledger and replicated among blockchain participants: they are run by each network node. If all nodes verify all conditions of the smart contract, then the transaction is executed and validated (i.e., appended to the chain).

Alongside smart contracts operate third-party services linking the blockchain network to the outside world, referred to as oracles [4]. Using oracles, smart contracts can carry on API calls to query trustworthy off-chain data sources that provide data feeds exposed as APIs. Oracles are multi-agent systems and can even be used for conflict-resolution management [148]. On a technical side, oracles consist of smart contracts serving data requests coming from other smart contracts [227]. The issue of re-centralization and trust coming with the idea of using trusted external data sources, known as the oracle problem, is currently a subject of investigation [32].

Leveraging these building blocks, several approaches have been leveraging blockchain as a trustworthy tool for managing inter-organizational business process information systems. Blockchain offers a trustworthy decentralized tool to manage the deployment and execution of business processes. It enforces the traceability of the status and execution of the task (who executed the task, and when) while enforcing the control flow agreed upon at design time with other partners [135]. With blockchain, no central actor is required to manage the process [152, 156]. Additionally, data management is trustworthy as blockchain logs are immutable. Moreover, smart contracts can contribute to the trustworthy automation of redundant tasks. Indeed, smart contracts can monitor processes and automatize contractualization or billing tasks without needing a third party, and research efforts are being made regarding the development of legally binding smart contracts [60]. However, process flexibility and end-user accessibility remain challenges in the blockchain environment. In addition, blockchain environment specificities raise data privacy concerns: the history of transactions is accessible to the blockchain network participants. Hence, sensitive information can leak and be accessible to competitors. At the same time, a shared process should enforce a separation of concerns: internal data such as model and execution logs should not be visible to the other partners. Hence, a trade-off between ensuring the privacy of partners' private processes and exposing the public process thus arises [194].

Thus, using blockchain-enabled processes for trust triggers the following research challenges: process traceability, process flexibility, and data privacy. We detail each one of these research directions hereinafter.

1.2 Motivating example

Let's consider an inter-organizational process: a florist requests a delivery service to fulfill an order issued by a customer. Figure 1.1 depicts the

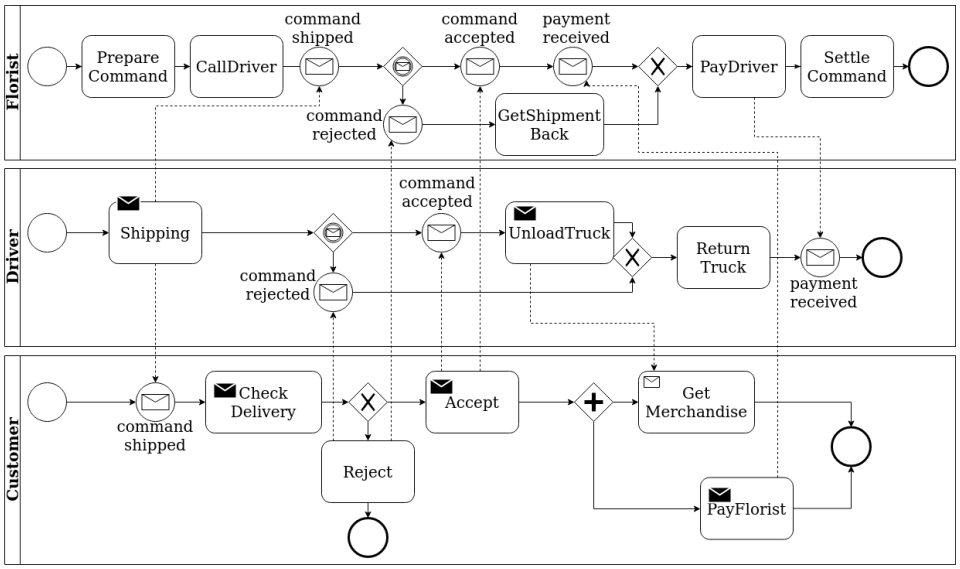


Figure 1.1: Motivating Example: BPMN Orchestration Diagram of Flower Delivery.

process using the BPMN (Business Process Model and Notation) standard and following an orchestration scheme. Several pools compose the workflow: the florist pool, the delivering carrier pool, and the customer pool. In this example, the inter-organizational process comprises public and private activities. *Shipping* or *Accept* are examples of public activities. The status of these activities is shared between partners for communication and coordination purposes. Upon executing the public task *Shipping*, Driver updates the activity status in the BPMS. The participants, Customer and Florist, will receive the status update. PrepareCommand is an example of a private activity. The activity is not shared with the other partners, and the execution status is managed in the private information system of the partner Florist. Finally, the execution of tasks follows an imperative control-flow: the execution of PrepareCommand will imply the execution of CallDriver, that will imply the execution of Shipping, etc.

The modeling choice of the cross-organizational process is also to be considered [164]. Figure 1.1 depicts an orchestration scheme: the global process is managed by a centralized entity. Choreography modeling can also be a fit in a cross-organizational setting as access to the global process is distributed across partners. Hence, a separation of concerns is provided by design.

In this cross-organizational setting, alongside modeling choices, trust can be challenged, e.g., if the customer declares that she did not receive one of the requested flower packages or if the florist suspects that delays occurred during the delivery. A trustworthy (or trustless) communication and coordination

tool such as blockchain could help ensure a trustworthy management of the process history data. The business process flow could be managed on-chain: hence, the history of services would be tamper-proof (knowing who unloaded the truck, who got the merchandise, what was checked before departure, etc.). Every participant could thus have access to the current state of the process by auditing the ledger. Governance of the business process, decentralized, would thus better fit the real-world state of the process. Moreover, such auditing could ease claim resolution as the ledger serves as a single source of truth. Smart contracts could address the need for task automation such as carrier selection, contracting based on preset criteria, or service settlement (pay carrier, pay florist). We build on this motivating example to illustrate our research problem and challenges hereinafter.

1.3 Research problem

Three important challenges arise regarding the development of blockchain-based cross-organizational processes.

The first challenge, illustrated in our motivating example, is the need for a trustless tool for coordination and decision-making in cross-organizational processes. Such a tool, embodied by smart contracts, could answer the need for a trustworthy deployment and execution of cross-organizational processes while enforcing the separation of concerns. The use of smart contracts to manage business process models with a decentralized focus has been investigated in [214, 112]. Each participant is identified as a blockchain participant, and a smart contract manages business process instances depicted as a BPMN diagram. For each BPMN activity execution, participants interact with the smart contract. Hence, the history of the transactions is stored in the blockchain ledger in a trustworthy and tamper-proof fashion. Nonetheless, in these approaches, the execution of private tasks off-chain is only mentioned, and the inner on/off-chain deployment and execution mechanism for process instances has not been detailed further. In our motivating example, the question about how to interconnect private tasks such as *PrepareCommand* in the florist view with the on-chain public view is not addressed.

Moreover, cross-organizational decisions such as resource allocation activities call for a trustworthy automation for adoption purposes. Indeed, the contractulization with one partner based on the outcome of a bidding auction should leverage tamper-proof data, and partners' bids should be processed without bias. Details regarding the decision process also require to be stored in a trustworthy and tamper proof fashion. In this context, smart contracts could be used to manage the runtime allocation of a given set of resources based on past services history stored on-chain. Hence, this design would ensure a certain quality of service while preventing siloed decision-making [159, 181, 162]. For example, the florist, who may not want to rely

on a single delivery company to ensure quality standards, could leverage a smart contract to find the carrier with the best quality of service (QoS) to ensure, e.g., a constant temperature during the delivery, or a just-in-time delivery. The smart contract resource binder could compute each available carrier's QoS to propose to the florist the best profile.

Hence, the first research question is thus *(RQ1) How to leverage smart contracts as a trustworthy distributed tool for coordination and decision making in cross-organizational processes?*

The second challenge relates to the dynamic nature of cross-organizational processes: such processes require (i) adaptation of the process at runtime and (ii) a dynamic assignment of actors.

Regarding process adaptation, participants in our motivating example may want to add a temperature sensor to deliver real-time data to monitor the temperature of the flowers though the smart contract managing the process is already deployed. Most related works consider changes in process orchestrations only [141, 49]. Additionally, approaches binding actors to roles in a process collaboration [125] currently push the burden of checking the transitive effect of new changes onto the new parties.

Regarding dynamic assignment, several rules, or constraints, have been implemented to retrieve resource-binding matches. Among such rules stand: (i) behavioral, structural, or execution constraints in [195], or (ii) roles, and statements in [121]. However, these notions are missing in the retrieved protocols. Moreover, to our knowledge, no QoS rules have been proposed in the literature to generate smart contract bindings (e.g., rules leveraging customer satisfaction or delivering carriers' punctuality). Additionally, the contracting stage is only addressed in [172] where parties negotiate binding terms.

Hence, the second research question is thus *(RQ2) How to deploy and execute in a flexible fashion cross-organizational processes managed on-chain?*

The third challenge relates to the privacy of sensitive data transiting in cross-organizational processes. Indeed, cross-organizational processes leverage sensitive information such as delivery price or truck size in our motivating example. Nonetheless, blockchain's open and transparent characteristic challenges this imperative for full or partial confidentiality in cross-organizational business processes.

Several publications propose to investigate blockchain-based sealed-bid auctioning [72, 103, 61, 71, 14, 192, 110, 51, 210, 22, 126]. Nonetheless, carrying on privacy-preserving auctions on blockchain with multi-party-computing reintroduces bidders interactivity, which is not desired to carry on auctions [192]. Additionally, using private blockchains lowers scalability and does not leverage smart contract automation benefits. Moreover, using trusted execution environments reintroduces a single point of failure as the content of offers is revealed to the enclave. Hence, there is a research gap in allocating services based on multi-objective sorting in a privacy-preserving

RQ1.1	How to carry out a separation of concerns that preserves the privacy of the private processes and trust of the public process for the deployment and execution of choreography processes in blockchain?
RQ1.2	How can smart contracts foster trustworthy and power-balanced decision-making for the runtime allocation of resources?

Table 1.1: RQ1 sub-challenges

fashion while ensuring auditability [132].

The third research question is thus *(RQ3) How to ensure the privacy of sensitive data while preserving the integrity and verifiability advantages of blockchain systems?*

These research questions can be further divided into several sub-questions. Some were previously discussed in related studies, and others were not previously handled but could be deduced from the flexible and sensitive nature of cross-organizational processes. We develop these research sub-questions according to RQ1 (c.f. Section 1.3.1), RQ2 (c.f. Section 1.3.2), and RQ3 (c.f. Section 1.3.3).

1.3.1 *(RQ1) How to leverage smart contracts as a trustworthy distributed tool for coordination and decision making in cross-organizational processes?*

Blockchain has been leveraged in the literature as a trustworthy coordination mechanism for collaborative business processes [214, 112, 115]. On-chain business process execution logs are not only tamper-proof but also made publicly available to participants, reducing power balance. Hence, the use of smart contracts as a trustworthy distributed tool for cross-organizational collaboration and decision-making should be investigated: table 1.1 presents our two related research sub-questions.

First, the potential for smart contracts to encapsulate business process models in a trustworthy fashion while preserving the separation of concerns shall be investigated. To better fit the collaboration aspect of cross-organizational processes and ensure a separation of concerns by design, we choose a choreography modeling approach. Business process choreographies comprise private processes carried by individual partners, where internal data such as model and execution logs should not be visible to the other partners. It also includes a public process where several partners collaborate in a coordinated way. In a choreography, all partners should trust the execution state of the public process. However, the trustworthy execution of the public process remains challenging as it is often managed centrally [112]. Additionally, a trade-off between ensuring the privacy of partners' private

processes and the exposure of the public process arises, e.g., the carrier should only have access to the set of tasks where she is involved and not the tasks of the customer or florist, and reciprocally. To our knowledge, research is scarce concerning the trustworthy deployment of choreographies leveraging blockchain. This deployment remains challenging as private information should not be shared between partners at design or runtime. Thus, the following question arises *RQ1.1 How to carry out a separation of concerns that preserves the privacy of the private processes and trust of the public process for the deployment and execution of choreography processes in blockchain?*

Additionally, a specific cross-organizational concern is allocating a participant to a given task in a dynamic fashion. Many cross-organizational processes call for more power balance and objectivity when assigning a task to one of several potential candidates. Public tasks such as allocation tasks should be objective and neutral regarding participants: power asymmetries, or collusion between participants, should be avoided. Hence the role of smart contracts as a reliable distributed tool for resource allocation shall be investigated more thoroughly while taking into account runtime customizable QoS metrics and leveraging past activity data shared between participants on-chain (e.g., the number of items delivered or the hour of delivery). Hence, we propose the following question: *RQ1.2 How can smart contracts foster trustworthy and power-balanced decision-making for the runtime allocation of resources?*

1.3.2 *(RQ2) How to deploy and execute in a flexible fashion cross-organizational processes managed on-chain?*

Due to the dynamic nature of cross-organizational processes, a blockchain-based business process management system should offer flexibility at runtime, e.g., adapt to a moving environment, support partnering, and operational business process agility [183]. Partnering agility consists of sharing competencies and upstream and downstream resources, and operational agility consists of working on speed and accuracy improvements, as well as cost economy, often by leveraging servitization. Hence our challenge is to improve model flexibility of cross-organizational processes in a trustworthy fashion by focusing on the partnering and operational aspects.

Table 1.2 presents the three research sub-questions.

Partnering agility should be taken into account when designing blockchain-based processes[183, 137]: models should integrate activities where the assignment to tasks can be realized at runtime (e.g., using push and pull strategies [70], and multi-criteria auctions leveraging, for example, the QoS of delivering carriers). For example, it should be possible to reallocate an activity to another actor if the actor initially assigned does not comply anymore, for instance, with the QoS requirements or if she is unavailable. We thus ask the

RQ2.1	How to foster actors' (re)allocation flexibility on running instances using a smart contract-based multi-criteria decision algorithm that leverages blockchain-based process history?
RQ2.2	How to model the flexibility imperatives of cross-organizational business processes?
RQ2.3	How to change the process model instances of blockchain-based cross-organizational processes?

Table 1.2: RQ2 sub-challenges

following sub-question: *RQ2.1 How to foster actors' (re)allocation flexibility on running instances using a smart contract-based multi-criteria decision algorithm that leverages blockchain-based process history?*

Operational agility should be examined in the blockchain context. The modeling language used to design business processes should offer flexibility due to the dynamic nature of cross-organizational processes. Hence, the modeling language should be flexible enough to adapt to process execution path variations at runtime. Therefore, we ask the following sub-question: *RQ2.2 How to model the flexibility imperatives of cross-organizational business processes?* Moreover, participants should be able to suggest modifications to the running process instance and negotiate change requests on-chain. We thus ask the following sub-questions: *RQ2.3 How to change the process model instances of blockchain-based cross-organizational processes?*

1.3.3 *(RQ3) How to ensure the privacy of sensitive data processed on-chain while preserving blockchain systems' integrity and verifiability properties?*

Blockchain offers a verifiable system; every action is recorded in a tamper-proof and decentralized fashion. In the blockchain network, all participants can access execution logs and retrieve the history of transactions, as well as the amount of gas spent for the execution payable smart contract functions. Additionally, the identity of participants is pseudonymous and not anonymous. Hence it is possible to re-identify users based on their transaction behavior (c.f., the Silk Road marketplace scandal where a drug dealing network was uncovered based on an analysis of the Bitcoin ledger [43]). Hence, if it is possible to keep sensitive information private using proxies in traditional cross-organizational business process management systems, the situation is more challenging in the blockchain context. Indeed, in the blockchain context, smart contract-mediated collaborations often occur in a competitive environment where competitors agree to use a blockchain to gain economic costs, e.g., they decide to bid for a service provisioning managed by a smart contract [214, 224]. Though multiple blockchain technologies can be used and interoperated [154], the privacy of sensitive data should nonetheless

RQ3.1	How to ensure a trustworthy access-control of business process activity data stored in smart contracts in blockchain-based cross-organizational processes?
RQ3.2	How to manage and compute numeric information in a privacy-preserving fashion using fully homomorphic encryption?
RQ3.3	How to ensure payment privacy in a cross-organizational process that preserves privacy while offering public auditability?

Table 1.3: RQ3 sub-challenges

be guaranteed while preserving the verifiability characteristic of blockchain systems. Hence, our third objective is to *maintain data and protocol integrity of cross-organizational processes while safeguarding data privacy.* Table 1.3 presents the privacy-related sub-questions linked to this challenge.

First, a reliable separation of concerns should be ensured between participants: activities and data should not be accessed and managed by all participants: they should only access the information they are concerned with. Hence the following sub-question raises: *RQ3.1 How to ensure a trustworthy access-control of business process activity data stored in smart contracts in blockchain-based cross-organizational processes?*

Additionally, the privacy of input and output data (e.g., information transiting from one task to another, such as the number of packages to deliver, a contract, or the hour of delivery) should be ensured in competitive contexts. In the context of privacy-preserving IoT data aggregation, [124] uses homomorphic encryption to preserve the inner value of IoT data: only ciphered data can be accessed by third parties while preserving aggregation operations. A blockchain, Dero.io, proposes to leverage homomorphic encryption for various applications such as voting or asset management [1]. Nonetheless, data processing such as auctioning is not addressed in this work. A privacy-preserving data processing of business data such as a price for a service or the volume offered in a truck is necessary to support flexible partnerships in a cross-organizational context. Hence, we propose the following refined sub-question: *RQ3.2 How to manage and compute numeric information in a privacy-preserving fashion using fully homomorphic encryption?.*

Finally, providing privacy and auditability of payment on blockchain systems has been a subject of concern in the literature since the beginning of bitcoin and cryptocurrencies [168]. Payment auditability is necessary to ensure trust in the system. It is facilitated with the blockchain ledger tamper-proof storing facility, which can be used as a trustworthy settlement log. However, a cryptocurrency payment reveals the content hidden in the

privacy-preserving auction though the winning service provider may not be willing to reveal the content of its bid to competitors [168]. Competitors can thus retrieve strategic positioning or trading secrets. Hence, auditability and privacy appear necessary at the binding and payment stages. We, therefore, state the following sub-question: *RQ3.3 How to ensure payment privacy in a cross-organizational process that preserves privacy while offering public auditability?*.

1.4 Thesis objectives, principles, and contributions

In this section, we present the main objectives of the thesis for answering the aforementioned research questions. We also propose a set of principles that will guide the fulfillment of the objectives. We finally present the main contributions of the thesis.

1.4.1 Thesis objectives and principles

In the light of the previously described research problems, the main objectives of this thesis are summarized as follows:

Objective 1 Identifying a business process modeling language that is flexible and a business process model that integrates a separation of concerns by design; implementing this modeling using a blockchain system (c.f., RQ1.1 and RQ2.2);

Objective 2 Integrating data to the blockchain-based business process management system, and leveraging dedicated off-chain storage (c.f., RQ3.1);

Objective 3 Making the public view upgradable and open to changes upon participants' requests at runtime (c.f., RQ2.2 and RQ2.3).

Objective 4 Carrying on runtime allocation by leveraging blockchain-stored tamper-proof data (c.f., RQ2.1 and RQ1.2);

Objective 5 Carrying on trustworthy privacy-preserving on-chain auctions while preserving the auditability of the smart-contract-based decision making (c.f., RQ3.2)

Objective 6 Providing a trustworthy, privacy-preserving, and scalable on-chain payment mechanism (c.f., RQ3.3);

To this end, we consider the following principles:

- **Automation:** the proposed systems should automatize the maximum tasks
- **Ease of use:** the proposed system should bridge the gap between (a priori) non-technical end-users, and smart contracts development
- **Privacy:** the proposed system should always take into consideration the privacy of users
- **Customer agility:** the proposed system should include business process participants as co-creators and testers

It is noteworthy that the proposed work in this thesis needs to be: (i) evaluated through experimental prototypes and (ii) collect end-user feed-backs on the proposed methods. Furthermore, the implementation, experiments, and results should be detailed.

1.4.2 Thesis contributions

To meet the above objectives while handling the described research issues, we propose the following contributions:

i **Blockchain as a trustworthy business process management tool: an on/off-chain declarative choreography deployment and execution system (c.f., Objective 1 and Objective 2)**

(a) We design and implement an on/off-chain deployment and execution strategy for on/off-chain choreographies, which enables a trustworthy business process management facility while enforcing separation of concern between participants at each step of the deployment and execution. More precisely, we propose an on/off-chain mechanism to generate public-to-private views of a declarative choreography and execute private views in a hybrid on/off-chain fashion. In doing so, no centralized party is necessary to control business processes while preserving the traceability of the process events. Meanwhile, we leverage a constraint-based business model language that enables the abstraction of the control flow under a set of constraints. By so doing, business modelers do not need to imagine all possible paths, a task that can be intensive and error-prone (ease of use principle). We moreover integrate a trustworthy smart contract-based access control to activity data (privacy principle).

ii **Control-flow and partnership flexibility:**

(a) *Trustworthy change mechanism for on-chain choreography instances (c.f., Objective 3)*: We propose a new approach

allowing for the change management of blockchain-enabled declarative business process choreographies modeled as DCR graphs (Declarative-Condition-Response graphs). Our system allows a partner in a running blockchain-based DCR choreography instance to change its private DCR process. A change impacting other partners is propagated to their affected processes using a smart contract. The change propagation mechanism ensures the compatibility checks between public DCR processes of the partners. We demonstrate the approach's feasibility through an implemented prototype and its effectiveness via experiments (customer agility principle).

(b) *Trustworthy resource allocation using blockchain (c.f., Objective 4):* We propose a system that leverages smart contracts for a dynamic selection of service providers. The system analyses service providers' performance stored as blockchain logs (automation and ease of use principles). The interest of this mechanism is to increase actor selection flexibility, as the selection may differ between process instances. Additionally, it enables an objective selection of service providers, as the same criteria are used to analyze each provider. Hence, information asymmetry and data tampering are prevented. We validate this approach's suitability in a real-world use case by deploying this mechanism in a procurement context for logistics services mapping using a design science research approach that integrates participants' feedback in the design process (customer agility principle).

iii **Data confidentiality in a blockchain environment**:

(a) *Trustworthy and privacy-preserving data computation (c.f., Objective 5)* To reconcile privacy imperatives with the benefits of blockchain, we propose to leverage fully homomorphic encryption (FHE) for blockchain-based calculations such as sealed-bid auctions. FHE is a cryptography protocol preserving operations over ciphered data. Smart contracts gather and orchestrate bid comparisons, while a computation oracle carries comparisons over ciphered data. Additionally, we propose to use the hybrid RSA/AES encryption protocol[1] to preserve bid confidentiality both on-chain and in the comparison oracle. Hence, our protocol compares competitive bids without any information leakage on the service providers' side or on-chain (privacy principle). In so doing, we contribute to the literature by designing a mechanism that addresses the privacy-preserving allocation problem using

[1]RSA stands for the name of its inventors, Rivest, Shamir, and Adleman, and AES stands for Advanced Encryption Standard

Research Question	Chapter	Publications	Contributions
RQ1.1	4	II	i(a)
RQ1.2	4	IV	ii(b)
RQ2.1	5	III	ii(b)
RQ2.2	5	I-II	i(a)
RQ2.3	5	V	ii(a)
RQ3.1	6	II-VI-VII	i(a)
RQ3.2	6	VII	iii(a)
RQ3.3	6	VI	iii(b)

Table 1.4: Mapping of research questions to corresponding thesis chapters, publications, and contributions.

blockchain and FHE while preserving the auditability and verifiability properties of the blockchain. We validate this approach through an implemented prototype.

(b) *Trustworthy and privacy-preserving payment (c.f., Objective 6).* Fully privacy-preserving schemes such as Monero, ARRR, or Epic Cash are not applicable in multi-party industrial processes, as partners already know each others. Hence, anonymity of the payment senders and recipients are not desirable. Meanwhile, we hypothesise that banks can be referred to as trusted entities for payment management. Hence, we propose a solution leveraging banks as trustworthy intermediaries while making the payment value secret (privacy principle). This solution uses an oracle bank and a per-collaboration payment token linked to a random value. Partners can use per-collaboration tokens to proceed to multiple payments while preserving values' privacy. Moreover, partners can program smart contracts to ensure escrows or carry conditional payment (automation and ease of use principles). Additionally, external peers can trust-worthily audit token transactions as they are stored on-chain. We demonstrate our approach's feasibility through an implemented prototype and its effectiveness via experiments.

1.5 Thesis outline

Section 2 introduces the main concepts related to blockchain and business process management used in the following manuscript. In Section 3, we present the main related work. Section 4, Section 6 and Section 5 present the three contributions related to trust, privacy, and flexibility respectively. Finally, we discuss and conclude this manuscript with Section 7.

Table 1.4 presents the mapping of the research questions corresponding to

the main thesis chapters (Section 4, Section 6 and Section 5), the publications, and the aforementioned contributions.

Chapter 2

Basic Concepts on Business Process Management and Blockchain

Contents

This chapter presents the basic concepts used in the remainder of this thesis, that will help address our research questions. In Section 2.1, we present basic concepts related to Business Process Management. In Section 2.2, we present basic concepts related to core concepts related to blockchain technology.

2.1 Business Process Management

Business Process Management refers to the discipline that focuses on the life-cycle of business processes [58] to analyze, design, implement, and continuously improve organizational processes [209]. In the following section, we present an overview of the business process life-cycle (Section 2.1.1) and then focus on two steps occuring in two stages of the life-cycle, namely

business process modeling (Section 2.1.2), and business process execution (Section 2.1.3).

2.1.1 Business Process Lifecycle

Figure 2.1 presents the the business process lifecycle. It comprises six steps [57]:

- business process identification consists in the identification of the process architecture relevant to a business problem (e.g., what are the main processes involved in a flower delivery;)
- business process discovery consists in the design of the as-is process model previously identified as relevant for the identified business problem (e.g., a BPMN diagram of the flower delivery process;)
- business process analysis consists in finding insights on the process weaknesses and their impact (e.g., for some delivery instances, the trucks leave with the wrong set of packages;)
- business process redesign consists in adapting the business process model to unify and correct the preidentified weaknesses (e.g., adding a package checking activity before shipping;)
- business process implementation consists in transforming the to-be process model in an executable process model: activities may be automated, and people are training for the organizational change;
- business process monitoring is the last step of the business process lifecycle: there, conformance checking is performed, and performance insights can be deduced from the event logs.

If bottlenecks, errors, or deviations are found during the monitoring stage, a new cycle starts, starting at the business process discovery stage (c.f., Figure 2.1.) Hence the business process life-cycle helps understand processes better and rationalize or automize them for speed, consistency, and quality improvements.

In the following, we focus on two stages that will be the focus of this manuscript, namely the business process discovery stage, that we refer to as the business process modelization stage, and the business process implementation stage, where we focus on the business process model execution managed by the business process information system.

2.1.2 Business Process Modelization

During the discovery stage, business process modelers produce a business process modelization that aims to formally represent a real-life business

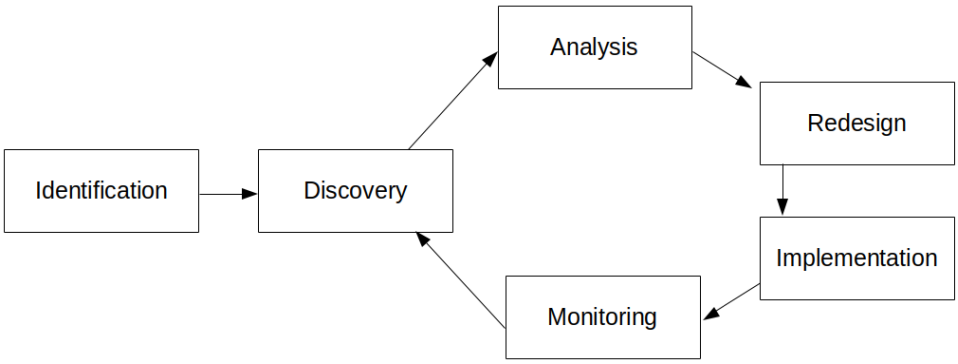


Figure 2.1: The BPM life-cycle

process. This representation comprises all the activities leading to the production of a business value. Activities comprise roles assigned to participants, services, or systems, and are linked to a set of physical and immaterial objects. Additionally, interactions between activities are also depicted. Such a model enables the communication between participants in an agreed-upon fashion. It also eases the autonomous execution of tasks (e.g., to call a service).

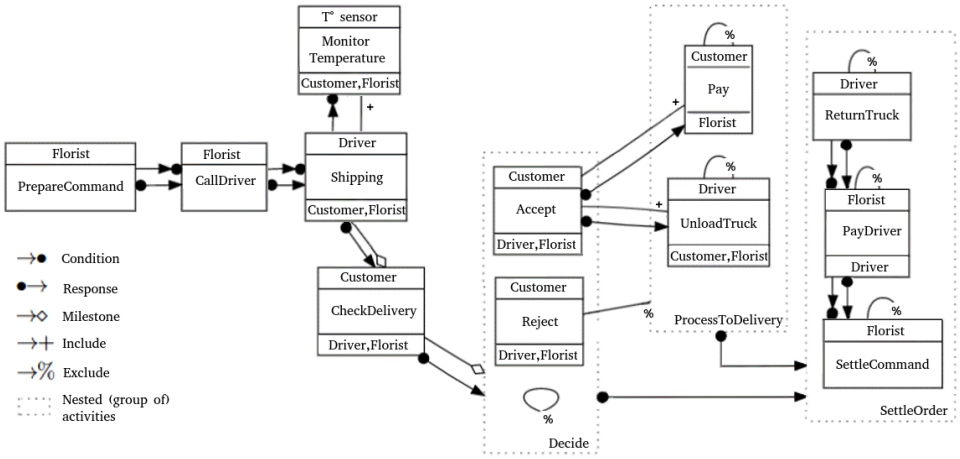
Several approaches coexist to model the interactions. The imperative approach consists into the explicit representiion of the paths along which a process instance runs. In this approach, activities can be linked together by sequential constraints. It has been standardized with a business process modeling language is referred to as BPMN An example of a BPMN model is depicted in Figure 2.2b. In reaction to the imperative approach, the declarative modeling approach aims to capture the constraints underlying the process, hence providing more flexibility to end users [63]. It has gained momentum in recent years, and is not yet standardized: several notations have emerged such as Declare, DCR Graphs, DMN, GSM, eCRG, or DPIL. For example, in DCR, the control-flow is abstracted into a set of pre and post-execution constraints. An example of a DCR business process model is depicted in Figure 2.2a.

2.1.3 Business Process Execution

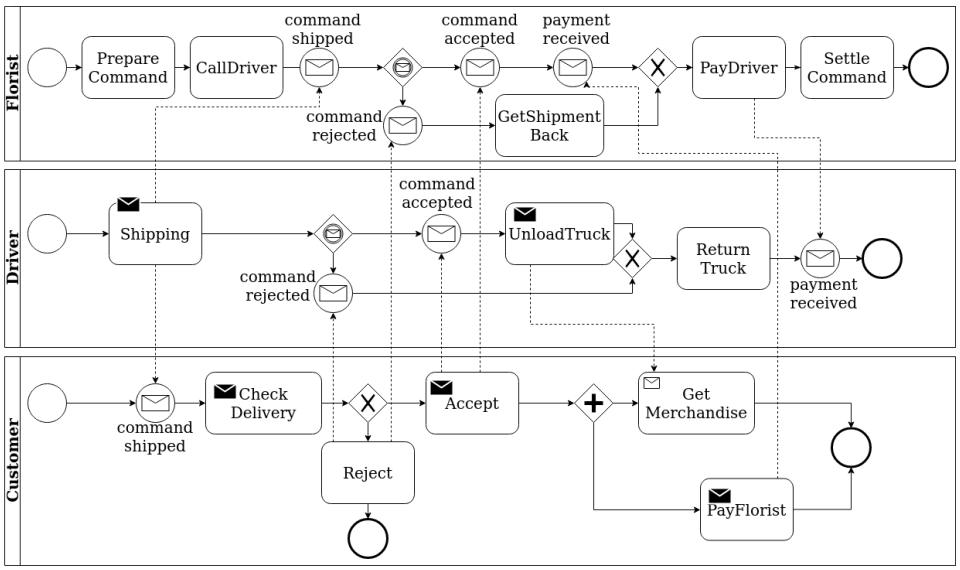
A business process management system can manage several instances of a process model. An instance comprises data specific to the execution, the state of execution, and the history of the execution.

If multiple role instances coexist at runtime, the assignment of one of the role instances to a task can be in a push fashion (participants request a task) or a pull fashion (the business process management system will assign the task to the participant).

Two execution strategies can be underlined, namely using orchestrations



(a) Declarative process of a flower delivery (DCR notation.)



(b) Imperative BPMN process of a flower delivery (BPMN notation.)

Figure 2.2: Examples of two modeling approaches of a flower delivery business process

or choreographies. With an orchestration, the management of business process instances is managed in a centralized fashion, by a trusted third party, or by one of the participants. In this setting, all activities, public and private, are managed by this central entity. Hence, the choice of a choreography is often motivated by the wish not to disclose internal and private activities to the other parties [58]. With a choreography, the management of business process instances is managed in a decentralized fashion by several partners. More precisely, the set of interactions between partners, the high-level *public view*, is shared between partners. It acts as contract for the coordination and communication between participants. Additionally, partners can refine their views by projecting the set of public interactions they are involved with over their roles and enriching such private views with internal activities. We talk about partners *private views*.

The execution of such decentralized processes is challenging, hence a decentralized information system, such as one leveraging blockchain smart contracts, is necessary. In the following, we present the main elements of the blockchain technology that will be leveraged in the rest of the manuscript.

2.2 Blockchain

Traditionally, identity management and services (e.g., transfer, redeem money) or accurate record management requires trust in a third party, whether through verified professionals or governments. Blockchain proposes an alternative to this scheme, where trusted third parties are not needed anymore. Blockchain can be defined as a protocol for storing and keeping track of the transfer of assets amongst multiple parties to ensure data integrity (i.e., storage should be immutable and transparent). It forms a distributed ledger where transaction records are batched into timestamped blocks. Each block is identified by a cryptography hash referenced by the previous block [42]. It is managed by a peer-to-peer network, i.e., the ledger is spread across multiple nodes, or servers [211], which prevents tampering attempts, as every participant holds a copy of the ledger. They lower the cost of trusted transactions by making databases tamper-proof by design [135]. The blockchain network corresponds to the set of nodes (clients) operating on the blockchain via the copy each one holds. It builds on (1) a tamper-proof data structure that captures the history of transactions, (2) algorithms that lead to consensus, and (3) market mechanisms that motivate the nodes to progress the network.

In the following, we present in more depth identity management in a blockchain network (Section 2.2.1), transactions management (Section 2.2.2), and smart contracts building blocks (Section 2.2.3).

2.2.1 Identity: reaching pseudo-anonymity with public and private keys

The notion of identity management onchain is key as each business process participant is assigned a set of public and private keys on chain and at the root of a trustworthy use of the blockchain technology. In the following, we present the key building blocks of identity management in blockchain networks.

Managing identity on-chain

Identity is the central concept used by users to receive, spend, or claim money. Traditionally, a username and password are issued by a central manager (e.g., the bank) to link the user to an account. User identity is confirmed through personal information such as a social security number or a name.

In the blockchain, identity is managed using public and private keys. Public keys are analog to a physical address, and private keys are analog to mailbox keys. The public key is the unique representation of each entity or user. The corresponding private key acts as a key to unlock the public key. In the blockchain, the public key is used for receiving money, while the private key is used to sign and send transactions. In practice, the private key is chosen randomly, for example, using a generator with a lot of entropy, and the public key is generated from the private key.

One of the main interests of using public and private key schemes is that no personal information is necessary to connect and prove one's identity. Additionally, the number of accounts is not limited, and there are no restrictions on keys taken.

Generating public and private keys

The digital signature scheme (often referred to as DSS) states that the pair (message, signature) can be considered secured if the recipients can verify [104]:

1. the message origin, i.e., whether the original sender has authorized the transaction;

2. the non-repudiation of the message, i.e., the original sender cannot backtrack;

3. the message integrity, i.e., the message cannot have been modified since sending).

Examples of key generation schemes are RSA (Rivest, Shamir, Adleman), ECDSA (Elliptic Curve Digital Signature Algorithm), ECDH (Elliptic Curve Diffie-Hellman), or DSA (Digital Signature Algorithm).

Finally, more complex schemes for multi-factor challenge-set identity authentication (referred to as MFSSIA) have been proposed in the literature to support a trustworthy identity authentication of process participants (humans, machines, software systems etc) [154].

2.2.2 Transactions and record-keeping :

The use of the blockchain for managing business processes is motivated by the fact that each action mediated by the blockchain will be recorded in a tamper-proof fashion. Hence, understanding the building blocks for a trustworthy record-keeping is necessary to understand how blockchain technology can be relevant to record business process executions in a competitive trustless context.

Account and UTXO models

A valid transaction is a transaction that (1) has been signed (there is proof of ownership), (2) has been signed by an account with available funds, and (3) there is no other pending transaction using the same funds.

With blockchain, two main models have been defined to validate transactions in a trustless distributed fashion. The blockchain keeps track of unspent money with the UTXO model (standing for *Unspent Transactions Output*, specific to Bitcoin [146]). Each account holds a set of unspent transaction outputs, which consists of money that has been sent to the account and not yet spent. Account models are similar to traditional bank accounts where a central manager keeps track of account balances: each user is assigned to an account with a balance, and money can be spent if the balance remains positive. Then, money is subtracted from the total balance. Ethereum's design illustrates this type of account.

Cryptography hash functions

The bedrock of the blockchain system is the **cryptography hash function**. Its goal is to enforce **information integrity**.

The cryptography hash function extends the hash function, which outputs, with some degree of deterministic randomness, 256 bits. It verifies three properties: pre-image resistance, second-image resistance, and collision resistance. These properties enforce trust and discourage any tampering attempts regarding data or identities. Indeed, the avalanche effect occurs at any change attempt: a small change in the input drastically changes the output. For example, Bitcoin's cryptography hash function is $SHA256^2$, Litecoin uses scrypt/RFC 7914, and Ethereum uses KECCAK-256, a hash function that won the NIST SHA3 competition in 2012 [26].

The two blocks of blockchain using cryptography hash functions are (1) the tamper-evident database and (2) the secured transactions mechanism. Hence,

cryptography hashes and asymmetric encryption such as ECDSA provide a pseudo-anonymity of the participants and enforce the non-tampering of the transactions. Pseudo comes from the fact that the ledger links user addresses and transactions. Anonymity is hence, by design, not fully guaranteed. Blockchain networks can be permissionless (any node can join or leave the network) or permissionned (nodes must be approved to join the network; leaving the network may be prone to more difficulties).

Hash function inputs are referred to as pre-images. Hash function outputs are referred to as images.

Cryptography functions are a type of hash function, written Ψ. With a cryptography hash function, we can take any pre-image of any size and obtain an output of 256 bits.

Cryptography hash functions are hash functions with three special properties:

1. The first propriety is *pre-image resistance*: given $\Psi(\theta_1)$, it is computationally difficult to determine θ_1. As a metaphor, intuitively, it is as difficult as tracing one identity from a given fingerprint.

2. The second propriety is *second-image resistance*: Given θ_1, it is computationally difficult to find some value θ_2 such that $\Psi(\theta_1)==\Psi(\theta_2)$. As a metaphor, intuitively, it is as difficult as finding someone with the same fingerprint as you.

3. The third propriety is *collision resistance*: It is computationally difficult to find θ_1 and θ_3 such that $\Psi(\theta_1) == \Psi(\theta_3)$. As a metaphor, intuitively, it is similar to trying to find two random people with the same fingerprint.

With a cryptography hash function, taking an infinite amount of inputs, a finite amount of outputs is possible, which does not prevent collisions. Nonetheless, cryptography hash functions are collision-resistant: it is hard for someone to find the two inputs that will produce the same output. Additionally, a slight change in the input produces a pseudo-random change in the output (it is referred to as the avalanche effect). Hence, it prevents "hot or cold" games with inputs to produce or predict outputs.

Different cryptography hash function algorithms exist. For example, Bitcoin uses $\Psi(\theta_1) = SHA256(SHA256(\theta_1))$ also referred to as $SHA256^2$, which was designed by the NSA.

Record-keeping (the blockchain).

Blockchain systems store data transactions hierarchically, using linked blocks of aggregated transactions [218]. This record-keeping storage facility is referred to as the blockchain ledger. Figure 2.3 illustrates such ledger. Each block contains (1) the reference to the previous block, (2) a tamper-evident

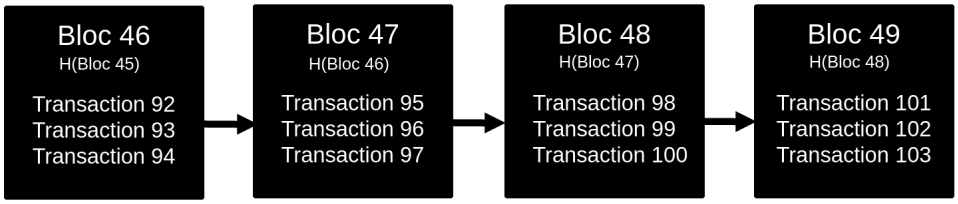


Figure 2.3: Illustration of a segment of the blockchain ledger (blocks 46-49)

digest of the transaction history to attest the integrity and block order, and (3) the list of the transactions to commit [205] stored as a Merkle tree. A Merkle tree (or hash tree) consists of a tree comprising two kinds of nodes, leaf, and non-leaf nodes, with two labeling strategies. Leaf nodes are labeled with the hash value of a data block. Non-leaf nodes are labeled with the hash value of the child nodes labels.

Block headers comprise three main elements ensuring the integrity of data.

First, the block header comprises the base of the Merkle tree of the transactions, also known as the Merkle root [136]. It corresponds to $\Psi(\Psi(\rho_l) + \Psi(\rho_r))$ where ρ_l and ρ_r are the respective left and right branches of the Merkle tree. The interest in the Merkle root is twofold. First, a slight change in one transaction can be traced back down easily due to the avalanche effect of the cryptography hash function. Hence, it enables checking easily if data is corrupted. Second, it provides direct proof of the inclusion of transactions to blocks: instead of proving all the transactions are correct, one can give the roots of each branch. If they are correct, the rest is correct. Additionally, it is hardly tamperable due to collision resistance.

Second, for proof-of-work-based blockchains, the block header comprises the block nonce (an acronym for number only used once), which corresponds to the number found through brute force computation by miners, i.e., the mining solution of the puzzle. Indeed, the proof of work consensus requires miners to solve a computationally difficult puzzle that is (1) adjustable (if more resources or more people enter the game, then difficulty should increase) and (2) easily verifiable. The problem of finding a 32-bit number, where the value to be found is a random integer between 0 and 2^32, answers this requirement. The puzzle difficulty corresponds to a representation of the expected number of computations required to find a block. It is implemented as a requirement of the leading number of 0s, which adjusts every 2015 block based on the average time spent to solve a block in Bitcoin.

Finally, the block header comprises a reference to the hash of the previous block, which is computed as $PrevblockHash = \Psi(blockHeader) = \Psi(prevBlockHash \parallel merkleRoot \parallel nonce)$. This hash links blocks together (with the block id).

Trust without trust: distributed systems and consensus mechanisms

Distributed systems A distributed system can be defined as a network of independent nodes, each representing a process, talking to each other via messages, and often accomplishing a common goal (e.g., keeping track of money transactions).

A distributed system verifies the following properties:

- *Concurrent components*:
- *Message sharing*:
- *No global clock*:
- *Potential failure of individual components*:

Consensus helps render the majority opinion and frame how nodes reach a general agreement on the ledger's state. The study of distributed systems and consensus helps manage fault tolerance by trying to answer the challenge of creating an overall system that is reliable though some components might be unreliable.

A distributed system should *offer correctness*, i.e., ensure safety (no undesired state can occur, i.e., a participant introducing a false transaction in the context of blockchain, as long as the protocol does not reach the specified threshold of misbehavior) and liveness (as long as the protocol verifies the right proportion of honest nodes, the system will behave as intended, i.e., in the context of blockchain, a correct transaction will eventually be added to the chain) [6].

To ensure correctness, one uses a consensus algorithm to enable the distributed system *reaching a consensus*, i.e., returning the majority value. The consensus shall offer the following three properties:

- *Validity*: any value decided upon must be proposed by one of the processes
- *Agreement*: all non-faulty processes must agree on the same value and will never decide on trivial, random, or different values
- *Termination*: all non faulty nodes eventually decide

The CAP (or Brewer's) theorem is a fundamental theorem for distributed systems. This theorem states that a distributed system can only verify two of the three following properties:

- Consistency (every node of the distributed system provides the most recent state)
- Availability (the system offers a consistent read and write access)

- Partition tolerance (the system works despite the disconnection of certain nodes)

With blockchain, partition tolerance is verified by design (nodes are independent and can fail). A trade-off between consistency and availability is thus at stake and varies depending on the choice of the consensus algorithm.

We present the two main consensus algorithms hereinafter and refer the reader to [140] for a broader vision (PBFT[1], voting-based, or federated consensus).

Consensus: ensuring all users stay on the same page. The trusted behavior of blockchain systems builds upon the consensus protocol used to update the chain of blocks [12]. A set of nodes (or users) hold a copy of the ledger and update it following a set of rules, a consensus protocol. The consensus protocol defines the protocol followed by the nodes to verify and append new transactions to the chain. This protocol ensures the tamper-proof growth of the database [218]. Proof-of-work and proof-of-stake are among the best-known consensus protocols. With proof-of-work, a puzzle needs to be solved by miners to append a new block and get a financial fee. With proof of stake, the node responsible for adding a new block of validated transactions has the most assets at stake. These rules help avoid malicious nodes or invalid transactions. Merkle trees ensure the ledger's integrity by linking transactions using cryptography rules.

In the traditional model, each user submits transactions to a central authority that decides on the validity of transactions. With blockchain, siloed decisions are not possible as it would be prone to double-spending attacks. Each user has to be concerted. The majority rule alternative, with proposers and voters, is also not applicable as Sybil attacks are possible: a malicious user could create enough nodes to obtain the majority.

To prevent double spending and Sybil attacks, the strategy, referred to as proof of work, is to apply a variation of the majority rule to make it prohibitive to cheat via Sybil attacks. To do so, users must pay to become transaction validators. They gather new transactions to form the next block. Forming the next block implies running a cryptography hash function several times. Indeed, validators must provide a solution to a hash puzzle that can only be solved through brute force computation. It takes around 10 minutes to solve the hash puzzle. By running this cryptography hash function, validators consume computation power, proving that they have worked and defusing cheating attempts. An economic incentive is added to avoid having one miner who would take control only: the first miner who solves the puzzle receives the authorization to form the next block and receives the block and fees reward (i.e., all transaction fees): he is compensated for its power. Hence, mining defuses the Sybil attack as there is insufficient computing

[1]Practical Byzantine Fault Tolerance

power to vote multiple times. It is to note that computing costs depend on mining hardware (e.g., CPU, GPU; Ethereum is ASIC-resistant[2]) and operations (e.g., energy consumption, network connectivity).

Alternatives to proof of work focus on spending something else than computing power. For proof of stake, the resource consumed is the native currency. Validators replace miners, they put their stakes into escrow. In case of misbehavior, their stake is destroyed. Another alternative to proof of work is proof of burn: the resource, e.g., one bitcoin, is burned, i.e., fully consumed, in exchange for a coin in another cryptocurrency. With proof of space, storage space in decentralized clouds is consumed. With proof of elapsed time, the resource consumed is time. They can be implemented using trusted execution environments (TEEs) and verifiable delay functions (VDFs). Such proof depends on assumptions of randomness and trust in the manufacturer (e.g., Intel for SGXs).

Choosing the right consensus algorithm Network consensus is necessary for blockchain to agree on the validity and order of the transactions in the ledger. The risk of nodes disagreeing on the state or order of transactions would otherwise create forks and have running variations of the ledger.

The choice of the consensus algorithm depends on (1) the blockchain network (public or private) as well as (2) the attack vector.

For public networks, an incentive mechanism is proposed for solving hashing cryptography puzzles to prevent Sybil attacks. The proof of work is the main consensus proposed to defuse Sybil attacks in public networks. Proof of work consensus is the consensus chosen for Bitcoin and the initial version of Ethereum. Its goal is to make it too difficult for malicious users to introduce fake branching. Mining is a time-based competition, as difficulty comes from the computation time necessary during mining. Indeed, the concept of mining is to make valid the longest chain. To append a fake branching, one must build the longest chain and possess more than 50 percent of the overall computing power. Several hashing algorithms can be used for proof of work, such as SHA-256, Blake-256, scrypt, or myriad. Nonetheless, the hash puzzle requires heavy computation, which causes energy concerns. To avoid energy waste, some propose to change the hash puzzle to solve current research problems, which are computing intensive such as protein folding. Several other consensus protocols have been proposed to circumvent this computing issue. The most famous one is the proof of stake consensus, which requires fewer CPU computations for mining than proof of work: node validators are the ones with the most coins at stake, not the ones that are the fastest to solve the hash puzzle. Proof of stake consensus consists of delegating the validation power (ordering and creating new blocks for the

[2]Application Specific Integrated Circuit

nodes to reach a consensus). A small group of elected people can validate the transactions. Elected people put to stake their assets, which prevents malicious behaviors, as validators are less prone to risk their assets. Among the advantages of proof of stake are energy efficiency and lower entry barriers, as computation power during mining is no longer necessary.

For private blockchain networks, incentive mechanisms are unnecessary as Sybil attacks are not prevalent. The Practical Byzantine Fault Tolerance (PBFT) consensus mechanism proposes a one-third resilience to attack vectors. Tangaroa, a Byzantine Fault Tolerant (BFT) variant of the Raft algorithm (Tendermint), offers one-third resilience. Ripple proposes trusted/whitelisted subnetworks that offer one-fifth resilience to attack vectors. Another example is SIEVE, one of the two consensus protocols used in Hyperledger Fabric (alongside PBFT). SIEVE adds speculative execution and verification phases to the PBFT algorithm to detect and filter possible non-deterministic requests.

2.2.3 Onchain execution logic with smart contracts

Finally, we present onchain executing logics building blocks, as these building blocks (smart contracts, oracles, IPFS[3], or tokens) will be leveraged to support business process management strategies onchain in the rest of the manuscript.

Smart contracts

Nick Szabo first theorized them in 1994, where the author defines smart contracts as "a computerized transaction protocol that executes the terms of a contract" [198]. Smart contracts can be approached as deterministic scripts, executing any on-chain logic expressed as a function of on-chain data inputs [42]. Smart contracts have a deterministic behavior because of a closed-world assumption: only the information stored on the blockchain is available at runtime. Smart contracts can run the business logic, self-enforce contractual clauses translated into code, and manage business processes in an autonomous fashion [135]. Hence smart contracts are relevant for managing data-driven interactions [42]. Smart contracts have their state and can take custody over assets on the blockchain, which is helpful for data-driven processes (e.g., for trading x assets for y others) and executing services on demand [31].

Contracts and smart contracts should be approached as two different concepts. Contracts can be defined as written or spoken agreements intended to be enforceable by law (cannot be broken, it is going to happen). A smart contract, on the contrary, is code. It facilitates, verifies, and enforces a digital contract's negotiation or execution.

[3]Inter Planetary File System

In practice, every node runs a blockchain virtual machine, and the blockchain network acts as a distributed virtual machine. Smart contacts are triggered by messages or transactions sent to their address and executed into this distributed virtual machine.

The main advantages of smart contracts are (1) the possibility to execute multi-step processes and (2) the possibility to inspect the code before launching a transaction. Hence, smart contracts offer informed decisions, certainty of execution, and verifiability over the process.

A deeper focus on Ethereum Ethereum is a decentralized platform building on blockchain and designed to run smart contracts. It acts as a distributed computer to execute code and as a distributed state machine as each new transaction changes the global state.

Ethereum smart contracts can be approached as autonomous agents that live inside the Ethereum network, triggered by transactions. They have direct control over the internal ether balance and internal contract state. Smart contracts have four main uses:

1. *store and maintain data* (e.g., a token currency, a list of memberships, etc.)
2. *manage contract* or relationship between untrusting users (e.g., financial contracts or escrow)
3. *provide functions* to other contracts and serving as a software library (e.g., for secured mathematical operations)
4. *manage complex authentication* (M of N multi-signature access)

The execution and verification of Ethereum smart contract transactions are realized using the Ethereum distributed computer. Every node of the blockchain network executes the smart contract. Afterward, they all reach a consensus (using proof-of-work) regarding the new network state. Miners competitively create blocks of transactions by running code and searching for a solution to the mining puzzle. Hence, there is no need for trusted third parties as a violation of a smart contract would imply subverting the entire network. Smart contracts offer secure peer-to-peer agreements that live on the blockchain in a tamper-proof fashion.

Every Ethereum node runs Ethereum virtual machines (also referred to as EVMs) to avoid OS incompatibilities between different machines. Ethereum virtual machines can be viewed as mini-computers running smart contract code. Every blockchain node compiles the Solidity code into EVM bytecode (a low-level, stack-based bytecode language, the instruction set to be executed by a processor). The compilation chain is the following: the solidity smart contract is compiled into the compiler language solc, then into bytecode (the opcode view), then the stack (hexadecimal), and finally the memory (into the

blockchain). It is to note that the opcode can be decompiled using reverse engineering techniques and that the pseudo-code can be traced back.

The halting problem states that it is impossible to determine ahead of time whether the contract will ever terminate (if, for example, a denial of service attack occurs). Hence, to prevent infinite loops that could be damaging when executed in a distributed fashion, Ethereum introduces the notion of a gas fee: every contract requires gas which fuels contract execution. Every transaction specifies the maximum quantity of gas to be consumed and the Ether gas price corresponding to the fee the transaction issuer is willing to pay per unit of gas. Hence, when purchasing gas, transaction issuers purchase distributed, trustless computational power.

In practice, the transaction fee is subtracted from the sender's account at the start of the transaction. If the transaction is successful, the remaining gas is refunded. Else (an infinite loop occurs), the execution is reversed, but the amount is not refunded, and an attacker looking to launch a DoS (Denial of Service) attack will need to supply enough ether to fund the attack. Additionally, the transaction issuer supplies gas for one validator to run the code only, which acts as an incentive mechanism. Indeed, only one block is added to the blockchain, and only one user is compensated for the computing power.

Smart contract executions are redundantly parallel to reach consensus, which is expensive and memory-intensive. Such behavior encourages developers to prefer off-chain computation for use cases that do not necessitate trust.

Blockchain oracles and IPFS

Some smart contract services depend on external data. Oracles bridge the closed blockchain network with web APIs to forward external data to smart contracts. Oracle services repeatedly trigger the same API and triangulate the results to prevent malicious behavior on the API side. An alternative use for oracles is to ask APIs to carry heavy computations, which are otherwise too expensive to process directly on smart contracts. Among commercial oracles stand Chainlink and Provable.

IPFS is an open peer-to-peer network for file sharing providing high throughput and low latency [19]. Two main building components are distributed hash tables and a Merkle directed acyclic graph (DAG). First, distributed hash tables provide high throughput and low latency for data distribution: nodes of the IPFS network can store and share data in a decentralized fashion. Second, Merkle DAG provides content addressing and tamper-resistance: data are identified uniquely, and permanently stored with integrity. In the blockchain context, IPFS provides a reliable and low transaction cost storage capacity [95]. This is especially of use for blockchains with a proof-of-work consensus, such as Ethereum, where storing data on

smart contracts is costly.

Payment tokens

Smart contracts can implement and manage blockchain tokens. These tokens can have multiple uses [158]: payment tokens, equity tokens, or cryptocurrency tokens, to name a few. With tokenization [186], tokens encapsulate sensitive data and lower the risks of exposure and sensitive information leakage. The Ethereum blockchain introduces the token smart contract standard [207]. This standard refers to the list of functionalities implemented by a smart contract for token management. Smart contracts create a new type of token by setting the total number of token supplies, the number of decimals of the token, its name, and its symbol. It also manages the tokens' first allocation from a token generator to participants while verifying the total number of token supplies limit. It keeps track of the token balance of participants and manages the transfers among participants.

Chapter 3

Related Work

Contents

3.1 Introduction

In this chapter, we review the existing works relevant to the topic of decentralized BPMS using blockchain. We classify these works according to the three following categories, each one referring to one of the research questions presented in the introduction: blockchain-based business process

management system (c.f., RQ1), business process management flexibility (c.f., RQ2), and business process privacy (c.f., RQ3). We detail in more depth in Sections 3.2-3.3 each one of the proposed categories. Finally, we compare related approaches in Section 3.5. We classify each work using a concept matrix presented by Webster and Watson [215] as a way of systematically collecting and analyzing the different blockchain-based business process management systems. By so doing, we identify research gaps that motivate the objectives previously identified in the introduction. We then conclude in Section 3.6.

The work presented in this chapter (mainly in Section 3.2, Section 3.4, and Section 3.3) was published in international conferences HICSS [91, 90] and BPM [27] and in the journal preprint [89].

3.2 Blockchain-based BPMS

In this section, we investigate related work on blockchain-based business process management systems. We do so by presenting the evolution of related work from empirical approaches leveraging blockchain to model-based approaches (c.f., 3.2.1). To investigate related work focusing on the challenges identified in RQ1, we propose a classification of such works according to (1) the choice of business process modelization which impacts the system flexibility and scalability (c.f., 3.2.2), (2) the public/private views separation which impacts confidentiality (c.f., 3.2.3), and (3) the deployment which impacts participants' trust (c.f., 3.2.4).

3.2.1 From empirical to model-based management of processes on-chain

Several approaches empirically show the usefulness of blockchains for asset management. In these approaches, smart contracts are developed from scratch and designed according to the business need. For instance, a luxury supply chain [229] mimics an asset monitoring process using a Hyperledger Fabric chaincode (a smart contract variant) derived from a BPMN collaboration diagram. There, an EPC-based IoT network composed of RFID chips is used to track assets. Similarly, a food delivery process is successfully implemented using a Quorum-based private blockchain [223]. Moreover, an industrial prototype of trusted energy performance contracts using Ethereum smart contracts is proposed in [80]. To improve scalability, block-free directed acyclic graphs, such as IOTA have also been proposed as a distributed ledger alternative to blockchains [55]. By removing blocks (each new transaction verifies former transactions), miners are removed. By removing miners, the threat of centralization implied by mining pools vanishes. This distributed ledger technology has been empirically used to trade energy in a peer-to-peer fashion though this architecture increases transaction time [165].

The empirical development of process smart contracts requires strong development skills, the design stage is therefore costly and time-consuming. This issue can be circumvented by abstracting smart contracts into sublayer stacks [54, 111, 13]. Figure 3.1 depicts the semi-automatic deployment approach used to deploy model diagrams to the blockchain. First, the diagram is fed to the system. It is ingested, interpreted, and mapped to a smart contract template with a translator module. Second, the generated smart contracts can be reviewed by a human operator. Lastly, the smart contract is deployed in the blockchain. By abstracting the underlying smart contract code to the eyes of the business modelers, the design process is faster and more reliable [122].

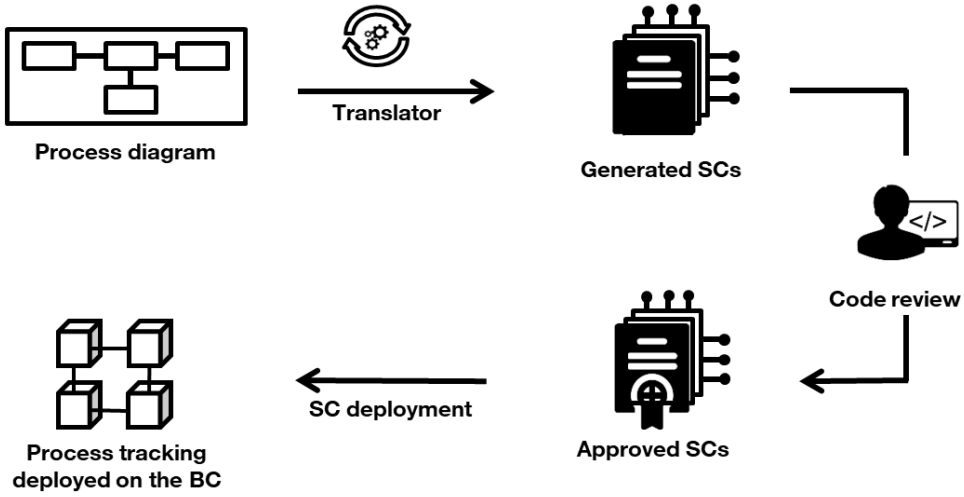


Figure 3.1: Model-engineering pipeline for blockchain-based BPMS.

In the following, we classify related works managing collaborative processes on-chain according to (1) business process modeling choices, (2) view-based approaches, and (3) deployment strategies.

3.2.2 Modeling stakes: focus on the imperative and declarative approaches

The **modeling paradigm** criterion refers to the process modeling choice used to represent collaborative processes on the blockchain. The choice of the modeling language impacts the way the control flow is described. It consequently influences the smart contract translation step and the flexibility of the process.

In preliminary work, the eSourcing conceptual framework that uses a smart contract application layer is proposed [153, 155]. Such a layer is used for transacting decentralized autonomous organizations (DAOs).

This framework helps build the life-cycles of business collaborations by stipulating the setup stage, the enactment, rollback, and termination of collaborations. Nonetheless, the eSourcing framework is process-tree-based (in earlier papers Petri-net based) and does not address declarative business process choreographies. We detail hereinafter the imperative and declarative modeling approaches proposed in the literature.

The *imperative modeling* approach consists of approaching business processes as an ordered sequence of enforceable tasks. BPMN is the standard notation used to depict processes in an imperative fashion. The literature reports two BPMN-based blockchain monitoring systems. Caterpillar [214] executes business processes fully on-chain. Its focus is on control flows: a translator component maps BPMN diagrams into a simplified Petri net translated into Ethereum's Solidity smart contract. On the execution side, partner instances are generated corresponding to the affectation of a partner to a role. A process instance is also generated. To ensure trust, each involved partner computes its own version of the contract, to be compared later. At runtime, a local trigger links API calls to blockchain transactions, and process history is stored using IPFS, a decentralized network protocol providing storage facilities [109]. Moreover, data structure optimizations have been implemented to cut execution costs [73, 122]. Similarly, Lorikeet [201] focuses on the mapping of BPMN choreography processes into smart contracts. Only the message flows between partners are stored on-chain. In both Caterpillar and Lorikeet approaches, though at different degrees, security and privacy issues are taken into account: for example through participant binding and asymmetric data sharing [54].

Other works such as [28, 96, 127, 193, 138, 60] use the *declarative modelling* approach where only execution constraints are specified. This modeling approach answers the need for more execution flexibility, as only the specification of a set of rules to be followed by the process is needed. Hence the sequencing of the tasks is indirectly enforced, and business process modelers do not need to specify all the execution paths at design time. One protocol, ADICO, focuses on institutional grammar. On the semantic side, institutions embody behavioral patterns among people. Strategies, rules, and norms frame these patterns. On the translation side, the semi-automated translation of textual inputs generates smart contracts [69]. The execution has two facets. First the translation is semi-automatic: the developer can customize the generated smart contract to prevent deviating cases. Then, smart contracts are instantiated: they are compiled into an EVM bytecode. The complexity of the contract and the predicted gas consumption are provided to the user before being committed to the blockchain network. Another protocol, BCRL (Business Collaboration Rule Language), focuses on controlled English sequences of the form when-if-then [11, 96]. The user declares the set of business rules to be ingested in a rule parser, which will, in turn, instantiate a RETE algorithm (a pattern-matching algorithm

adapted to rule-based systems). A smart contract hosted on Hyperledger Fabric embeds the rule engine. A dedicated API triggers the engine when needed. Finally, Dynamic Condition Response (DCR) graphs build on declarative event process flows [92, 189, 127]. On the modelization side, each node represents an event. The ordering of the events is made through role assignment (person or machine) and causal or conflictual relationships. The strength of this approach is the ease of modelization and the flexibility of process execution paths. A research work proposes LTL for smart contract parametrized pre and post-execution conditions, however without including implementations [96]. Authors in [138, 193] use the artifact-centric language. Pre and post-state conditions of the artifacts indicate the completion of an activity. The artifact-centric process modeling relies on the guard-stage-milestone principles. The guards are the set of conditions to be met to trigger a stage activation. The milestones are the set of conditions to be met to settle a stage. The stages are the set of tasks to be executed. Two blockchain-based BPMS implementations of this approach prove the validity of the model [138, 193]. In [28], authors propose a set of transformation rules to convert touristic itineraries presented in XML to BP choreographies which are then implemented as smart contracts and executed on the Ethereum platform [28]. A research work builds on this approach to auto-generate Solidity smart contracts using a legal contract markup language based on XML [60]. However, the authors do not consider the separation between global and local views in the choreography.

3.2.3 View-based approaches

The **view-based** criterion refers to the separate display of the global process: in a view-based setting, participants only have access to their tasks. This criterion is important as the separation of concern is necessary for the adoption of a mechanism in a coopetition context: competing partners do not wish to share parts of their internal processes with their peers.

Regarding traditional view-based approaches, authors in [37, 107] use process views to build an abstracted version of each partner's private processes in order to hide its internal structure. In [107], authors define a Symbolic Observation Graph (SOG) for each choreography participant. A SOG is an abstraction of the reachability state graph of a formally modeled process (e.g., an LTS). The nodes in the SOG are meta-states, i.e., a set of states connected by unobserved (internal) activities, and the edges are labeled with observed (interaction) activities. The SOG of the choreography process is the product of the SOGs of the participants. In [37], roles inter-connect via a set of virtual activities. These virtual activities abstract choreography interactions, and are enacted by a trusted third party. In these works, partners' privacy is reached by separating public and private views. However, trust issues remain as shared execution logic and data are managed in a centralized fashion,

often by a third-party [112].

In the literature, several research works leverage blockchain for business process model management but do not consider the public/private view separation [96, 193, 138, 28, 176, 64, 161, 127, 214]. For example, in [127, 214, 120], authors handle orchestration schemes only. A choreography is considered in [112] but the authors do not expand on the participants' private workflows execution and deployment. Though the generation of the public and private views in [112] is suggested, projections are not enforced in a trustworthy fashion in this work. The Dibichain protocol proposes to minimize information stored on-chain to preserve privacy during the exchange of supply chain information by storing only links to in-house data storage locations in [194]. Nonetheless, smart contract-based automation of processes such as resource allocation or payment escrow is not considered in this approach.

3.2.4 Business process instance deployment strategies

The **deployment** criterion refers to the mechanism chosen for deploying an instance of a business process model on the blockchain. As cross-organizational processes hold both internal and public activities, a strategy is needed to connect the onchain process management system to the offchain ones stored that are stored in a decentralized fashion in each partner's private information system.

Regarding fully on-chain schemes, a translator maps directly BPMN [214, 112, 120], DCR [127], or XML [28] models into Solidity. Additionally, a custom interface binds local execution engines with blockchain in [112]. In [176], authors run choreographies with Bitcoin instead of smart contracts. Two research works advise the direct end-to-end deployment of public processes [96, 193]. Finally, a smart contract stores the hash of an artifact-based multi-party process but no details are given on off-chain tasks [138].

Regarding hybrid on/off-chain schemes, a hybrid on-chain/off-chain business process execution has gained interest in recent years. A BPMS vendor[1], proposed a set of on-chain/off-chain business process connectors [161]. Nevertheless, processes are intra-organizational and the system allows only monetary operations. In [64], a gateway enables a business process belonging to one organization running on an off-chain process execution engine to interact with heterogeneous blockchains. They propose a unique identifier to access the smart contract from the off-chain world. Nonetheless, off-chain business processes are intra-organizational and modeled in BPMN. The private and public business processes are connected using an array of integers stored on the blockchain ledger in [123]. Nonetheless, the approach leverages an orchestration scheme instead of a choreography

[1] https://www.bonitasoft.com/

one. To deal with the matters of privity and observability in a smart contract, authors in [108] define privity spheres to limit the read-access of data values to specific sets of participants. The most general public sphere allows the entire blockchain to read the data. Then this sphere can be reduced to a group of authorized participants. However, the management of fully off-chain tasks is not addressed in this work.

In summary, regarding flexibility efforts for business process modelization (RQ2.2), most blockchain-based collaborative processes focus on orchestration languages, especially on the BPMN standard, that do not encourage modelization flexibility (c.f., chapter basic concepts, Section 2.1). For studies using declarative languages, several declarative languages have been used such as DCR, XML, ADICO, ESML, or BPEL, efforts that call for further studies.

Furthermore, with our objective of a deployment and execution strategy that enforces a trustworthy separation of concerns (RQ1.1), some works propose a hybrid on/off-chain deployment approach preserving deployment trust, but without taking a view-based approach (hence, not addressing our separation of concerns objective). Among papers focusing on declarative choreographies, proposed solutions do not distinguish the partners' internal processes and the public view of the choreography when deployed to the chain.

3.3 Bringing flexibility to blockchain-based BPMS

As business environments such as laws, regulations, new competitors, or market strategies evolve continuously, the flexibility of the blockchain-based BPMS shall be investigated. In this section, we investigate related work focusing on two specific types of changes in order to address RQ2: (1) runtime control-flow changes (Section 3.3.1), and (2) partner flexibility in the case of procurement activities (Section 3.3.2).

3.3.1 Control-flow flexibility with runtime process instance changes

The change management criterion refers to the possibility to change the business process model instance once deployed on-chain. This change support is necessary due to the dynamic nature of processes: a study of change variability is necessary to reach modeling flexibility, as well as a study of the reaction to different change requests to achieve execution flexibility [114].

Change management at runtime in procedural processes has been studied in [66] where change propagation algorithms ensure the behavioral and structural soundness of choreography partners' private processes after the change. In [99, 67], authors consider the change negotiation phase but no mechanism is proposed to ensure that all partners have trustfully applied

the change. An approach integrates change to BPEL process choreographies [62]. Authors divide internal and public changes and formalize both changes. Participants only have access to their private views and define and submit changes that if valid are merged into other participants' private views. Nonetheless, changes are defined for the BPEL language, and change negotiation and propagation are not addressed in this work. In all these works, blockchain is not proposed as a trustworthy tool to carry on change management.

The CoBuP architecture enables change on the deployed process model to improve business process model flexibility [123]. Authors separate a BPMN model interpreter from the data structures defining the process model. By so doing, the logic workflow of the BPMN model is not statically encoded in the process instance smart contract but collected off-chain from the process model and dynamically added to it. Hence role changes are possible at runtime. Nonetheless, the authors focus on a BPMN orchestration approach only. Meanwhile, collaborative decisions on (1) late binding and un-binding of actors to roles in blockchain-based collaborative processes, (2) late binding of subprocesses, and (3) choosing a path after a complex gateway help support process model changes deployed on-chain [125]. A policy language enables the description of policy enforcement rules such as who can be a change initiator and who can endorse a change. However, the authors do not consider ADD/REMOVE/UPDATE change operations. Additionally, the private processes of roles are not considered, nor is the propagation of the effect of the new decisions over partners.

Change management has also been studied in DCR processes, mainly through runtime changes. The first efforts appear with the notion of DCR fragments where simple change/add/remove operations are implemented [141]. Authors follow the *build-and-verify* approach to apply incremental changes to the fragments. This approach consists of continuous iterations of (i) modeling, (ii) verifying that the new graph is free of deadlocks and livelocks, and (iii) executing until a further adaptation is required. Nonetheless, partner trust in the change propagation of DCR choreographies is not addressed in this work. In [145], authors use a *correct-by-construction* approach on running instances of DCR graphs. The structure underlying a DCR is a labeled transition system. Starting from a user-defined change, authors define a reconfiguration workflow. During the transition period, old requirements are disabled, and verified subpaths of activity executions are enabled. This setting holds until new requirements are verified. However, not every reconfiguration problem has a solution, and for every change, one has to build a new reconfiguration workflow, which is not easy for large models. Indeed, this approach requires heavy calculations to discover the verified subpaths. Authors extend this work with a fully automated technique based on formal specifications in [144]. Finally, in [49], authors use a set of rules ensuring the correctness of new instances of DCR graphs by design. Any new change operation

must respect these rules to prevent any unwanted behavior. Nonetheless, no blockchain support is provided in these works to carry on trustworthy change management.

3.3.2 Partner flexibility with runtime blockchain-based procurement

We now focus on partner flexibility in the case of procurement activities. This type of flexibility is necessary as partnerships are dynamic in cross-organizational processes: for example, a reallocation may be necessary to fit a temporary constraint (i.e., if a carrier cannot be on time for delivery) or to fulfill delivery standards that may vary from one delivery to another (i.e., for flowers or chemical goods).

Resource-binding refers to resource allocation mediation within or between organizations following a given policy [216], a mediation that cannot be avoided or broken. Compared to traditional allocation systems, blockchain-based allocation protocols unlock trustworthy process automation [181, 195, 121]. Indeed, the integrity of the protocol can be ensured as all carriers are considered before attributing a request, while historical data stored in the blockchain is tamper-proof. When a conflict occurs, the history of transactions can be retrieved and used as the single source of truth [156, 148]. Blockchain-based mappings can go one step further by managing the end-to-end service enactment autonomously and reliably.

Retrieved use cases are mainly anchored towards the autonomous delegation of computational tasks [212, 147, 142], and energy requests in smart grids [203, 117, 142]. The blockchain access control is mainly public, as use cases target public markets. It is also to note that two papers advocate permissioned blockchain access for privacy purposes [212, 117]. Thus resource-binding smart contracts often target public markets and choose a corresponding blockchain access control. Regarding the *binding Scheme* which summarises the existence of QoS binding rules and binding agreements (i.e., smart contract-based agreements), several rules, or constraints, have been proposed to choose or validate a resource binding match. Among such rules stand: (i) behavioral, structural, or execution constraints in [195], or (ii) roles, and statements in [121]. However, these notions are missing in the retrieved protocols. Moreover, to our knowledge, no QoS rules have been proposed in the literature to generate smart contract-based bindings. Moreover, the notion of a contractualization stage is not addressed except in [172] where the contractor and the resource negotiate the contract terms. Finally, several behavior control schemes have been implemented within resource-binding smart contracts. Incentives can foster service completion [15, 117, 147]. Escrow mechanisms moreover exist [169, 203, 147] to prevent the non-completion of an agreement and facilitate the settlement between the tenants once the service completes.

On the one hand, several papers investigate the use of blockchain and smart contracts for the autonomous allocation of services. In [151] a setup stage formalization is proposed for blockchain-based resource binding selection: it comprises the selection of the possible sets service request/resource provider, population with participants, and negotiation stages that will finally lead to contractualization, which can take the form of a digital agreement. QoS-based allocation rules are proposed as one type of selection strategy. Other papers leverage smart contracts for autonomous contractualization [203, 147], incentives to encourage service completion [117, 147], or delivery settlement [169]. However, these papers do not use oracles to compute QoS ratings.

On the other hand, several papers propose a design science research approach to investigate the use of blockchain applications. Two papers, [119, 3], focus respectively on developing blockchain-based IoT applications and smart-parking. However, the retrieved design principles do not specifically concern multi-criteria QoS-based procurement mappings. Two design science research papers, [143, 187], focus on the use of blockchain for logistics but do not focus specifically on the multi-criteria QoS-based procurement. The first paper focuses on the management of bills of lading via blockchain. Retrieved design principles are process digitization, tamper-proof storage, accessibility, and user authentication. The second paper focuses on food supply chains; the retrieved design principles are mainly related to data privacy and keeping sensitive data off-chain.

To summarize, regarding control-flow flexibility (RQ2.2), most related works consider changes in process orchestrations only [141, 49]. Additionally, approaches binding actors to roles in a process collaboration [125] currently push the burden of checking the transitive effect of new changes onto the new parties. This checking is likely done manually, which can lead to errors. Finally, even when the change propagation soundness is dealt with, the proposed approaches do not provide a mechanism that ensures choreography partners project the change and propagate it trustfully.

Moreover, regarding partnership flexibility (RQ2.1), the retrieved approaches propose resource-binding mechanisms based on price criteria. However, allocating a resource based on multiple criteria stored on-chain for a dynamic and trustworthy QoS-based allocation has not been addressed yet.

3.4 Bringing privacy to blockchain-based BPMS

In this section, we focus on two specific cross-organizational activities prone to distrust that can benefit from blockchain while demanding privacy: procurement and payment. The procurement stage consists of allocating a task to a service provider after examining service provider competitors.

Challenges focus on the objective and customized examination of competitors while preserving the confidentiality of sensitive data on the blockchain. The payment stage occurs at the end of the service delivery between two parties: it is also prone to trust requirements regarding the payment protocol, as well as confidentiality of payment content, as competitors share the blockchain transaction logs. We detail in the following the related work regarding these two stages.

3.4.1 Privacy preservation for on-chain offer comparison

We focus the study on sealed-bid auctions as in such auctions, bidders explicitly do not have access to bids from other competitors, and are requested to submit only one bid per auction.

Multi-party computation, alongside zero-knowledge-proof [77], can be used to conduct sealed-bid auctions. With zero-knowledge proofs, a prover can demonstrate knowledge of a piece of information without leaking information directly to the verifier. In the context of sealed-bid auctions, the non-interactive zero-knowledge proof variant, which consists of one-way communication between the prover and the verifier (c.f. zk-SNARKs [18]), is often used by bidders to compare offers pairwise in private channels without revealing the content of the bids [210, 22, 126]. They reveal the result of each smart contract comparison using evidence techniques with zero knowledge disclosure. The main limitation consists of the fact that bidders need to interact with each other. Hence, issues may arise if one bidder is not willing to participate, takes more time than needed, or if too many bidders need to interact. Additionally, the smart contract can reconstruct the bid ordering (e.g., from the most to the least expensive if the auctions are on a price), which reduces the privacy of bids.

In [51], a hybrid public/private blockchain scheme, combined with encryption technics, is proposed to carry on privacy-preserving auctions. The public blockchain is used to gather bids, and once the auction terminates, the auctioneer can access the content of the bids in the private blockchain. Such architecture answers the need for low auction costs and low latency. Nonetheless, the auctioneer must orchestrate and deploy the auction on the private chain. This may reduce the benefits of smart contracts as a trustworthy and autonomous third party, and reintroduces security and scalability downsides.

Other approaches use a smart contract to gather offers and compare them on-chain using a trusted execution environment or enclave [72, 103, 61, 71, 14, 192, 110]. A smart contract bridges the gap between customers and the enclave. Partial homomorphic encryption is used to gather offers in a confidential fashion on-chain, forwarded in an asymmetric fashion by bidders. An enclave then deciphers offers off-chain, as in [72]. In [103], a trusted execution environment computes allocations in a blockchain environment,

using an oracle to track node preferences. However, in these approaches, the enclave has access to the content of offers and this can result in a single point of failure. Additionally, partial homomorphic encryption, which is used e.g., in [71, 14] does not allow the combination of different operations (addition, subtraction, multiplication, division), which is necessary to carry on typical aggregation strategies used to compute a multi-objective comparison of offers.

3.4.2 On-chain privacy-preserving payments

In permissionless blockchains, private payments can be reached using mixing services. Cryptocurrency assets are mixed, anonymizing transactions and disconnecting the sender from the receiver. For example, Dash implements a decentralized mixing service coupled with a chaining facility on bitcoin [56]. Nonetheless, Bitcoin does not provide smart contracts: programmable payments are unavailable. Additionally, auditability is computationally intensive due to using an ahead-of-time decentralized trustless mixing strategy. SilentWhispers [130] also leverages a mixing facility as payment is processed using intermediary nodes. Additionally, temporary and long-term encryption keys are used for internode payments. By so doing, transactions are not linkable, i.e., it is not possible to backtrack transaction history, thus hampering auditability. Nonetheless, the key management scheme adds complexity to the payment scheme. Finally, Monero [150] proposes a mixing technique coupled with ring signature to hide both payment issuance and value. Nonetheless, an issue arises regarding the ring signature's size, which impacts the transaction sizes directly and processing speed. Several works tried to address this limitation, such as [196] and [101] but, similarly to Bitcoin, no smart contract facility is provided in Monero.

Alongside mixing strategies, encryption and zero-knowledge proofs protocols can enforce the privacy of payments.

Zerocash [184], based on bitcoin, uses zk-SNARKs to hide the payment's sender, receiver, and amount. Users pay each other privately while corresponding anonymized transactions are stored on-chain. This method requires a trusted setup, and transaction generation is computationally expensive (two minutes are necessary to generate a transaction using zero-knowledge proof according to [101]). Several extensions to Zerocash have been proposed in the literature to add an auditability layer: [74] adds an accountability layer to audit transactions and enforce spending limits, and [65, 30] propose partial anonymity for auditability using anonymity sets, coupled with El Gamal encryption and Schnorr zero-knowledge proofs. Audits can occur in a permissioned setting using verifiable public-key encryption. As a downside, the anonymity set depends on the set's size, and an encryption key management system is necessary to manage auditability. Additionally, all these solutions are based on Bitcoin, and partial payment is not provided.

Another strategy consists of hiding payment content using various

semi-homomorphic schemes (Pedersen commitment, Paillier, or El Gamal algorithms) and zero-knowledge proofs [139, 36, 173, 39]. Often, Pedersen commitments encode the amount and types of assets to be transferred. Zero-knowledge proofs show the validity of a transaction once it has been processed. For example, Zerocoin [139] converts bitcoins into zero coins that offer anonymity using Pedersen commitments and zero-knowledge proofs. Additionally, the zerocoins can be reconverted into bitcoins without any origin leakage. Nonetheless, Zerocoin does not offer auditing facilities and focuses on Bitcoin, hence does not offer smart contract facilities. MiniLedger focuses on a permissioned account-based blockchain, aiming for banks as end-users [36]. It uses Pedersen commitment and NIZK proofs to hide transaction values, senders, and recipients. Each bank-to-bank collaboration, i.e., a transaction from one bank to another, uses a unique encryption key to decipher their assets privately. Additionally, storage cost is independent of the number of transactions as MiniLedger leverages pruning technics.

Nonetheless, due to Pedersen commitment schemes, off-chain systems must transmit the openings of outgoing commitments, which complexifies the design and adds a layer of trust to senders. Additionally, if one user fails to open one commitment, its account will be unusable [39]. To circumvent the Pedersen commitment issue, Pretty Good Confidentiality (PCG)[39] proposes twisted El Gamal encryption and zero-knowledge proof. They offer to display transactions and public keys on the ledger for auditability. However, it is impossible to collect amounts or provide partial payments forcibly.

Several papers use privacy-preserving payment channels [230, 226, 131, 202, 85]. [230] proposes token payment schemes in a private blockchain consortium. A private blockchain acts as the interoperability domain issuing tokens. The banks use private payment channels built on this master blockchain to allocate transactions. Only the transaction hash, without further transaction details, is recorded in the master blockchain to preserve privacy. Hence, only banks involved in a transaction can access the details. As a limitation, private channels complexify the information system as each bank needs to open several channels to proceed with payments with other banks. In [226, 131], the Chameleon hash function guarantees that users cannot track payments under the condition that at least one intermediate payment node is honest. Nonetheless, an issue arises if intermediate nodes collude with each other. In [202], authors leverage Elliptic curve cryptography to hide transaction content. In [85], payment is processed off-blockchain in private payment channels using an untrusted intermediary. Nonetheless, in both approaches, auditability is not provided.

Another strategy uses trustworthy intermediaries to decipher encrypted payments and proceed to payments privately. In Bolt [79], the focus is set on intermediated payments carried on the Bitcoin blockchain. Privacy-preserving payment channel schemes are presented, including one leveraging trusted intermediaries. Solidus [34] offers a privacy-preserving protocol for

asset transfer in intermediated bilateral transactions. Banks act as mediators or proxies to hide transaction graphs and values in this system. They do so using ORAMs (oblivious random access machines, which prevent servers from learning about data [78]) coupled with zero-knowledge proof. Nonetheless, no dedicated auditing functionality is proposed, though banks can open the content of relevant transactions upon request.

In summary, regarding allocation privacy, a balance between auditability and full privacy is still challenging in the related work (RQ3.2). In [230], auditability is limited as only the transaction hash is accessible to auditors. Moreover, multi-party-computing reintroduces bidders' interactivity, which is not desired to carry on auctions [192]. Moreover, using trusted execution environments reintroduces a single point of failure as the content of offers is revealed to the enclave. Finally, fully homomorphic encryption has not been investigated in the context of on-chain auctions.

Regarding payment privacy (RQ3.3), several approaches offer fully privacy-preserving payment mechanisms such as mixing strategies, or payment channels. Indeed, mixing strategies have been proposed to anonymize transactions such as in Monero [150], ARRR, or Epic Cash, while private payment channels can ensure privacy and transaction scalability [230]. However, a fully privacy-preserving payment scheme is not desirable in our cross-organizational context, as partners already know each others. Instead, the following requirements arise : (1) auditability of the payment senders and recipients, (2) privacy of the payment value, (3) no computation intensive schemes for usability (i.e., inversely to schemes leveraging zero-knowledge proofs such as PIVX or Solidus [34]), (4) leverage of smart contracts for automation purposes. Additionally, in an industrial context, we hypothesise that the trust issue is more on the process managmeent governance than on banks, hence banks can be referred to as trusted entities for payment management [79, 34].

3.5 Comparison and Discussion

3.5.1 Evaluation Criteria

Based on the challenges discussed in the previous chapter, we propose the following criteria to evaluate related work revolving around blockchain-based business process management systems:

- **(E1)**: Trustworthy support using blockchain. This criterion is deduced from the research sub-question RQ2.1 (section 1.3.2);
- **(E2)**: Declarative modeling language. This criterion is deduced from the research sub-question RQ2.3 (section 1.3.2);
- **(E3)**: Trustworthy and privacy-preserving separation of concerns

during deployment and execution. This criterion is deduced from the research sub-questions RQ1.1 (section 1.3.1) and RQ3.1 (section 1.3.3);

- **(E4)**: Process instance control-flow change. This criterion is deduced from the research sub-question RQ2.4 (section 1.3.2);
- **(E5)**: Dynamic and customizable actors allocation task decision making based on a smart contract, and leveraging past customers history This criterion is deduced from sub-question RQ1.2 (section 1.3.1) and sub-question RQ2.2 (section 1.3.2);
- **(E6)**: Privacy and auditability of the allocation stage by confidentially computing sensitive metrics. This criterion is deduced from the research sub-question RQ3.2 (section 1.3.3);
- **(E7)**: Privacy and auditability of the payment stage. This criterion is deduced from the research sub-question RQ3.3 (section 1.3.3).

3.5.2 Summary

Table 3.1 summarizes the features of the presented approaches according to previously introduced criteria E1-E7.

For the first criterion (E1), the notion of a trustworthy inter-organizational business process support using blockchain is addressed in most related work except for several papers where blockchain is cited as a possible solution among others ([194]), or not considered as it was not the direct focus of the paper ([66, 67, 99, 62, 141, 145, 144, 49]). We present these works as they match some of our other criteria.

For the second criterion (E2), most blockchain-based collaborative processes cited in the literature do not consider declarative choreographies. A majority focuses on orchestration languages, especially on BPMN ([214, 201, 112, 120, 64, 123, 108, 66, 67, 99, 123, 125]). Moreover, no standard exists yet regarding declarative languages, and several declarative languages such as DCR, XML, ADICO, ESML, or BPEL, call for further study in the blockchain context.

For the third criterion (E3), we focus on the challenge of the trustworthy and privacy-preserving separation of concerns during deployment and execution. We classify related work with respect to view-based approaches and the existence of a hybrid on/off-chain deployment. Some works propose a hybrid on/off-chain deployment approach preserving deployment trust, but without taking a view-based approach [161, 64, 123, 108]. Among papers focusing on declarative choreographies, proposed solutions do not distinguish the partners' internal processes and the public view of the choreography when deployed to the chain.

Limitations identified in the related work for (E2) and (E3) hence help refine Objective 1 (c.f., Section 1.4.1): *identifying a business process modeling language that is flexible and a business process model that integrates a separation of concerns by design*) with (i) a focus on a declarative language and (ii) by means of an on-chain choreography management.

For the fourth criterion (E4), we investigate process instance control-flow change by looking at the presence of choreography, propagation soundness, and propagation trustworthiness. Most of the related work considers changes in process orchestrations only [141, 49] and even when authors bind actors to roles in a process collaboration [125], they do not consider the transitive effect of new changes on the partners' parts of the process. Moreover, even when the change propagation soundness is dealt with, no mechanism is proposed to ensure that choreography partners project the change and propagate it trustfully. Hence these identified limitations validate Objective 3 (*making the public view upgradable and open to changes upon participants' requests at runtime.*)

For the fifth criterion (E5), we focus on the dynamic and customizable actors allocation task decision-making based on a smart contract by looking at the notions of multi-criteria filtering and sorting and smart contract e-contracting. The retrieved approaches propose resource-binding mechanisms based on price criteria. However, allocating a resource based on multiple criteria stored on-chain (some to be filtered and some to be sorted) to allow for a dynamic and trustworthy QoS-based allocation has not been addressed yet. Hence, this research gap validates Objective 4 (*carrying on runtime allocation by leveraging blockchain- stored tamper-proof data.*)

For the sixth criterion (E6), we focus on allocation privacy by looking at the following criteria: allocation privacy, bidder non-interactivity, no trusted entity access to clear data, and no auctioneer orchestration (as we aim at full automation). With [230], auditability is provided but limited as only the transaction hash is accessible to auditors. Carrying on privacy-preserving auctions on the blockchain with multi-party-computing reintroduces bidders' interactivity, which is not desired to carry on auctions [192]. Additionally, using private blockchains lowers scalability and does not leverage smart contract automation benefits. Moreover, using trusted execution environments reintroduces a single point of failure as the content of offers is revealed to the enclave. Hence, we validate Objective 5 (*carrying on trustworthy privacy-preserving on-chain auctions while preserving the auditability of the smart- contract-based decision making*).

For the seventh criterion (E7), which focuses on the payment stage, we investigate post-payment privacy, programmable payment, and payment auditability. Several approaches use private payment channels ([230, 226, 131, 202, 85]) to ensure privacy and scalability when transactions increase, but they offer little [230] to no auditability. In [184, 94, 139], semi-homomorphic encryption and zero-knowledge-proof schemes are combined to hide the

transaction value, sender, and receiver identity. Following the same approach, a set of papers ([149, 9, 74, 65, 30, 36, 39]) also provide auditability functionalities. Nonetheless, zero-knowledge proofs and semi-homomorphic encryption come with computation issues. Additionally, semi-homomorphic encryption comes with complicated encryption key management. Mixing strategies such as in [56, 150, 196, 101] for UTXO models, and [130] for account-based models have been proposed to anonymize transactions. Nonetheless, no focus on auditability is provided in these papers, except for [56] which proposes a retroactive auditability function. The auditability affordance is computationally expensive, and Dash does not offer smart contracts, hence no partial or programmable payment. In [79, 34], banks are leveraged as trustworthy intermediates to carry on on-chain payments. Cryptography technics such as zero-knowledge proof in [34] ensure privacy, but auditing functions are not provided. Hence, we validate Objective 6 (Providing a trustworthy, privacy-preserving, and scalable on-chain payment mechanism).

3.6 Conclusion

In this chapter, we present the current efforts devoted to building blockchain-based cross-collaboration BPMS. For this purpose, we conduct a survey aiming to determine the challenges linked to trusted decentralized process monitoring, identify the proofs-of-use developed in the literature and compare them to outline existing research gaps. We classify related work according to the following analytical perspectives: (i) blockchain-based business process deployment and execution, (ii) control-flow and partner flexibility, and (iii) allocation and payment privacy.

Hence, in this chapter, the following research gaps appear: (i) a lack of constraint-based process modeling in the blockchain context, (ii) a lack of process flexibility in the choreography context (behavioral and actor flexibility), (iii) a lack of approaches preserving process privacy and auditability in a blockchain context.

In this thesis, we propose to address these challenges by:

- Taking a declarative choreography modeling approach (c.f., Section1.4.2, contribution i). Indeed, most existing works use an imperative paradigm such as BPMN. However, we chose to model choreographies with a declarative language that abstracts the control-flow through a set of rules or constraints [63, 76], namely Dynamic-Condition-Response (DCR) graphs [190, 93]. We believe that the declarative paradigm corresponds to the dynamic nature of choreography interactions, as business modelers cannot predefine all the execution paths of a model in constant evolution. Only essential constraints are specified in the model. Additionally, choreographies help separate clearly the public

and private views, hence easing the separation of concerns for business process model deployment, execution, or runtime change. On/off-chain deployment and execution strategies for constraint-based models of choreographies have not been addressed in the literature, as well as runtime change of the corresponding control flow instance.

- Enhancing runtime flexibility by investigating control-flow and partnerships flexibilities (c.f., Section1.4.2, contribution ii). Indeed, research on the flexibility of blockchain-based declarative business process management is scarce, especially regarding strategies concerning a trustworthy change proposition. What is more, research on a QoS-based runtime partner allocation, which leverages blockchain, is lacking. Hence, we propose two mechanisms in this thesis to foster blockchain-based business process management flexibility, a flexibility that is necessary for the system to adapt to moving environments.

- Preserving sensitive metrics confidentiality while aiming at full smart contract automation (c.f., Section1.4.2, contribution iii). In most related work, sensitive metrics confidentiality comes at the expense of auditability limitations, or partner interactions, hence reducing automation benefits coming from smart contracts. In the context of cross-organizational processes, both confidentiality and auditability of the allocation or payment stages are paramount to foster the adoption of such technology.

Hence the three following chapters present research works aiming at answering these objectives by respectively addressing (i) blockchain-based on/off-chain deployment and execution, (ii) flexibility of the control-flow and partners allocation, and (iii) privacy of allocation and payment.

Table 3.1: Related works categorization according to the evaluation criteria E1-E7. (*)=no experimentation, (ND)= not detailed, (NA)=not applicable

Papers	**E1**	**E2**	**E3**		**E4**			**E5**		**E6**				**E7**		
	Blockchain	**BP modeling**	**View-based**	**Deployment**	**Choreography**	**Propagation soundness**	**Propagation trust**	**QoS allocation**	**On-chain contracting**	**Allocation Privacy**	**Bidder interactivity**	**TEE access to clear data**	**Auctioneer orchestration**	**Post-payment privacy**	**Programmable payment**	**Payment auditability**
[229]	x	-	-	on-chain	NA	NA	NA	NA	NA	-	NA	NA	NA	-	NA	-
[223, 165]	x	-	-	on-chain	NA	NA	NA	-	-	NA	NA	NA	NA	-	NA	-
[80]	x	-	-	on-chain	NA	NA	NA	-	-	x	NA	NA	NA	-	NA	-
[153, 155]	x(*)	ESML	x	NA	x	NA	NA	-	x	-	NA	NA	NA	-	NA	-
[214]	x	BPMN	-	on-chain	-	NA	NA	NA	NA	-	NA	NA	NA	-	NA	-
[201]	x	BPMN	x	on-chain	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[69]	x	ADICO	-	ND	-	-	-	-	-	-	NA	NA	NA	-	NA	-
[11]	x	BCRL	-	on-chain	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[127]	x	DCR	-	on-chain	-	NA	NA	-	NA	NA	NA	NA	NA	NA	NA	NA
[96]	x(*)	LTL	-	on-chain	-	NA	NA	-	-	-	NA	NA	NA	-	NA	-
[193]	x(*)	AC	-	on-chain	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Continued on next page

Table 3.1 – continued from previous page

Papers	E1	E2	E3		E4			E5		E6				E7		
	Blockchain	BP modeling	View-based	Deployment	Choreography	Propagation soundness	Propagation trust	QoS allocation	On-chain contracting	Allocation Privacy	Bidder interactivity	TEE access to clear data	Auctioneer orchestration	Post-payment privacy	Programmable payment	Payment auditability
[138]	x	AC	-	on-chain	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[28]	x	XML	-	on-chain	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[60]	x	XML	x	on-chain	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[107, 37]	-	NA	x	NA	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[112]	x	BPMN	x	ND	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[194]	ND	ND	x	NA	x	NA	NA	NA	NA	x	NA	NA	NA	NA	NA	NA
[120]	x	BPMN	-	on-chain	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[176]	x	NA	-	on-chain	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[161]	x	NA	-	hybrid	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[64, 123]	x	BPMN	-	hybrid	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[108]	x	BPMN	-	hybrid	-	-	NA	-	NA	NA	NA	NA	NA	NA	NA	NA
[22, 126]	x	-	-	-	-	-	-	-	-	x	x	NA	-	NA	NA	NA
[51]	x	-	-	-	-	-	-	-	-	x	-	NA	x	NA	NA	NA

Continued on next page

Table 3.1 – continued from previous page

Papers	E1	E2	E3		E4			E5		E6				E7		
	Blockchain	BP modeling	View-based	Deployment	Choreography	Propagation soundness	Propagation trust	QoS allocation	On-chain contracting	Allocation Privacy	Bidder interactivity	TEE access to clear data	Auctioneer orchestration	Post-payment privacy	Programmable payment	Payment auditability
[72, 103, 14, 61, 71, 110, 192]	x	-	-	-	-	-	-	-	-	x	-	x	-	NA	NA	NA
[130]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	NA	-
[139, 150, 196, 101, 184]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	-	-
[9]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	x	x
[56, 149, 74, 65, 30]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	-	x
[36]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	NA	x
[39]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	-	NA	x
[230]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	x	x
[226, 131]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	x	-
[202, 85, 79, 34]	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	x	-	-

Continued on next page

Table 3.1 – continued from previous page

Papers	E1	E2	E3		E4			E5		E6				E7		
	Blockchain	BP modeling	View-based	Deployment	Choreography	Propagation soundness	Propagation trust	QoS allocation	On-chain contracting	Allocation Privacy	Bidder interactivity	TEE access to clear data	Auctioneer orchestration	Post-payment privacy	Programmable payment	Payment auditability
[66, 67, 99]	-	BPMN	x	NA	x	x	-	NA	NA	NA	NA	NA	NA	NA	NA	NA
[62]	-	BPEL	x	NA	x	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[123]	x	BPMN	-	on-chain	-	x	x	NA	NA	NA	NA	NA	NA	NA	NA	NA
[125]	x	BPMN	-	on-chain	-	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[141, 145, 144, 49]	-	DCR	-	NA	-	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[151]	x	NA	NA	NA	NA	NA	NA	ND	x	-	NA	NA	-	NA	NA	NA
[203, 147, 117, 169, 119, 3, 143, 187]	x	NA	NA	NA	NA	NA	NA	-	x	-	NA	NA	-	NA	NA	NA
Our approach	**x**	**DCR**	**x**	**hybrid**	**x**	**x**	**x**	**x**	**x**	**x**	**-**	**-**	**-**	**x**	**x**	**x**

Chapter 4

Declarative Choreography Management with Blockchain

Contents

4.1 Introduction

As mentioned in Section 1.1, a cross-organizational process can be defined as a process scattered across different organizations. It comprises private processes carried out by individual partners, where internal data such as business data, actors, or business entities, should not be visible to the other

partners. It also includes a public process where several partners collaborate in a coordinated way.

In this setting, all partners should trust the execution state of the public process. Hence, a trade-off between ensuring the privacy of partners' private processes and exposing the public process thus arises. Model flexibility is also at stake, as processes are dynamic: partners should be able to change their internal processes without impacting the public process [37]. Thus, the following questions arise:

1. *How to carry out a separation of concerns that preserves the privacy of the private processes and trust of the public process for the deployment and execution of choreography processes in blockchain?* (c.f., Section 1.3, RQ1.1)

2. *How to model the flexibility imperatives of cross-organizational business processes?* (c.f., Section 1.3, RQ2.2)

3. *How to ensure a trustworthy access-control of business process activity data stored in smart contracts in blockchain-based cross-organizational processes?* (c.f., Section 1.3, RQ3.1)

In the literature, as mentioned in the chapter basic concepts, subsection 2.1.3, business process choreographies answer the need for such separation of concerns by clearly specifying coordination tasks [2, 112]. In addition, the public process is shared between participants to limit privacy leakages. Meanwhile, private views hold the set of (1) internal tasks of a particular partner not disclosed to the other partners and (2) communication tasks in which this partner is involved, i.e., the projection of the public view over this partner [2]. However, the trustworthy execution of the public view remains challenging as it is often managed centrally [112].

Blockchain has been leveraged in the literature as a trustworthy coordination mechanism for collaborative business processes [214, 112]. As mentioned in the related work chapter, subsections 3.2.3 and 3.2.4, in [214], a smart contract manages the public workflow of an orchestration. However, in this approach, the execution of private tasks off-chain is only mentioned, and the inner mechanism has not been detailed further. Additionally, in [112], the smart contract is used to manage a choreography public view, and so doing, enforcing the order of messages. Nonetheless, this work suggests a private/public separation, but only the public view mechanism is implemented. Additionally, there is no on/off-chain enforcement of projections during the deployment of the process instance.

Thus, to our knowledge, none of the retrieved works addresses the trustworthy deployment of choreographies. This deployment remains challenging as private information should not be shared between partners at design or runtime. Moreover, none of the retrieved works proposes a detailed mechanism for executing projections using a hybrid on/off-chain mechanism.

In this chapter, we contribute to the literature through a unified solution for designing and executing business process choreographies in a hybrid on/off-chain fashion.

The first contribution of this chapter is presented in Section 4.4. We propose a mechanism for deploying the global process, which offers trustworthiness while preserving the separation of concerns. Participants build the global process incrementally from a public view stored in a smart contract during deployment. Each participant will compute off-chain its role projection comprising public events where she is involved, and private events are kept off-chain for privacy concerns. This way, private control flows remain in the participants' process engines, while blockchain systems ensure a tamper-proof public view. The blockchain cannot access private events; aggregating all role projections will render the global process.

The second contribution, presented in Section 4.5, is a hybrid on/off-chain mechanism for executing cross-organizational choreographies. The roles execute their internal tasks off-chain in their local process execution engine. Meanwhile, a smart contract manages public interactions. When the smart contract receives an interaction request initiated from one of the roles (sender or receiver(s)), it executes the task and communicates its state back. The roles update their private states accordingly. Hence, we achieve a trustworthy separation of concerns preserving partners' private processes' privacy.

Most existing works use an imperative paradigm such as BPMN. However, we chose to model choreographies with a declarative language that abstracts the control-flow through a set of rules or constraints [63, 76], namely Dynamic-Condition-Response (DCR) graphs [190, 93]. We believe that the declarative paradigm corresponds to the dynamic nature of choreography interactions, as business modelers cannot predefine all the execution paths of a model in constant evolution. Only essential constraints are specified in the model.

4.2 Basic concepts

In the following section, we introduce basic concepts related to DCR graphs (Section 4.2.1) and DCR choreographies (Section 4.2.2) that will be used in the rest of the chapter.

4.2.1 DCR graphs

DCR graphs are one of many declarative business process modeling languages whose formalism is presented in [93]. We refer to the following definition (cf [93]):

Definition 4.2.1 (**DCR Graph**). A DCR graph G is a tuple (E, M, L, f, $\longrightarrow \bullet$, $\bullet \longrightarrow$, $\longrightarrow \diamond$, $\longrightarrow +$, $\longrightarrow \%$), where:

- E is a set of events
- $M = (In, Pe, Ex) \subseteq E \times E \times E$ is a marking
- L is a set of labels
- $f : E \longrightarrow L$ is a labelling function
- $l \subseteq E \times E$ for $l \in \{\longrightarrow \bullet, \bullet \longrightarrow, \longrightarrow \diamond, \longrightarrow +, \longrightarrow \%\}$ are relations between events.

Following Definition 4.2.1, business processes are modelled as a set of *events* E linked together with *relations*[1]. Markings M capture the graph's state at runtime by referring to the triplet (currently included *events* In, currently pending *responses* Pe, previously executed *events* Ex). Relations model in a loosely fashion the constraints linking two *events*. The end-user can enact any enabled activity at any time and more than one time during a process instance execution.

DCR graphs hold five types of relations. Two relations, *condition* and *milestone*, model pre-execution constraints. They restrain the enactment of an *event*. The *condition* relation from *GetOrder* to *CallDriver* in Fig. 4.1a implies that *GetOrder* must have been launched, not necessarily terminated, for *CallDriver* to start. The *milestone* relation from *Shipping* to *CheckOrder* implies that *Shipping* must be finished for *CheckOrder* to start. Three relations translate the effects of an *event* execution to the remaining activity markings. The *inclusion* relation from *Accept* to *Pay* states that the execution of *Accept* unlocks *Pay*. On the opposite, the *exclusion* relation from *Reject* to *Pay* states that the enactment of *Reject* forbids the execution of *Pay*. Finally, the *response* relation from *Accept* to *Pay* sets *Pay* to pending when *Accept* is executed (i.e., *Pay* waits for completion).

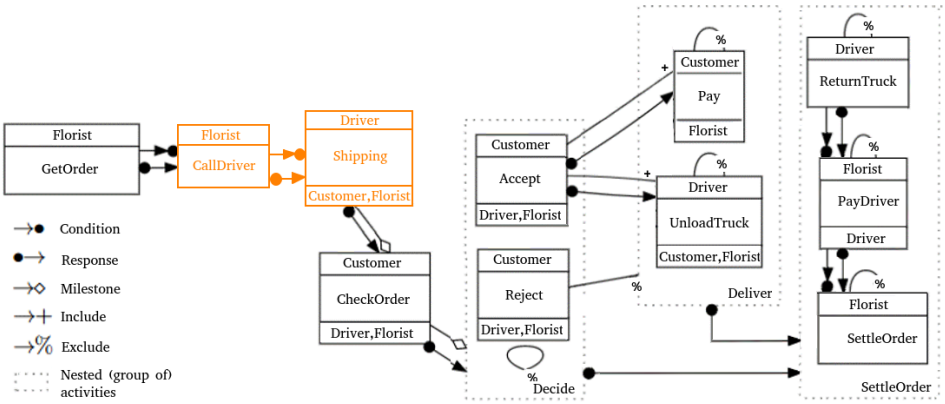
In the manuscript, we refer to Γ as the set of *relations* of the graph: Γ=$\{\longrightarrow \bullet, \bullet \longrightarrow, \longrightarrow \diamond, \longrightarrow +, \longrightarrow \%.\}$.

4.2.2 DCR choreography

A *DCR choreography* [93, 167] models and executes DCR graphs in a distributed way. It comprises choreography *events* that ease coordination between independent entities and internal events. We reconcile the definition of a DCR choreography proposed in [93] and formalize it as follows:

Definition 4.2.2 (**DCR choreography**)**.** A DCR choreography Φ is a triple (G, I, R) where G is a DCR graph, I is a set of interactions, and R is a set of roles. An interaction i is a triple (e, r, r') in which the event e is initiated by the role r and received by the roles $r' \subset R \setminus \{r\}$. For an event $e \in E$, e.type is the type of the event, e.type $\in \{\epsilon, \gamma\}$, where (i) *epsilon*

[1] A DCR event is equivalent to a BPMN activity.



(a) DCR graph of Flower Delivery *(in orange, a DCR subgraph)*

(b) Projection of the orange sub-graph over *Driver (Driver private view)*

(c) Projection of the orange sub-graph over *Florist (Florist private view)*

Figure 4.1: DCR graph, and projections of a DCR graph chunk (in orange).

denotes the set of internal events in G, i.e., events having one initiator $r \in R$ and (ii) *gamma* are the set of interactions in G ($gamma = I$).

Figure 4.1a represents the DCR choreography of the delivery process presented in the introduction in BPMN. The process involves three participants: Customer, Florist, and Driver. In this example, *Shipping* is a choreography interaction sent by Driver and received by Florist and Customer. *GetOrder* is an internal event of Florist.

4.3 Motivating Example

Table 4.1 illustrates several executions of the DCR graph instance of the delivery process (Figure 4.1).

Each column corresponds to an event marking of the graph in the form (included, pending, executed). Each line stands for an event query triggered. For example, initially, no event is executed or pending. The event *GetOrder* is included in the execution set. Thus the initial marking of GetOrder is (1,0,0). Upon executing *GetOrder*, the markings of *GetOrder* and *CallDriver* are updated. *GetOrder* is now executed while *CallDriver* becomes included and pending. Hence corresponding markings are (1,0,1) and (1,1,0).

Each participant controls the internal and choreography events where she is involved. We define this set of events as her private view. For example,

Table 4.1: Evolution of the markings of the DCR graph in Figure 4.1a

	Markings (included,pending,executed)							
	GetOrder	CallDriver	Shipping	CheckOrder	Accept	Reject	Deliver	SettleOrder
(init)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
GetOrder	(1,0,1)	(1,1,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
CallDriver	(1,0,1)	(1,0,1)	(1,1,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
Shipping	(1,0,1)	(1,0,1)	(1,0,1)	(1,1,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
CheckOrder	(1,0,1)	(1,0,1)	(1,0,1)	(1,0,1)	(1,1,0)	(1,1,0)	(0,0,0)	(0,0,0)
Decide (Accept)	(1,0,1)	(1,0,1)	(1,0,1)	(1,0,1)	(1,0,1)	(0,0,0)	(1,1,0)	(0,0,0)
Deliver	(1,0,1)	(1,0,1)	(1,0,1)	(1,0,1)	(1,0,1)	(0,0,0)	(1,0,1)	(1,1,0)
SettleOrder	(1,0,1)	(1,0,1)	(1,0,1)	(1,0,1)	(1,0,1)	(0,0,0)	(1,0,1)	(1,0,1)

the sub-graph in orange in Figure 4.1a depicts the global view of a process involving three partners: Florist, Driver, and Customer. Figure 4.1b and Figure 4.1c depict respectively the private views over Driver and Florist.

Requirements arise when dealing with the execution of such choreography. The activities for which some of the participants are not interested in (e.g., *SettleOrder*) or confidential (e.g., *GetOrder*) must be kept private. The public view must express by design the information and requirements needed to execute the workflow. Moreover, public activities must be tamper-proof, and the execution flow fulfilled to keep on with the agreed-upon flow. The system must offer integrity by design. If a claim occurs, the system becomes the single source of truth.

Former works on private and public views have been proposed before blockchain emergence [37, 107]. Separation of concerns is reached by separating public and private views. However, trust in the execution of the public view is still needed. Blockchain brings two interesting properties to our research: decentralization and tamper-proof logs. Thus, the public view of a business process could be completely decentralized by design while ensuring trust through the tamper-proof logs property.

Nonetheless, two questions arise in this setting to preserve the separation of concerns between participants, which we address in the following sections.

The first question, addressed in Section 4.4, concerns the deployment of the global process in each local BPMS. The deployment shall not be managed by a centralized entity that would then upload the public view on-chain. Otherwise, the trust issue would rise again. Additionally, the question of ensuring that projections are completed off-chain while avoiding any information leakage remains.

The second question, addressed in Section 4.5, concerns the execution of the global graph. The smart contracts act as an entry point to ensure the correctness of the execution of the public view. The mechanism managing the two-sided public/private execution of tasks needs to be defined to ensure that each participant can manage its projection in a trustworthy fashion.

4.4 Design time: Generating Public and Private Views

This section presents the hybrid on/off-chain protocol developed to generate the partners' view-based projections. We first introduce concepts related to public and private views of a DCR choreography (c.f., Section 4.4.1). We then present the protocol, which comprises two steps, first the translation from the DCR process model of the public and private views to a bitvector representation (c.f., Section 4.4.2), and then the hybrid on/off-chain generation of views (c.f., Section 4.4.3).

Regarding the hybrid on/off-chain generation of views, a smart contract comprising (1) DCR execution constraints rules and (2) a list of workflows initially empty is used to manage DCR graph instances. Once all participants agree on the public process model at design time, one of the participants (chosen randomly or nominated by its peers) instantiates the DCR graph instance in the smart contract. The DCR graph instance comprises the relation matrices and markings of the public view (cf. Section 4.2). It also comprises the list of the role addresses linked to each activity. Roles (e.g., Driver, Florist, or Customer) are each assigned to a public blockchain address. Finally, the DCR graph instance comprises the IPFS hash of the textual input. The hash serves as a unique identifier for the workflow. Then, each participant computes her private view by combining the public view with its internal events. The output is a bitvectorized DCR graph. These private views constitute the entry point for the hybrid runtime execution. Finally, once the generation of role projections is fulfilled, the smart contract unlocks the process instances for execution.

4.4.1 Public and private views of a DCR choreography

Let (G, I, R) be a DCR choreography (cf. **Definition 4.2.2**), we define this DCR choreography through its public view G_γ and private views G_r, $\forall\ r \in R$, which are derived from G. We formalize G_γ and G_r, $\forall\ r \in R$ as follows:

Definition 4.4.1 (Public view)**.** G_γ is a tuple $(E_\gamma,\ M_\gamma,\ L_\gamma,\ f_\gamma,\ \longrightarrow \bullet_\gamma,\ \bullet \longrightarrow_\gamma,\ \longrightarrow \diamond_\gamma,\ \longrightarrow +_\gamma,\ \longrightarrow \%_\gamma)$, where:

1. $E_\gamma = \{e \in I\}$

2. $M_\gamma = (In_\gamma, Pe_\gamma, Ex_\gamma)$ where $In_\gamma = In \cap E_\gamma$, $Pe_\gamma = Pe \cap E_\gamma$, and $Ex_\gamma = Ex \cap E_\gamma$
3. $f_\gamma(e) = f(e)$
4. $L_\gamma = img(f_\gamma)$
5. $\longrightarrow \bullet_\gamma = \longrightarrow \bullet \cap ((\longrightarrow \bullet\ E_\gamma) \times E_\gamma)$
6. $\bullet \longrightarrow_\gamma = \bullet \longrightarrow \cap ((\bullet \longrightarrow\ E_\gamma) \times E_\gamma)$
7. $\longrightarrow \diamond_\gamma = \longrightarrow \diamond \cap ((\longrightarrow \diamond\ E_\gamma) \times E_\gamma)$
8. $\longrightarrow +_\gamma = \longrightarrow + \cap ((\longrightarrow +\ E_\gamma) \times E_\gamma)$
9. $\longrightarrow \%_\gamma = \longrightarrow \% \cap ((\longrightarrow \%\ E_\gamma) \times E_\gamma)$

   Hence, $l_\gamma \in \{\longrightarrow \bullet_\gamma, \bullet \longrightarrow_\gamma, \longrightarrow \diamond_\gamma, \longrightarrow +_\gamma, \longrightarrow \%_\gamma\}$

In our motivating example, the public view of the DCR choreography depicted in Figure 4.1a comprises $E_\gamma = \{Shipping, CheckOrder, Accept, Reject, Pay, UnloadTruck, PayDriver\}$, the set of markings M_γ, the set of labels of the events (e.g., "Shipping" or "CheckOrder"), associated with the labeling function f_γ, and the set of interactions linking these events.

Definition 4.4.2 (Private views). For a role $r \in R$, G_r = a tuple (E_r, M_r, L_r, f_r, $\longrightarrow \bullet_r$, $\bullet \longrightarrow_r$, $\longrightarrow \diamond_r$, $\longrightarrow +_r$, $\longrightarrow \%_r$), where:

1. $E_r = \{e \in E \mid Initiator(e) = r \cup Receiver(e) = r\}$
2. $M_r = (In_r, Pe_r, Ex_r)$ where $In_r = In \cap E_r$, $Pe_r = Pe \cap E_r$, and $Ex_r = Ex \cap E_r$
3. $f_r(e) = f(e)$
4. $L_r = img(f_r)$
5. $\longrightarrow \bullet_r = \longrightarrow \bullet \cap ((\longrightarrow \bullet\ E_r) \times E_r)$
6. $\bullet \longrightarrow_r = \bullet \longrightarrow \cap ((\bullet \longrightarrow\ E_r) \times E_r)$
7. $\longrightarrow \diamond_r = \longrightarrow \diamond \cap ((\longrightarrow \diamond\ E_r) \times E_r)$
8. $\longrightarrow +_r = \longrightarrow + \cap ((\longrightarrow +\ E_r) \times E_r)$
9. $\longrightarrow \%_r = \longrightarrow \% \cap ((\longrightarrow \%\ E_r) \times E_r)$

   Hence, $l_r \in \{\longrightarrow \bullet_r, \bullet \longrightarrow_r, \longrightarrow \diamond_r, \longrightarrow +_r, \longrightarrow \%_r\}$

Algorithm 1: Marking Vectorization of a private view

Data: $G_r = (E, l)$
Result: the list of included, executed, and pending marking vectors

```
1  Function initializeMarkings(E, l):
2      var len ← length(E);
       // INITIALIZE VECTORS
3      var In ← Vector(size : len);
4      var Pen ← Vector(size : len);
5      var Ex ← Vector(size : len);
       // DETECT INITIALLY INCLUDED EVENTS
6      var i=0;
7      forall e ∈ E.ϵ do
8          var hasPreceedingEvent ← FALSE;
9          forall rel ∈ l do
10             if rel.target == e then
11                 hasPreceedingEvent ← TRUE;
12                 break;
13         if not hasPreceedingEvent then
14             In[i] ← 1 ; // NO PRECEEDING EVENTS
15         i=i+1;
16     return [In, Pen, Ex]
17 End Function
```

In our motivating example, three private views exist, one for each choreography partner (Driver, Customer, and Florist). For the sub-graph in orange in Figure 4.1a, Figure 4.1b and Figure 4.1c depict respectively the private views over Driver and Florist. E_{Driver} comprises the set of events where Driver is involved (e.g., public events such as *Shipping*, and private events such as *ReturnTruck*), their markings, their labels, and the interactions linking these events.

4.4.2 Translating DCR graphs into bitvectors

The public and private views are initially described as textual input following the semantics prescribed in [47]. The reader can find input examples of the delivery process in Appendix 7.3.1. Additional examples can be found in the source code repository of our prototype, in the folder *dcrInputs*[2]. We translate each view into a bitvector representation for execution in the

[2] https://archive.softwareheritage.org/swh:1:dir:211c6bdb1ce9f256c363caee54a56f53ada05d9b;origin=https://github.com/tiphainehenry/hybridChoreo;visit=swh:1:snp:6376000436aeaf872f39fb5f1a9d5aaa417c5cea;anchor=swh:1:rev:a5dd01e0ed0cb034c1e35daa0e6b3c6e08ff6d97

off-chain and on-chain process execution engines [214, 127]. We describe in the following paragraph the approach to computing such representation.

The bitvector representation comprises (1) the five relation matrices of the DCR graph and (2) the three markings of the graph. The five relation matrices are computed out of an input view. For each relation $[e_i \longrightarrow e_j]$, the item a_{i_j} in the relation matrix is set to one. Besides, we generate the three initial bitvector markings of the graph (Algorithm 1, lines 3-5). The *executed* and *pending* initial markings are set to zero as no event has been executed yet. The *included* state of the event is set to one if it has no pre-execution *condition* (Algorithm 1, lines 6-15). We now illustrate the Florist projection bitvectorisation. First, we generate the five relation matrices. In the Florist private projection, a *condition* relation links *CallDriver* and *Shipping*. Thus, *Condition*$[id_{CallDriver}, id_{Shipping}]$*=1*. The same protocol follows for each relation of the graph. We then compute the three markings of the projection. The *pending* and *executed* bitvectors are filled with eleven zeros (one for each event of $E_{Florist}$). The Florist included bitvector is filled similarly, except for *GetOrder* which is set to one (no pre-condition).

4.4.3 Hybrid on/off-chain generation of views

The generation of views comprises two steps: the on-chain public view first and private views.

The public view managed on-chain, G_γ, is the DCR graph consisting of the set of choreography events, i.e., events having one or many receivers and their relations, that model participants' interactions. A representative of all participants first generates the approved bitvector representation of the public view (Figure 4.2, step1). The public view consists of choreography events and their relations. Each role has a public blockchain address, and choreography events are mapped to a sender role. Moreover, the representative saves the textual public view input to IPFS to keep track of it and saves the hash into the smart contract. The smart contract locks the process instance while waiting for each participant projection (Figure 4.2, step2). Initially set to zero, a variable named σ, keeps track of the number of projections realized. The process instance is unlocked for execution when σ equals the number of participants. For example, the public events of Figure 4.1a are {*Shipping, CheckOrder, Accept, Reject, Pay, UnloadTruck, PayDriver*}. The smart contract stores these events and relations where at least two public events are involved. The internal events of Figure 4.1a ($\{ReturnTruck\}$ for Driver, or $\{GetOrder, CallDriver, SettleOrder\}$) for Florist are kept off-chain.

Once the public view populates the smart contract, each participant fetches it (Figure 4.2, step4). The private projection is generated by extracting all the events of G_γ where the participant is an initiator or a receiver in a choreography event. We conjointly extract relations connecting these events. Afterward, the participant combines off-chain the public view

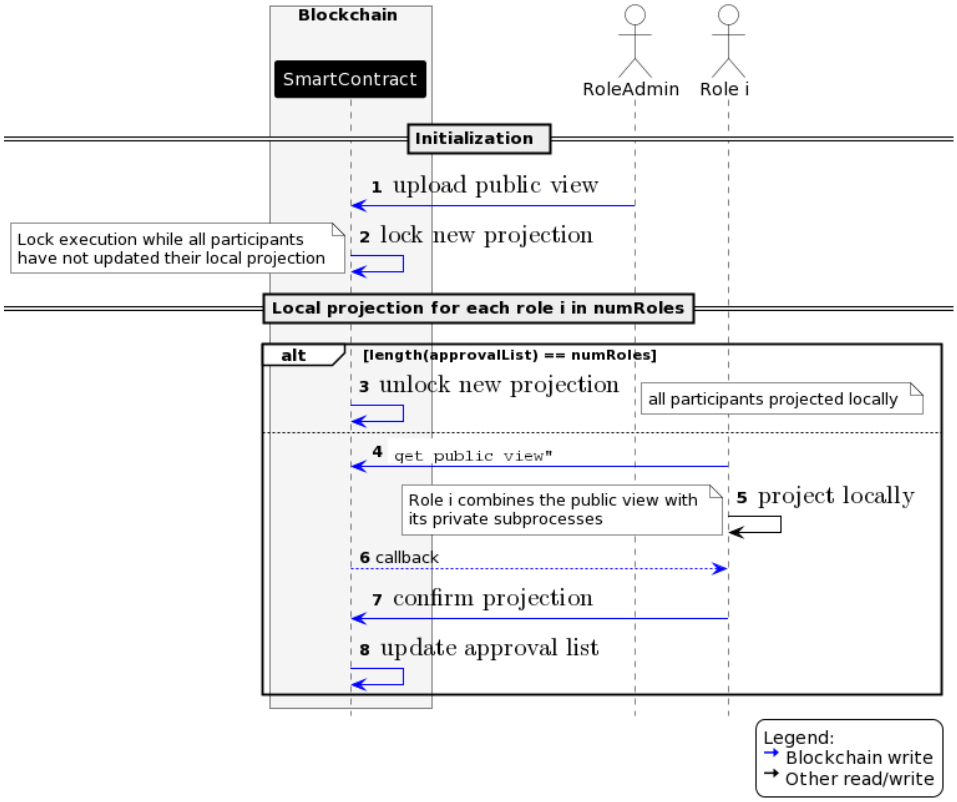
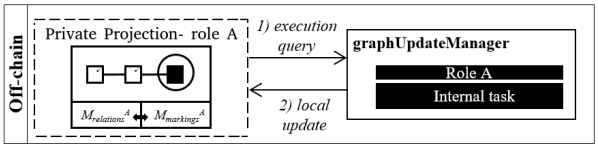
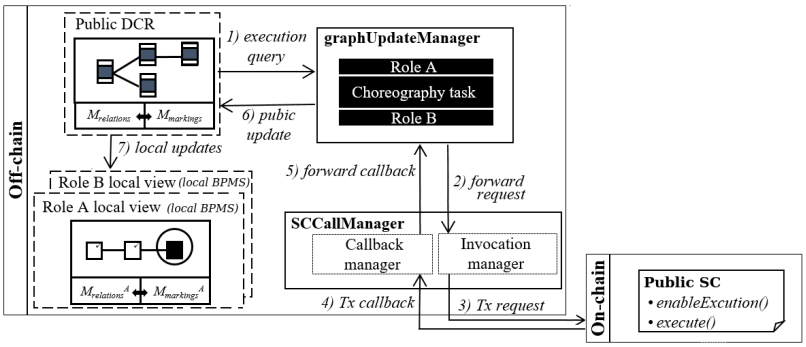


Figure 4.2: Sequence diagram of the hybrid on/off-chain design protocol



(a) Execution of an internal event



(b) Execution of a choreography event

Figure 4.3: The execution scheme logic of DCR choreography events

with its internal events (Figure 4.2, step4). The obtained projection over the role r is G_r. A dedicated smart contract function named, *confirmProjection()*, enables participants to update σ after the local projection. The function uses two mapping variables. The first mapping, $\gamma_{approval}$, records whether a participant has generated its local projection. The second mapping, γ_{fetch}, records whether the participant did fetch the public view (necessary condition to realize the projection). The following constraints restrain σ update: (i) the sender's address must belong to the list of addresses white-listed in the smart contract, (ii) participants can only update the variable once, and (iii) must have fetched the public projection first. In our motivating example, Florist asks for the public projection of the smart contract. The smart contract verifies that its address belongs to the white list, forwards the public view to Florist, and updates σ to 1. Florist projects the view over her role. She obtains a set of receive events: {*Shipping, CheckOrder, Accept, Reject, Pay, UnloadTruck*}, and one send event {*PayDriver*}.

She then adds its internal activities {*GetOrder, CallDriver, SettleOrder*} to the projection. Lastly, Florist triggers $confirmProjection()$.

4.5 Hybrid Off/On-chain Runtime Execution

Our approach proposes a hybrid execution at runtime comprising two strategies. On the one hand, the private DCR execution engine of the involved participants manages the private projections. This strategy is

described in Section 4.5.1. On the other hand, a smart contract called V_{DCR} triggers the execution logic of the public tasks on the blockchain. This strategy is described in Section 4.5.2.

An event execution query comprises the name of the event and its class: *internal*, or *choreography*. The execution logic depends on the event class.

4.5.1 Managing internal execution requests off-chain

Participant executes private events off-chain (cf. Figure 4.3a). For an internal event, the private process engine looks at its private markings (see Figure 4.3a). If the event is enabled[3], we apply post-execution constraints to the bound events (i.e., events are set to pending, included, or excluded), and update the marking accordingly.

In our motivating example, the execution request of *GetOrder* (Figure 4.1a) will succeed: it does not have any pre-execution constraint. Thus, the executed marking of the event *GetOrder* will be set to one. The post-execution constraints (*condition* and *response*) will unlock *CallDriver* and set its pending marking to one.

4.5.2 Managing choreography events execution requests on-chain

Private and public projections communicate via choreography events (via send and receive events). The smart contract V_{DCR} handles the execution of the choreography send and receive events (cf. Figure 4.3b). V_{DCR} holds the bitvector representation of the public view and two functions: *enableExecution()* checks the enabling preconditions, and *execute()* computes the enabled event and updates the marking vectors.

The execution of a choreography event follows the subsequent steps (see Figure 4.3b). First, the backend receives an execution query (step 1) and forwards it to the smart contract API (step 2). The latter sends a transaction to V_{DCR} to call the function *enableExecution()* (step 3). The transaction includes the event's name to execute, the event initiator, the receiver (if it is a choreography event), and the event state (enabled, included, executed). If the activation conditions are verified, the function *execute()* updates the event state (the three bitvectors) and the public projection state (the five relation matrices). The transaction callback containing the updated states is sent back to the smart contract API (step 4), which forwards it to the local backend (step 5). The backend updates the public projection (step 6). Changes are propagated to the concerned private projections (step 7).

In our motivating example, we suppose that Florist executed the private tasks *GetOrder* and *CallDriver*. Driver now launches the execution request of

[3]An event is *enabled* if the following preconditions are fulfilled: the event is included, and the *condition* and *milestone* relations are executed

the public event *Shipping* to the smart contract (Figure 4.1a). The function enableExecution() evaluates to true as the event is set to pending. The smart contract function execute() then updates the onchain marking of *Shipping*, and updates the marking of *CheckOrder* to pending. A notification event is emitted to the backends of the participants Florist, Customer and Driver, and the markings of their local projections are updated accordingly.

Choreography events are by nature of interest to process participants. V_{DCR} makes their execution management trustworthy as its behavior is deterministic, and the choreography states stored in the smart contract are tamper-proof.

In the following sections, we will refer to Driver as Carrier A_1.

4.6 Implementation and Evaluation

In this section, we present a prototype implementing the aforementioned protocol in the Ethereum blockchain (Section 4.6.1). We then evaluate its latency and scalability by running a set of experiments on our motivating example business process model, as well as two business process models presented in the literature (Section 4.6.2).

4.6.1 Implementation

Our proof of concept is a hybrid on/off-chain business process engine managing declarative choreographies [4]. We use a Ganache testnet to deploy the public smart contract V_{DCR}, which manages each process. V_{DCR} comprises (1) execution constraints rules and (2) a list of workflows initially empty.

We use the August 24th, 2022 conversion rate (1ETH=1,663.76€) in the following.

The initial cost of deployment of V_{DCR} is 0.06413472 ETH (i.e., 106.7€). Additionally, a smart contract manages roles authentication and access control rules. Its deployment cost is 0.05859442 ETH (97.5€) for a gas usage of 1,953,149.

For each workflow, RoleAdmin (1) generates the public view bit vector representation of the DCR choreography (Section 4.4), (2) saves the textual public view input to IPFS, and (3) registers the new workflow on-chain by calling the function uploadPublicView. The workflow is identified by the IPFS unique hash. Participants interact with the smart contract via API calls to generate their private views. Afterward, the process instance is

[4]Code repository: `https://archive.softwareheritage.org/swh:1:dir:211c6bdb1ce9f256c363caee54a56f53ada05d9b;origin=https://github.com/tiphainehenry/hybridChoreo;visit=swh:1:snp:6376000436aeaf872f39fb5f1a9d5aaa417c5cea;anchor=swh:1:rev:a5dd01e0ed0cb034c1e35daa0e6b3c6e08ff6d97`

released for execution. The local process execution engine executes internal events off-chain and forwards choreography events to the blockchain.

4.6.2 Evaluation

We evaluate the protocol using experiments that aim respectively to measure (1) the smart contract transaction costs for the deployment of a cross-organizational process, (2) the transaction processing time and gas fees, and (3) a comparison between BPMN and DCR choreography executions onchain. These metrics are important because they provide insights regarding the latency and scalability of the solutions based on different use cases.

To do so, we instantiate three cross-organizational processes in the platform: we test two workflows from the literature: the invoice and oncology workflows [190], and the motivating example. We run the experiments on a personal computer with an Intel i5 core CPU and 4GB of RAM.

We evaluate the public-to-private projection costs when deploying the three processes mentioned above (cf. Table 4.2a). For each workflow, the public view registration cost is worth 0.068352 ETH (113.7€) for the delivery workflow, 0.040947 ETH (68.1€) for the invoice workflow, and 0.065019 ETH (108.2€) for the oncology workflow. Afterward, each role fetches the public view and confirms its projection. The delivery and invoice workflows share the same costs for fetching the public view and confirming the projection. Such cost, corresponding to updating $\gamma_{approval}$ and γ_{fetch}, is proportional to the number of roles registered. The total cost for instantiating a choreography corresponds to the public view upload and the number of roles $\#R$ times the private projection cost. It is worth 0.078534 ETH (130.7€), 0.051129 ETH (85.1€), and 0.079795 ETH (132.8€) for the delivery, invoice, and oncology workflows, respectively. Hence, the public-to-private total projection cost depends on the number of roles and events. Indeed, the more activities, the highest the public view upload will be. Similarly, the more participants, the highest the total cost of private projections will be, though the total projection cost is divided equally between participants. Nonetheless, the transaction cost behavior of the public view may underline scalability limitations for more complex public processes.

We also evaluate the system's performance at runtime. Table 4.2b presents the results obtained after the enactment of one trace. The reported execution time factors in the transaction confirmation time. The average transaction fees requested for a task execution are smaller than the process instantiation ones. Moreover, the average execution time for a private task is smaller than that needed for a public task. Indeed, we compute private activities off-chain. Thus the execution time of a private event corresponds comprises checking the event's nature (private or public) and updating private markings. Conversely, the execution of public activities comprises an interaction with the blockchain network. The latency induced by executing public tasks

onchain may hinder the adoption of this solution for time-intensive processes. Nonetheless, these experiments underline the benefits of executing private tasks locally to reduce the overall execution time.

Finally, we compare the transaction costs of our approach to the BPMN-based experiments presented in [122]. We translate into DCR choreographies the two open-sourced BPMN choreographies presented in [214], namely *supply chain* and *incident management*. We deploy and execute the choreography in our prototype and compare the results. Table 4.3 shows the instantiation and task execution average gas fees; task execution fees correspond to the average cost of execution of a task. A gain of 26,412 gas for the supply chain workflow and 189,404 gas for the incident management workflow can be noticed with the DCR approach. Thus, the DCR-based smart contract requires fewer fees for instantiation than the BPMN one in these workflows. Regarding task execution costs, the modeling choice does not seem to impact gas fees: a gain can be noticed with DCR in the supply chain workflow but not in the incident management one. The number of gateways (2 in the supply chain and six in the incident management workflow) may explain this disparity. Indeed, each exclusive gateway is translated into an include and a response relation for each decision path in the DCR model. Such translation may explain the gas difference.

In summary, the public-to-private total projection cost depends on the number of roles and events, which may be a limiting factor for more complex public processes. The average execution time for a private task is smaller than that needed for a public task which requires a smart contract interaction. Hence these experiments underline the benefits of executing private tasks locally to reduce the overall execution time. Finally, the DCR-based smart contract requires fewer gas fees for instantiation than the BPMN one in these workflows, though the modeling choice does not seem to impact gas fees; experiments on graphs of alternative complexity should confirm these preliminary results. These experiments give proof to the first contribution i(a) (an on/off-chain declarative choreography deployment and execution system), and validate objectives 1 and 2.

4.7 Conclusion

This chapter presents the contribution i(a) presented in Section 1.4.2. We address the research question RQ1.1 on the need for a trustworthy separation of concerns (c.f., Section 1.3) by leveraging the management of business process choreographies using blockchain. We model choreographies with a declarative language called DCR. This language offers loosely-constrained models to meet the flexibility requirements of cross-organizational processes (c.f., Section 1.3, RQ2.2). The public view of the choreography is stored in a smart contract, and participants generate their private view off-chain to

enhance privacy at design time. On the execution side, internal events are executed locally for privacy concerns, while choreography events are executed on-chain for accountability concerns (c.f., Section 1.3, RQ3.1).

This approach represents a first effort to separate a declarative choreography's public and private views and proceed with its hybrid off/on-chain management. Results confirm the advantages of separating public from private events to ensure privacy while leveraging blockchain as a decentralized execution infrastructure. Moreover, the local execution of private events leads to time and economic gains. Our approach works if there is no public event. Then, no public projection is generated. Multi-instance choreographies are also possible: then, DCR graph instances are added chronologically to the smart contract.

Through the presented experiments, we validate Objective 1 presented in Section 1.4.1. These experiments show the feasibility of the solution. The DCR-based smart contract requires fewer fees for instantiation than BPMN workflows, and the execution in an off-chain setting of private activities reduces the overall process execution latency. Nonetheless, the total deployment cost is proportional to DCR graphs complexity (i.e., the number of activities) and the number of participants. This behavior calls for further studies to scale the solution to more complex graphs.

The work presented in this chapter has been published in the International Conference on Service-Oriented Computing [87].

Table 4.2: Hybrid on/off-chain Projection and execution costs

(a) Public-to-private projection costs, *W.* =*Workflow*

Step	**Role**	**Function**	**Delivery W.**	**Invoice W.**	**Oncology W.**
A	RoleAdmin	*uploadPublicView()*	0.068352 ETH	0.040947 ETH	0.065019 ETH
B1	Role r in R	*fetchPublicView()*	0.002006 ETH	0.002006 ETH	0.002139 ETH
B2	Role r in R	*confirmProjection()*	0.001388 ETH	0.001388 ETH	0.001555 ETH
Total Cost = A + $\#R$.(B1+B2)			0.078534 ETH	0.051129 ETH	0.079795 ETH

(b) Task execution costs *(Pub/Pri= public/private tasks).*

Workflow					**Tx. Fees**	**Exec Time**	
Name	*#Parties*	*#Pub*	*#Pri*	*#Constraints*	*Task Exec*	*Pub*	*Pri*
Delivery	3	9	1	28	0.0093 ETH	15s	1s
Invoice	3	8	2	15	0.0069 ETH	10s	1s
Oncology	4	10	3	21	0.0117 ETH	19s	2s
				Mean	0.0093 ETH	14.6s	1.3s

Table 4.3: Gas fees comparison of BPMN [122], and DCR choreographies (our approach).

Workflow	**#Tasks**	**#Gateways**	**Gas fees**	**[122] (BPMN)**	**Our approach**
Supply Chain[214]	10	2	Instantiation	1,100,590	1,074,178
			Task exec.	566,861	478,527
Incident Mgt.[214]	9	6	Instantiation	1,119,803	930,399
			Task exec.	324,420	456,887

Chapter 5

Control-flow and Partnership Flexibility

Contents

5.1 Introduction

As noted in Chapter 3, imperative and declarative process modeling paradigms have been used to execute blockchain-based business processes. Proposed techniques include translation of BPMN collaboration models into smart contracts [214] and execution engines of declarative orchestration processes called Dynamic-Condition-Response (DCR) graphs [127]. However,

dealing with changes in blockchain-enabled business processes remains an open research issue [125]. Business processes managed by "static" smart contracts cannot be upgraded because the smart contracts are immutable once deployed. Efforts exist to support basic versioning in smart contracts or a registry for smart contracts [222], or to manage governance changes using soft-forks [52]. Nevertheless, managing changes in business processes, particularly in running instances, needs more fined-grained and advanced change management techniques.

Control flow changes can be necessary in a running choreography instance [66, 206]. A change may consist of a simple change operation (ADD/REMOVE/UPDATE) or a combination of change operations [141, 66]. A change in a partner process instance may affect other partners' process instances. Hence, change must be propagated to the affected partners of the choreography instance [66, 182]. One has also to ensure that neither the structural nor behavioral compatibility of partners processes are violated after a change [2, 66, 48, 50]. Structural compatibility checks consist of ensuring that there is at least one potential *send* message assigned to a partner with a corresponding *receive* message assigned to another partner[50]. Behavioral compatibility refers to ensuring that the choreography process after the change is safe and terminates in an acceptable state. In other words, no deadlocks should occur between partners' public processes during the choreography execution after change [66, 48].

Flexibility regarding partners' allocation is also necessary at runtime in complement to control-flow changes, e.g., for a dynamic task assignment to participants at runtime [8]. In the context of task binding, contractors need to be sure that a newly hired service provider will meet quality standards [159]. Several resource-binding mechanisms have been proposed [98, 204] that are often organized in a centralized way. Due to power asymmetries, this centralized architecture may induce distrust. Thus, the need for a trustworthy and objective resource-binding mechanism arises.

To our knowledge, the integration of change management, especially change propagation, in blockchain-based declarative choreographies management systems has not been studied. Similarly, the transfer of Quality of Service (QoS)-mechanisms to resource-binding smart contracts has been under-investigated. To do so, we focus in this chapter on the following research questions:

1. *How to change the process model instances of blockchain-based cross-organizational processes?* (c.f., Section 1.3, RQ2.3). In light of the related work, we refine this question as follows: *How to guarantee correct change propagation in declarative blockchain-based choreography processes such as DCR graphs?*

2. *How to foster actors' (re)allocation flexibility on running instances*

using a smart contract-based multi-criteria decision algorithm that leverages blockchain-based process history? (c.f., Section 1.3, RQ2.1)

3. *How can smart contracts foster trustworthy and power-balanced decision-making for the runtime allocation of resources?* (c.f., Section 1.3, RQ1.2)

To answer these research questions, we first present basic concepts and our motivating example in Section 5.2. We then propose the following contributions.

First, we provide in Section 5.3 novel change management techniques in blockchain-enabled declarative choreographies. To do so, we build on and extend the solution presented in chapter 4 with the change management mechanism. DCR graphs are specified as a set of rules interpreted at runtime. They directly represent business requirements, and thus it is easier to add or update constraints if the business requirements change [49]. Our approach allows a partner in a running DCR choreography instance to change its private process, which may impact its interactions with other partners. Changes are mainly ADD/ REMOVE and UPDATE operations applied to the DCR choreography events and relations [141, 66]. We only focus on these change operations as they are challenging by themselves, and any change to a process can be written as a combination of these operations [82]. The smart contract records the involvement of concerned partners in a tamper-proof fashion, and smart contract transactions act like "approval checkpoints" during change negotiation and propagation. Hence, if misbehavior occurs, e.g., a partner projects wrongfully the change, claim resolution is eased between partners as the blockchain stores the negotiation and propagation history on-chain.

Second, we present in Section 5.4 an algorithm leveraging smart contracts to filter and sort a set of available candidates based on their past blockchain history. Such matching, done autonomously, generates a digital agreement that can be linked to a BPMS to manage the settlement of the allocated services. We thus provide a transparent and reliable protocol computing a set of tamper-proof QoS ratings. In so doing, we contribute to the literature by addressing the lack of research regarding the fair (i.e., trustworthy and reliable) enactment of QoS allocations. This work addresses the trust challenge through smart contracts enforcing an agreed-upon binding protocol and through the blockchain ledger offering a tamper-proof history of services. It also addresses the automation challenge through the management of smart contract hiring, contractualization, and settlement stages. Hence, this system decreases the end-users/platform interaction. Blockchain addresses service providers' and issuers' collusion risks by design as no single entity controls the binding platform. Such a system also preserves the integrity and transparency of the QoS protocol: it prevents tampering with the service history while ensuring a deterministic binding protocol.

Finally, we evaluate both contributions in Section 5.5.

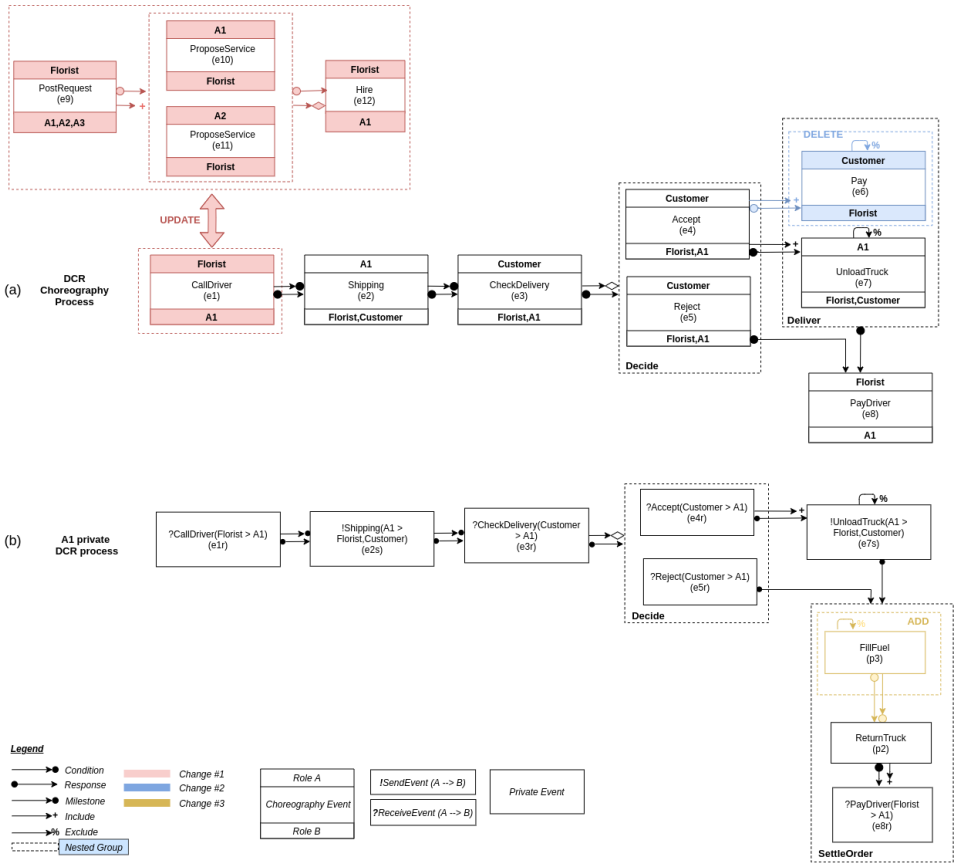


Figure 5.1: DCR choreography process and Carrier A_1 private DCR process of the flower delivery process

5.2 Basic Concepts and Motivating Example

5.2.1 Control-flow change

Both choreography and private DCR processes in our flower delivery motivating example are susceptible to control-flow changes, where a control-flow change comprises a set of change elements and a combination of change operations.

Figure 5.1 illustrates three possible change operations on the flower delivery motivating example, namely the DELETE, ADD, and UPDATE operations. Florist decides to launch a procurement auction instead of relying on a single carrier for quality concerns. She *updates* the task $e1(CallDriver, Florist \longrightarrow A_1)$ with a new business process model subprocess describing the auction (change #1). Additionally, Florist may have passed a monthly contract with Customer for flower deliveries, and

thus Customer does not need to pay for flowers each time a new delivery is carried. Hence, Customer *deletes* the task $e6(Pay(Customer \longrightarrow Florist))$ (change #2). Finally, A_1 may wish to *add* a new private activity *FillFuel* before returning the truck to the initial loading location (change #3).

We now define more formally the concepts of a change element (Definition 5.2.1) and a change operation (Definition 5.2.2).

Definition 5.2.1 (**Change Element**)**.** Let Φ=(G, I, R) be a DCR choreography. G_{Ref} is a change element (also called refinement element) if and only if one of the following conditions is met:

1. $G_{Ref} \in \{\epsilon \cup \gamma \cup \longrightarrow \bullet \cup \bullet \longrightarrow \cup \longrightarrow \diamond \cup \longrightarrow + \cup \longrightarrow \%\}$
2. $G_{Ref} = (e \in \{\epsilon \cup \gamma\}, m_e(in, pe, ex), \{\longrightarrow \bullet \cup \bullet \longrightarrow \cup \longrightarrow \diamond \cup \longrightarrow + \cup \longrightarrow \%\})$

This means that, G_{Ref} is a refinement element if it is either (1) an atomic element, i.e., an internal event such as $p3$ in Figure 5.1(b), or an interaction such as $e6$, or one of the five relations such as the condition relation linking $e6$ and $e4$; (2) a DCR fragment, i.e., a sub-graph with the minimal configuration: {one event, initial marking of the event, one relation} such as the subset $e9 - 12$ and their relations (one include, one milestone, and two responses) in Figure 5.1, change #1.

To define the change operations, we refer to [141] where authors propose three change operations on DCR orchestration processes. We re-adapt these operations to be used in the context of DCR choreographies and where the change element can be one of the three types defined in **Definition 5.2.1**. Change operations are of three major types[1]:

Definition 5.2.2 (**Change Operation**)**.**

- $\Phi \oplus G_{Ref}$ to *ADD* the refinement element G_{Ref} to the original DCR choreography Φ. To apply the change, one has to compose the refinement element with the original graph, i.e., one has to take the union of events, labels, relations, and markings of the composition's two parts.
- $\Phi \ominus G_{Ref}$ to *REMOVE* a change element G_{Ref} from Φ. For example, to remove an interaction, one has to remove it from the set of interactions γ, its marking from the marking (In,Pe,Ex) of the graph, as well as the incoming and outgoing constraints, coming to/ going from this interaction.
- $\Phi[G_{Ref} \mapsto G'_{Ref}]$ to *UPDATE* a change element. For the case of an event (internal or interaction): the *UPDATE* operation is used for

[1]We use the same notation of the operations defined in [141]

replacing one event with another or re-labeling it. To replace, for example, one interaction with another, one has to update the set of interactions γ with the new interaction, the marking of the graph, and the set of incoming and outgoing constraints.

For instance, in Figure 5.1, Florist decides to launch procurement auctions to allocate delivery services to carriers. The request is sent to all registered carriers (here A_1, A_2, and A_3) ($e9$). A_1 and A_2 decide to propose their offer to Florist ($e10$ and $e11$). Afterward, Florist decides on the best offer for the service request ($e12$). Consequently, the changes to make are: (i) add the partners A_2 and A_3, (ii) an UPDATE operation where the interaction $e1$ is replaced with the DCR fragment $\{e9, e10, e11, e12\}$. These changes are represented in red in Figure 5.1(a) and are called *public changes*.

Change #1 and change #2 control-flow changes affect carriers' internal processes. Thus, the UPDATE and DELETE change must be propagated to all carriers' process instances. Partners should trustfully access (1) the shared view of the change and (2) the tamper-proof history of the negotiation and propagation status to avoid any conflict or misunderstanding [99]. Additionally, the public task $e6$ ("Pay") is composed of two messages, namely the send and receive messages, respectively assigned to Customer and Florist. When Customer DELETES the send message "Pay", a structural incompatibility occurs as the corresponding receive message is still present in the Florist process. Hence, behavioral and structural compatibility should be ensured.

To proceed with such a change: (i) the change should be negotiated (agreed on or not) by the involved partners, (ii) the change proposition should be examined by all involved partners, and (iii) the negotiation outcome should be tamper-proof to avoid that someone diverges from the common understanding, and (iv) the change should be correctly propagated [93, 49].

5.2.2 Partner flexibility

After the DCR choreography update of change #1, Florist can now carry on auctions to allocate a carrier to a delivery request. This choice is motivated by the will to not rely on a single delivery company to ensure quality standards.

This auction, also known as the freight transportation procurement process (FTSP), is one of the primary activities in the logistics field. The FTSP holds three stakeholders: the shipper Florist (i.e., the service buyer) who initiates the allocation request, the carrier (i.e., the service seller) who sells its delivery services, and an intermediary, often a digital platform, responsible for the carrier-shipper allocation [113, 216]. It comprises two stages: first, the matching stage, where the set [service request/resource provider] is defined, and then the allocation, which generates a mediation contract, the CMR (i.e., CMR Convention: Convention on the Contract for the International Carriage of Goods by Road). It is worth noting that the

Table 5.1: Registered carrier profiles. P_{F_i} the i_{th} filtering criteria, and P_{O_i} the i_{th} QoS optimization criteria

	P_{F_1}	P_{F_2}	P_{F_3}	P_{O_1}	P_{0_2}
Carriers	Date	City	Equipment	Experience	Delay
A_1	D1	Geneva	Cold storage	110	10
A_2	D2	Zurich	-	90	0
A_3	D1	Geneva	Cold storage	32	20
A_4	D2	Geneva	Cold storage	32	20

platforms' service procurement process will be the same for either shipper-to-carrier or carrier-to-carrier allocations.

In our example, Florist wishes to find the carrier with the best quality of service (QoS) to ensure, e.g., a constant delivery temperature or just-in-time delivery. She asks for flower delivery from Geneva to Zurich at date D1. She wants the carrier closest to Geneva and the truck to offer cold storage. She wishes for a carrier with good experience and is not concerned about delay.

A_1, A_2, A_3, and A_4 are the four competing delivering carriers wishing to win the auction. Table 5.1 presents their profiles. Each profile comprises (1) three filtering criteria: availability date, city, and equipment, and (2) two optimization criteria: the candidate's experience (number of former deliveries) and the average delay in minutes.

One option for Florist to carry on the QoS-based allocation is to rely on an intermediary responsible for collecting carriers' profiles and keeping track of their QoS history. However, the resource binder can become a source of failure. Some carriers may want to hide past services to disguise their poor QoS record or collude with the resource binder to gain market shares. Consequently, the need for a trustworthy binding protocol arises.

5.3 Control flow flexibility

Partners coordinate their processes and propose/receive changes to/from other partners. We aim to make it possible for each partner to (i) modify its private DCR process and (ii) suggest a change to the DCR choreography monitored in the blockchain (Section 5.3.1). If the change request is fully private, for e. g., it concerns an internal event or a relation linking two internal events (private-to-private relation) or a relation linking an interaction to an internal event (public-to-private relation), then the private process of the partner updates accordingly. If the change is public, i.e., concerns an interaction or a relation linking two interactions (public-to-public relation) or a relation linking an internal event to an interaction (private-to-public relation), then a negotiation stage starts (Section 5.3.2), followed by a propagation stage (Section 5.3.3).

Table 5.2: Proposed allowed (AR) and denied (DR) change rules for a DCR process

Type	**Rule**
AR1	Change condition / response / milestone relations
DR1	Inclusion of an excluded event
DR2	Exclusion of an included event
AR2	Block temporarily/ permanently an included event

5.3.1 Step 1: Change Proposal

The role initiator defines the change of its private DCR process. She may modify internal events, interactions, and relations linking events. The introduction of a change is called refinement (cf. **Definition 5.2.1**). It is done before submitting it to other partners for examination.

A set of integrity rules need to be defined to ensure the correctness of the updated graph, i. e., the non-violation of the behavior of the original graph when one introduces a change. These rules should ensure the graph's safety (no deadlocks) and liveness (no livelocks). A DCR graph is deadlock-free if, for any reachable marking, there is either an enabled event or no included required responses. Whereas liveness describes the ability of the DCR graph to completion by continued execution of pending response events or their exclusion.

To do so, we leverage non-invasive adaptation rules, originally introduced in the context of DCR orchestrations, to DCR choreographies [49]. We divide these rules into rules describing (i) allowed change rule (AR) and (ii) denied change rule (DR) presented in Table 5.2.

One can ADD/REMOVE/UPDATE condition, response, and milestone relations (AR1). The only restriction is not to have condition/response relations cycles to avoid deadlocks. However, one cannot include an already excluded event (DR1), nor can she exclude an already included event (DR2). One alternative is to temporarily or permanently block an event (AR2). We suppose we want to permanently block a DCR graph G of executing an event e. We refine with the fragment Q: $Q = \{\ e$: (0,1,0), e': (0,1,0)), $e' \longrightarrow \bullet\ e'$, $e' \longrightarrow \bullet\ e\ \}$. Here, e can never fire (again) because it depends on e'. Moreover, by excluding and including e', one can selectively enable and disable e.

In our example, the change proposal #1 replaces $e1$ with the fragment composed of the events $\{e9, e10, e11, e12\}$. The change proposal evaluates to *true* because $\forall i$, (ARi) evaluates to *true* and $\forall j$, (DRj) evaluates to *false*.

5.3.2 Step 2: Change request and negotiation

This step is divided into two sub-steps: change request first (Step 2.1), followed by a change negotiation (Step 2.2).

Step 2.1: Change request The smart contract stores the list of change requests assigned to process instances as a hash map. Ongoing process instance changes are recorded with the identification hash of the current process instance ψ_G. The identification hash corresponds to the IPFS hash of the process instance description.During the change request life-cycle, the request is assigned to a status belonging to {Init, BeingProcessed, Approved, Declined}. Status is set to Init if no change request is ongoing, to BeingProcessed during the negotiation stage, to Approved or Declined once the change request is processed by all endorsers.

Algorithm 2 presents the smart contract function registering a change request. The identity of the change initiator is checked: it should belong to the list of partner addresses (line 2). If the workflow is not attached to a change request (line 3), then the change request is created (lines 4-8). The hash of the redesigned workflow is stored in $\psi_{G'}$ (line 4). This identification hash corresponds to the IPFS description of the requested redesigned public workflow $\psi_{G'}$. The change request status is set to "$BeingProcessed$" (line 5). The addresses of the change initiator and endorsers $R_{endorsers}$ are attached to the request (lines 6-7). Endorsing partners are, for example, in the case of adding a choreography interaction i (i) the sender and the receiver(s) of the event and (ii) partners connected directly with the choreography interaction. The change initiator also sets two response deadlines, t1 for change endorsement and t2 for change propagation, to be checked by the smart contract (lines 8-9). Finally, the smart contract emits a change request notification to all partners listening to the smart contract (line 10). If one of the change endorsers does not reply before deadline t1 during endorsement or t2 during propagation, an alarm clock triggers a smart contract function canceling the change request. If one of the change endorsers does not reply before deadline t1 during endorsement or t2 during propagation, an alarm clock triggers a smart contract function canceling the change request at a specified block in the future corresponding to t1 or t2. It consists of a smart contract function being called by incentivized users triggering the smart contract at the desired timestamp [116]. Upon trigger, the smart contract function sets the change request status to canceled and emits an event notifying partners that the change has been canceled. By so doing, we prevent any deadlock due to one of the partners not responding.

In Figure 5.1, Change #1 is public as it concerns three partners, namely Florist, A_1, A_2, and A_3. Hence a negotiation must occur between the partners to reach a consensus on the proposed change before propagating it. Florist launches the change negotiation by triggering the smart contract. The smart contract updates the change requests list linked to ψ_G with the following information: [(1) $\psi_{G'}$ the IPFS hash of the updated process description which comprises the operation UPDATE($e1$) with ($e9$+$e10$+$e11$+$e12$), (2) the list of endorsers: $\{a_{A_1}, a_{A_2}, a_{A_3}\}$, (3) Change negotiation deadline t1 = 72h, (4) Change propagation deadline t2 = 120h]

Algorithm 2: Request change smart contract function

Data: $\gamma_{requests}$ the list of change requests, $A_{endorsers}$ the list of endorser addresses, ψ_G the current IPFS workflow hash, $\psi_{G'}$ the IPFS hash of requested change description, $t1$ the deadline timestamp for change endorsement, and $t2$ the deadline timestamp for change propagation

Result: emits change request notifications to endorsers

1 **Function** `requestChange`(ψ_G, $\psi_{G'}$, $A_{endorsers}$, $t1$, $t2$)**:**
2 require a_{sender} belongs to the list of business partners;
3 **if** $\gamma_{requests}[\psi_G].status == Init$ **then**
4 set $\gamma_{requests}[\psi_G].\psi_{G'} \leftarrow \psi_{G'}$;
5 set $\gamma_{requests}[\psi_G].status \leftarrow$ "$BeingProcessed$";
6 set $\gamma_{requests}[\psi_G].initiator \leftarrow a_{sender}$;
7 set $\gamma_{requests}[\psi_G].endorsers \leftarrow A_{endorsers}$;
8 set $\gamma_{requests}[\psi_G].t1 \leftarrow t1$;
9 set $\gamma_{requests}[\psi_G].t2 \leftarrow t2$;
10 emit RequestChange(ψ_G, $\psi_{G'}$, $A_{endorsers}$, a_{sender});
11 **else**
12 emit Error `// an ongoing change request is being processed`
13 **End Function**

Step 2.2: Change negotiation All partners subscribe to the change request events emitted by the smart contract. Endorsing partners must send their decision requests to the smart contract based on the rules in Table. 5.2. If the change, once computed on the endorser's process, respects all ARi and DRj rules, then the endorser approves the request. It is otherwise rejected. The smart contract collects the decisions from the endorsers to lock (or not) the choreography instance and proceed (or not) with the change. We detail both stages hereinafter.

Algorithm 3 presents the smart contract function receiving one endorser's decision. The endorser address $a_{endorser}$ should belong to the list of registered addresses (line 2) and not have answered the change request already (line 3). The change request should also be processable, i.e., its status should be set to "$BeingProcessed$" (line 4). The endorser response δ is processed if all conditions are met. If δ equals 1 (line 5), the endorser has accepted the change. Its response is saved into the change endorsement list (line 6), the notification of acceptance is sent to all endorsers as well as the change initiator (line 7), and the smart contract checks whether the instance needs to be locked (line 8). The $lockInstanceChecker$ function assesses whether all endorsers have accepted the change: the change endorsement list $\gamma_{endorsement}$ should be filled with ones. At this stage, no further execution of included events is allowed, and the mechanism waits for pending events to terminate.

Algorithm 3: Endorser decision management smart contract function

Data: $\gamma_{requests}$ the list of change requests, $a_{endorser}$ the endorser address, $A_{endorsers}$ the list of endorsers, ψ_G the hash of the current workflow, $\psi_{G'}$ the hash of the desired workflow, δ the endorser response $\in \{0, 1\}$

1 **Function** `endorserRSP`(ψ_G, $a_{endorser}$, δ)**:**
2 require($a_{endorser} \in E$);
3 require($\gamma_{requests}[\psi_G].\gamma_{endorsement}[a_{endorser}]$!= 1);
4 require($\gamma_{requests}[\psi_G].status ==$ "$BeingProcessed$");
5 **if** $\delta == 1$ **then**
6 set $\gamma_{requests}[\psi_G].\gamma_{endorsement}[a_{endorser}] \longleftarrow 1$;
7 emit AcceptChange($\psi_{G'}$, $a_{endorser}$);
8 lockInstanceChecker(ψ_G)
9 **else if** $\delta == 0$ **then**
`// declineapprovalOutcomes`
10 set $\gamma_{requests}[\psi_G].status \leftarrow$ "$Declined$";
11 emit DeclineChange($\psi_{G'}$, $a_{endorser}$);
12 **else**
13 emit Error($\psi_{G'}$, $a_{endorser}$);
14 **End Function**

The status of the change is then updated to "*Approved*".

In our example, we suppose that endorsers confirmed the change request ($\delta_{A_1} = 1$, $\delta_{A_2} = 1$, and $\delta_{A_3} = 1$) while respecting t1. The smart contract locks the instance for change propagation. As it manages the negotiation process, a tamper-proof record of the negotiation is accessible to all partners. This prevents conflicts and eases potential claim resolutions.

5.3.3 Step 3: Change propagation

Change propagation is to apply the change effect after the negotiation phase succeeds to (i) the affected partners' DCR public processes, and (ii) each partner propagates the change effect to its private DCR process. To ensure the correctness of the change propagation, we introduce the following property (*Property 1*) where ν is a change, r is the initiator of change ν, and $R_{endorsers}$ are the set of endorsers, G_r,$G_{r'}$ are respectively the public DCR process of r and r' where $r' \in R_{endorsers}$:

Property 1. *if* $effect(c)_{\|G_r}: \Rightarrow G_r$ is correct-by-construction and $\forall r \in R_{endorsers}$, $effect(c)_{\|G_{r'}}: \Rightarrow G_{r'}$ is correct-by-construction *then* G_r and $G_{r'}$ are compatible $\forall r \in R_{endorsers}$.

Property 1. states that if G_r is correct-by-construction and if $\forall r' \in R_{endorsers}$, $G_{r'}$ are also correct-by-construction, then compatibility is verified.

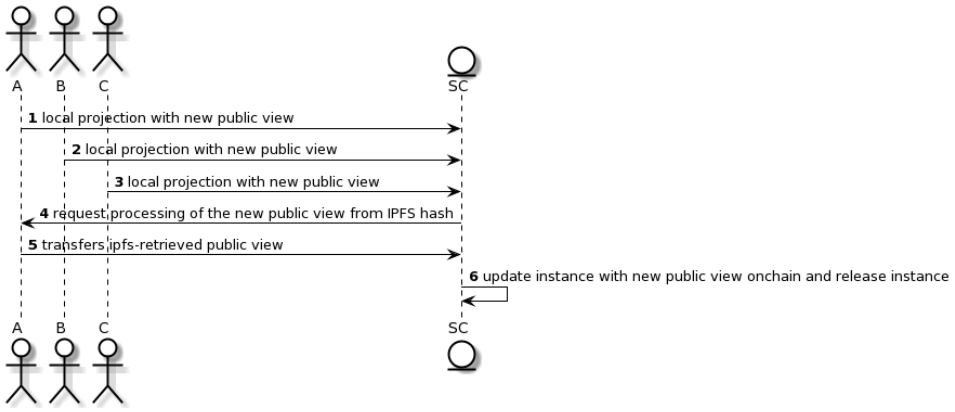


Figure 5.2: Sequence diagram of the propagation stage illustrating the interactions between partners and the smart contract

In fact, a public DCR process G_r is correct-by-construction means that computing the effect of a change ν over r introduces no deadlocks in G_r (see Sect. 5.3.1). Thus, if the DCR public models of the change initiator and endorsers are safe, i.e., no deadlocks can occur, they can communicate adequately after the change and are consequently compatible with each other. The smart contract enforces propagation correctness given that it maintains the tamper-proof record for the endorsement and application of the change effect across the partners. This is described in the following.

Figure 5.2 depicts the sequence diagram of the change propagation interactions between partners and the smart contract. Each partner projects locally the DCR choreography in its projection using the process description given in the IPFS hash (Figure 5.2 step 1-3). Algorithm 4 presents the function triggered by partners to confirm the projection to the smart contract. A list γ_{propag} keeps track of the propagation status, i.e., it records the private projection of each partner. The function checks that the partner belongs to the list of endorsers (line 2) and that the endorser has not projected locally yet (line 3). The smart contract detects the completion of all local projections once γ_{propag} is filled with ones and notifies the change initiator. The change initiator then retrieves the new DCR choreography that was saved into IPFS using $\psi_{G'}$ (Figure 5.2 step 4) and forwards it to the smart contract (Figure 5.2 step 5). The smart contract updates the relations and markings stored in the process instance and resets the status of the change of the workflow instance: a new change request can be processed (Figure 5.2 step 6).

In our motivating example, the propagation of change #1 occurs with partners Florist, A_1, A_2, and A_3 updating their private DCR process with the approved change. They first retrieve the change description stored in IPFS under $\psi_{G'}$. They then project the updated public change description

Algorithm 4: Confirm change propagation smart contract function

Data: $\gamma_{requests}$ the list of change requests, $a_{endorser}$ the sender address, $A_{endorsers}$ the list of endorsers, ψ_G the hash of the current workflow

Result: manages the record of projections of the new public view

1 **Function** `confirmProjection`(ψ_G, $a_{endorser}$)**:**
2 require($a_{endorser} \in E$);
3 require($\gamma_{requests}[\psi_G].\gamma_{propag}$[id] != 1);
4 set $\gamma_{requests}[\psi_G].\gamma_{propag}$[id] $\longleftarrow$ 1;
5 emit LogWorkflowProjection(ψ_G);
6 **End Function**

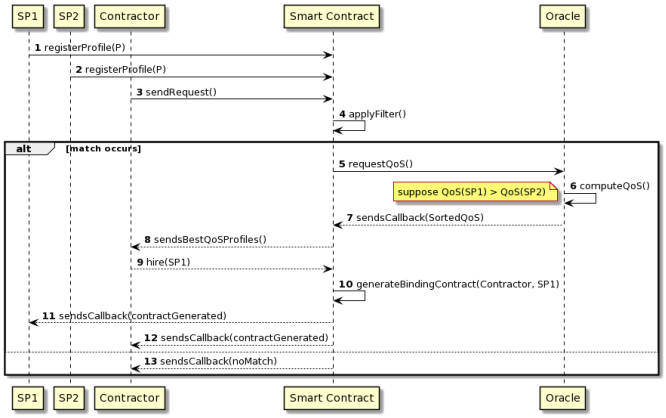


Figure 5.3: Sequence Diagram of the binding solution (SP= Service Provider)

on their role following the same approach as in Section 4. For example, A_1 will retrieve the activities $\{e9, e10, e12\}$. A_1 then combines this projection with private activities $\{p2,p3\}$. Once all projections have been done and notified to the smart contract, the change initiator *Florist* triggers the smart contract. The trigger updates the process description of the running instance, e.g., the updated relation matrices, activity markings, and access controls (c.f. Section 4 for a more detailed explanation).

5.4 Actor flexibility

Alongside control-flow change, the allocation of partners to tasks should also be flexible. We present a trustworthy mechanism building on QoS-based allocation to address this requirement.

Figure 5.3 presents the matching stage sequence diagram. Suppose two service providers subscribe to the smart contract (e.g., delivery carriers) (step 1-2). First, the contractor sends the service request to the resource-

binding smart contract (step 3). The smart contract will scan resources to find matching candidates based on the filtering criteria (step 4). If any candidate meets the required filtering criteria, the smart contract calls the oracle that triggers an external API (step 5). The oracle computes the QoS rate of the successful candidates (step 6). Upon completing the request, the oracle sends a callback to the smart contract with the sorted profiles (step 7). The smart contract will, in turn, send the best profiles to the contractor (step 8). The contractor answers the smart contract with the profile she likes best (step 9). The smart contract generates a binding agreement on-chain to confirm the resource-binding and manage the service fulfillment later (step 10). It also triggers a success callback directed towards the contractor (step 11) and the elected service provider (step 12). If no matching occurs, a no-match event is sent instead (step 13). It is to note that a fully autonomous binding is also possible. In this case, the smart contract will initiate the matching and generate the binding without asking the contractor to choose a candidate among the retrieved profiles.

We present in more detail each stage of the solution hereinafter. We introduce the instantiation of the platform in Section 5.4.1. We then present the smart contract-based candidates' filtering and sorting strategy in Section 5.4.2. We finally present the service binding and fulfillment mechanism in Section 5.4.3.

5.4.1 Platform instantiation

At design time, a set of representative participants gathers to instantiate the binding platform (e.g., delivery carriers and florist unions). They specify the set of possible binding parameters accessible by the smart contract. Each resource profile, managed by the resource-binding smart contract will be constituted with this set of parameters. These parameters are of two kinds: filtering criteria and optimization criteria. The representative participants also specify a factory binding agreement linked to the smart contract. This factory agreement, once instantiated, will be used by the contractor and service provider to manage the service fulfillment. We suppose a set of initially registered resources. A matrix stores these profiles. Each line corresponds to a candidate profile, and each profile comprises: (i) the resource availability, (ii) the set of filtering parameters agreed upon at the platform design time, and (iii) the set of optimization criteria.

Competing carriers A_1, A_2, A_3, and A_4 are registered in the resource binding smart contract functionality of the business process management platform. The smart contract stores the filtering metrics P_F for each candidate and optimization metrics P_O presented in Table 5.1. Our florist sends a request to the resource-binding smart contract. She asks for flower delivery from Geneva to Zurich at date D1. She wants the carrier closest to Geneva and the truck to offer cold storage. She wishes for a carrier with

good experience and is not concerned about delay. Thus she sets the weight for experience to 1 and the weight for the average delay to 0.6. Florist asks for the resource-binding smart contract to retrieve the delivery carrier with the best weighted QoS profile.

5.4.2 Filtering and sorting candidates

In the following, let S be the service request, U be the candidates' matrix, P_F the list of filtering criteria, P_0 the list of optimization criteria, and W the optimization weights. Each weight $\in$ [0,1]. The smart contract U matrix stores P_F and P_0 for each profile. A binding links S to a service provider.

The matching stage aims at retrieving the resource with the best QoS profile compared to a service request: it filters and sorts resources stored in U. The contractor sends the binary filter X of the required filtering criteria to the smart contract. If the i_{th} criterion is needed by the contractor, then $X[i] = 1$. Algorithm 5 presents the filtering function of the resource-binding smart contract. An internal binary filter assesses a candidate's eligibility. Each index holds for a candidate. The value assigned to the index of a matching resource is 1, 0 otherwise. Initially, all indexes equal 0 (line 2). A boolean variable ω assesses candidates' matching status. We suppose there is no initial matching by setting this variable to False (line 3). We then assess each candidate's profile based on the required attributes X (e.g., a truck with cold storage) (lines 6-12). If at least one candidate fits, we retrieve its availability (line 13). If the candidate is available, the ω variable equals True, and the internal filter is set to 1 (lines 14-16). We then return the filter (line 19).

Afterward, the smart contract must sort filtered candidates according to their QoS. If only one resource matches, the profile is returned to the contractor. If more than one resource matches, a multi-objective sorting is launched to retrieve the best QoS profile. We use the optimization criteria P_0 to compute each matching resource's QoS. Each P_O is retrieved from the resource profile stored in U and is thus tamper-proof. The QoS η is calculated as a weighted normalized mean, using both P_O and the set of weighing parameters W:

$$\forall i \in U,\ \eta_i = \sum_{j=0}^{n} W_j \left\| P_{O_j} \right\|. \tag{5.1}$$

If a weighting factor equals zero, then P_O is not considered for the QoS computation. If it equals one, then P_O will be fully considered for the QoS computation.

An oracle realizes all calculations to reduce the smart contract's computation loads. The resource-binding smart contract needs funds to pay for the oracle fees. The oracle sends the API request if the resource-binding smart contract is funded. During the oracle process, several oracle

Algorithm 5: Filtering algorithm

```
Data: U, P_F, P_{F_availability}
Result: Filter[]
Function filter(U, P_F, P_{F_availability}):
    Filter = Zeros(U.length);
    bool ω=false;
    ind=0;
    forall i in U do
        .
    length bool matchAttributes=true;
    forall a in P_F do
        if (a == 1) and (U[i][a] ! = a) then
            matchAttributes=false;
            break;
    ω = checkAvailability(P_{F_availability},U[i]);
    if ω then
        filter[ind] = 1;
    ind++;
    return Filter
End Function
```

machines trigger the API. If the result computed by each oracle is correct, the smart contract receives the API output in a dedicated callback function. This callback function provides the best profiles retrieved by the QoS computation API.

The multi-objective is performed using a weighted mean on normalized data. However, it remains an illustrative example. Other techniques such as Pareto optima could be considered.

In our delivery example, the resource-binding smart contract first retrieves matching candidates, i.e., the carriers available at date D1 who are located in Geneva and whose truck is equipped with cold storage: carriers A_1 and A_3 are thus retrieved by the smart contract. The internal filter is thus [1,0,1]. The QoS optimization matrix is then computed by looping around the filtered candidates. It is forwarded to the oracle that computes η_{A_1} and η_{A_3} by applying equation (1). We obtain $\eta_{A_1} = 1.01 > QoS_{A_3} = 0.69$. The resource-binding smart contract retrieves both QoS and enacts the binding between the florist and Carrier A_1.

5.4.3 Service binding and fulfillment

The smart contract generates a digital binding agreement stored in the smart contract. The identity of the contractor, and service provider, the service

execution agreement (e.g., a service execution data, and a deadline, the pay, and optional incentives), as well as the service details (e.g., the type of tasks to be realized, or comments on the contractor or service provider side), are stored in the digital agreement. The funds are locked in the smart contract to ensure the contractor's engagement and ease the payment process. The business process management system can use the binding agreement later to manage the execution process. Additionally, smart contracts could ease claim resolutions by carrying on an agreed-upon settlement plan. Upon service completion, the client provides feedback on the smart contract. If it is positive, the service provider may receive an optional bonus. Otherwise, the smart contract forwards the funds to the contractor. Additionally, the smart contract (1) updates the service status, (2) resets the resource availability, and (3) updates the service provider's QoS.

As an illustration, the binding agreement {Florist, Carrier A_1} is stored in the list of ongoing agreements of the smart contract. It records the service and merchandise details {D1, Geneva, cold storage; 1000 flowers, 10m3}. The 500 CHF pay and the 10 CHF incentive for timely deliveries are locked into the smart contract, waiting for process completion. The process status will evolve from *scheduled* to *ongoing* to *finished*. Let's suppose the delivery occurs without any delay. Pay and incentive are sent to Carrier A_1 on *finished*. The resource-binding smart contract also updates Carrier A_1's experience QoS to 111. Its delay QoS remains constant.

5.5 Implementation and evaluation

In the following section, we investigate the feasibility of these mechanisms by implementing and evaluating each solution on the blockchain. In Section 5.5.1, we implement a change management support for DCR choreographies. We evaluate the prototype by investigating gas costs required to trigger the smart contract during the change request and propagation stages. In Section 5.5.2, we build a QoS-based resource allocation mechanism applied to the logistic environment. We evaluate the prototype quantitatively through smart contract gas costs and latency experiments. We also evaluate it qualitatively with a design science research protocol.

5.5.1 Runtime DCR change

Implementation We extend the DCR choreography management platform aforementioned with change support for running DCR graph instances[2]. V_{DCR} initially comprised (1) execution constraint rules and (2) a list of

[2]The extension code is accessible at `https://archive.softwareheritage.org/swh:1:dir:8bde592496c231084627448dbd2399446f1cf2ce;origin=https://github.com/tiphainehenry/adaptiveChangeDCR;visit=swh:1:snp:89cd57a878c58593ac077da79d99a0f00cb8ac41;anchor=swh:1:rev:fac4a9a64d86fac19dcb11515b35de2de402ff1e`.

workflows initially empty. It is enriched with the list of change requests linked to the list of workflows. The initial cost of deployment of V_{DCR} is 0.10667554 ETH (177.5€) for a gas usage of 3,555,855. V_{DCR} is deployed in Ropsten[3].

Each partner can edit the running instance of their choice. Editing is done using the panel manager, a tool to update DCR graph descriptions. Users can add private and choreography interactions, condition, response, include, exclude, and milestone relations. They can also use the panel manager to remove and update events and relations. The panel manager implements integrity rules presented in subsection 5.3.1: the panel verifies the soundness of the desired change operation. Hence, we obtain a redesigned DCR graph that is correct-by-construction. After editing, if the panel manager detects a public change, it triggers the smart contract. The smart contract registers the request and forwards it to the partners identified by the panel manager. Each partner accesses the change request and answers back to the smart contract (cf. subsection 13). If the change request is accepted by all, the propagation mechanism described in subsection 5.3.3 is launched.

Smart contract evaluation costs The initial cost of the deployment of the motivating example instance is 0.04181024ETH (69.56€) or 1,393,674.67 gas. Indeed, the consensus algorithm used in the Ethereum blockchain is a proof of work [31]. Hence, each smart contract transaction (except reading transactions) is payable to compensate miners for computation costs. We evaluate the transaction costs to assess the computation costs related to the change negotiation and propagation functionalities.

In our motivating example, three changes occur. Change#1 is fully public: the choreography interactions ($e9 - e12$) replace $e1$. Eight public-to-public relations (four response relations, two include relations, and two milestone relations) are added to the set. The change initiator is Florist. In change #2, the choreography interaction $e6$ and the include and response relations between $e4$ and $e6$ are deleted. Change initiator is Customer. In change #3, the private event $p3$ is added to the graph, as well as two private-to-private relations (include and condition). The change initiator is Carrier A_1. In the following, we investigate the public negotiation and propagation smart contract costs for change #1.

Table 5.3 presents the transaction costs in ETH and euro (€) induced by the smart contract execution during the negotiation and propagation stages. We indicate the gas usage for each transaction. Gas is used to compute the transaction fees compensating miners for their computation power in the blockchain cryptocurrency.

Regarding the negotiation stage, Florist first launches the change request

[3]Ropsten smart contract address: 0x523939C53843AD3A0284a20569D0CDf600bF811b. This address can be used with Etherscan to access the record of transactions.

Table 5.3: Smart contract propagation costs (Neg.=negotiation, Prop.=propagation)

Stage	Step	Partner	Gas Use	Cost (ETH)	Cost (€)
Neg.	LaunchNego	Florist	225,485	0.00676454	11.3
	Case Decline	A_1	43,473	0.00130418	2.2
	Case Accept	A_1	53,582	0.00160745	2.7
		A_2	56,515	0.00169544	2.8
		A_3	56,515	0.00169544	2.8
Prop.	Update proj.	A_1	67,520	0.0020256	3.4
		A_2	67,130	0.0020139	3.3
		A_3	67,130	0.0020139	3.3
	Update proj.	Florist	55,299	0.00165899	2.8
	Update SC	Florist	913,740	0.0274122	45.6

for replacing one public task with a new fragment of two public tasks. The transaction fees for the request are 0.00676454 ETH (225,485 gas), the highest fees in the negotiation stage. Indeed, the fee for declining or accepting a role is worth around 0.0016 ETH (0.00130418 ETH to decline and 0.00160745 and 0.00169544 to accept). Nonetheless, all fees are of the same magnitude (0.001 ETH).

Regarding the propagation stage, the transaction fees of the smart contract correspond to two stages. First, the change endorsers, A_1, A_2, and A_3, apply the change effect to their private processes. No transaction fee is requested to fetch the IPFS hash of the new DCR choreography. However, a transaction fee is necessary to update the smart contract list γ_{propag} recording the projections. The smart contract notification of the local update is worth 0.0020 ETH for the endorsers (around 3.3€per local projection).

The change initiator, Florist, finally updates her projection. The cost to switch the workflow locally is 0.00165899 ETH. Florist sends a transaction to update the DCR choreography on-chain using the same tool used to deploy a new instance on-chain. The cost for switching the DCR choreography on-chain is 0.0274122 ETH. It is one order of magnitude higher than other transaction fees and similar to the cost of instantiating a new instance on-chain. Indeed, this similarity comes from the update of relation matrices and markings.

Hence, propagation transaction fees are higher than the negotiation ones. Additionally, the propagation cost mainly comprises the cost of the DCR choreography update. Finally, we appraise these costs to have more significance in a heavy negotiation scenario such as loan assessment.

We finally investigate execution times for the enactment of one trace. The reported execution time factors the transaction confirmation time obtained on the test network. On average, the execution time of on-chain interactions

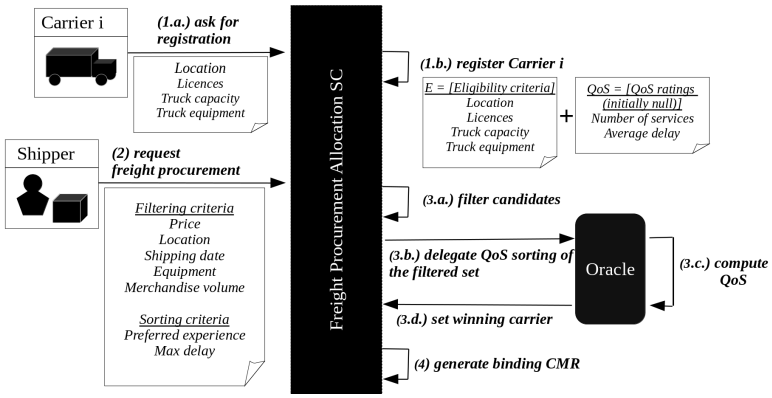


Figure 5.4: Blockchain-based FTSP mapping protocol

is 14.7s. Additionally, the average time for IPFS transactions is 7.6ms. The change initiator, Florist, needs to process four on-chain transactions and two off-chain transactions with IPFS. Each change endorser (i.e., $A1$, $A2$, and $A3$) must process three on-chain transactions and two IPFS transactions. Hence, the change management cycle takes 193s if all participants launch their transactions on trigger (4+3+3+3 on-chain transactions requiring 14.8s on average and 2+2+2+2 IPFS transactions requiring 7.6ms on average).

In summary, onchain negotiation requires less gas fees than the onchain propagation mechanism; the propagation cost mainly comprises the cost of the DCR choreography update. Both gas fees are proportional to the number of participants involved in the change. Finally, execution times depend on the latency of both smart contract calls, and IPFS calls.

5.5.2 QoS-based resource allocation

In this subsection, we implement (Section 5.5.2) and evaluate (Section 5.5.2) the QoS-based resource allocation mechanism presented in this chapter. We apply this mechanism to the logistic environment: we build a decentralized trust-free FTSP mechanism answering the needs of shippers striving to find the most qualified delivery drivers in open and dynamic markets (c.f. [86] for more details on the design science research approach). Shipment allocation is managed by a smart contract, the smart contract sorting carrier offers based on the QoS stored on-chain. We then focus on the academic contribution by discussing the nascent design principles emerging from the development and evaluation cycles in Section 5.5.2.

Implementation

The prototype architecture is multi-tiers, with (i) an application layer, (ii) the backend (smart contract, oracle) running on a blockchain, and (iii) a

Rest API for bridging the application layer with blockchain.

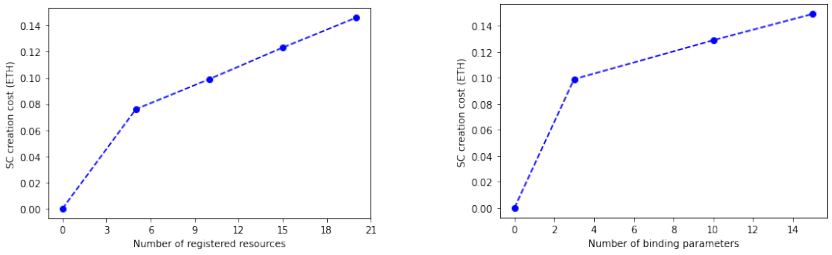
The front end is built with React and web3.js for API calls to the blockchain. The smart contract is written in Solidity and deployed to the Ethereum Ropsten testnet[4].

The blockchain-based FTSP mapping process follows the subsequent steps (cf. Figure 5.4). Initially, representatives from both shippers and carriers instantiate the FTSP mapping. The representatives state the smart contract's metrics to compute the on-chain QoS. Afterward, the carrier-shipper mapping can occur. Two stakeholders, carriers and shippers, interact with the FTSP smart contract. Carriers share eligibility information with the smart contract (e.g., location, licenses, truck capacity, and equipment) (step 1.a.). The smart contract associates this information with initially empty QoS metrics that record the history of their past services (e.g., number of deliveries and average delay) (step 1.b.). Shippers trigger the smart contract to find a carrier matching their needs by specifying filtering criteria (e.g., price, location, shipping date, equipment, merchandise volume) and sorting criteria (e.g., preferred experience or maximum delay) (step 2.). The smart contract will filter the candidates (step 3.a.) and delegate the QoS computations according to the required sorting criteria to the oracle (step 3.b.-3.c.). The *Provable* oracle generates API calls from our smart contract. Smart contracts pay a fee to the oracle at each query to compensate for computation costs. Thus, during platform instantiation, we fund the smart contract to ensure the processing of binding requests. The oracle provides to the smart contract the best-matching carrier (step 3.d.). Finally, the smart contract generates a digital delivery agreement as a decentralized version of the CMR (step 4.). The user (shipper or carrier) can retrieve all contracts bound to their public address.

We refined the prototype following two iterative cycles. User tests underlined the need to display computation information to enhance trust in the system. Indeed the black-box effect appeared, linked to users using a technology they are not experts with occurs. Users can only trust the machine as they do not know the underlying protocols. This dependency may trigger mistrust. We thus displayed all resource profiles to the users and the ratings obtained for each resource. The second evaluation round consisted of focus group discussions. Testimonies underlined the need for an almost entirely autonomous system. We revisited the artifact by merging the matching and allocation stages into one single step.

Power asymmetry between carriers and shippers is addressed through shared governance, which can occur through an open or consortium blockchain: each actor participates equally in the consensus protocol.

[4]Code is available at `https://archive.softwareheritage.org/swh:1:dir:cc1dc66c637f8427836d808745f1bf1630816527;origin=https://github.com/tiphainehenry/smart-logistics;visit=swh:1:snp:d97e3a03446f13ef7112faba5abec02254568449;anchor=swh:1:rev:e06b091de377c1bfe29644fb2b539af99381dd1e`.



(a) Creation cost according to the number of resources initially registered

(b) Creation cost according to the number of binding parameters

Figure 5.5: Smart contract creation cost depending on the number of resources and binding parameters

Information asymmetry is addressed by making it accessible to shippers carriers' former services rating on the blockchain ledger. Additionally, a power balance is reached by putting into escrow the carrier's pay, as well as the carrier's caution. Hence, the shipper and the carrier are bound after the mapping validation. The smart contract enables platform operating costs reduction by managing allocations autonomously. The smart contract also allows contractual flexibility as it speeds cash flow; payment is enacted right after the shipping is settled. Finally, the combination of (1) the blockchain ledger, (2) the smart contract protocol, and (3) the oracle-based QoS computations ensures tamper-proof and objective management of the allocation requests. With the proposed architecture, oracles share QoS-computation results with the blockchain, and shippers and carriers interact directly with the blockchain to manage delivery requests. Hence, there is no more intervention from a trusted third party. Thus, a blockchain system can manage carrier-shipper mappings in a secure, trusted, and reliable way. As a side note, regarding carrier coopetition, a carrier wishing to delegate the delivery service to another carrier can take the role of the shipper. The generated CMR contracts hold a link to the initial CMR contract. When the new carrier-carrier CMR is settled (i.e., the delegated delivery has been fulfilled), the status of both CMR contracts is set as settled, and the smart contract transfers the corresponding escrowed payments. The QoS rating obtained for the delivery is stored in both carrier profiles.

Evaluation

We now evaluate the prototype by investigating the gas fees necessary for deploying the resource-binding smart contract. We then evaluate the smart contract's resource-binding fees and latency.

Deployment Costs We evaluate the smart contract deployment costs on Ropsten (cf Figure 5.5). The deployment of the resource-binding smart contract into the blockchain network is payable as miners need compensation for the computation resources linked to the consensus protocol.

We use the August 24th, 2022 conversion rate (1ETH=1,663.76€) in the following.

Figure 5.5a depicts the smart contract creation fee variations according to the number of registered resources (i.e., its id, availability, and the set of binding criteria). The smart contract creation fee increases with the number of resources initially populating *Candidates*, from 0.08 ETH for five resources to 0.15 ETH for twenty resources. Figure 5.5b depicts the smart contract creation fee according to the number of binding resources (i.e., the number of filtering and optimization criteria). The smart contract creation fee similarly increases with the number of binding parameters, from 0.09 ETH for three binding parameters to 0.15 ETH for fifteen parameters. Experiments show that creation costs depend on the size of *Candidates*, and consequently on (1) the number of resources initially registered and (2) the number of binding parameters: the bigger the candidates' matrix, the more expensive the smart contract creation fees. Each candidate could pay for being added to the matrix. However, scalability issues could arise if the number of registered candidates generates a smart contract cost above the blockchain limit threshold.

Binding Costs We then evaluate the smart contract's resource-binding fees and latency (cf Figure 5.6). Each resource-binding request requires the contractor to pay a smart contract transaction fee. This fee compensates the blockchain miners for processing the binding smart contract function. The average matching cost corresponds to the oracle fees required for GET requests to the external QoS computation API. The price is manually tuned to fit price requirements. In our experimental setting, it is thus constant for the oracle. The matching cost is 0.47 ETH, and the contracting cost is 0.0021 ETH. A resource-binding request is relatively expensive compared to traditional Ethereum transactions, often around 0.01 ETH at the time of writing[5]. The binding price depends on the string conversion required to transform the parameters into an URL for the API call. Other oracles such as Chainlink use dedicated tokens instead of ETH and could decrease this cost.

We trigger ten times the resource-binding smart contract to evaluate the latency of the proposed solution while reducing network noise. We measure the oracle call and callback time. Figure 5.6 represents the results displaying the oracle call and callback, as well as the cumulative response time. The median oracle response time is 19 sec. The median callback response time is

[5]See https://bitinfocharts.com/, accessed on February 28th, 2021

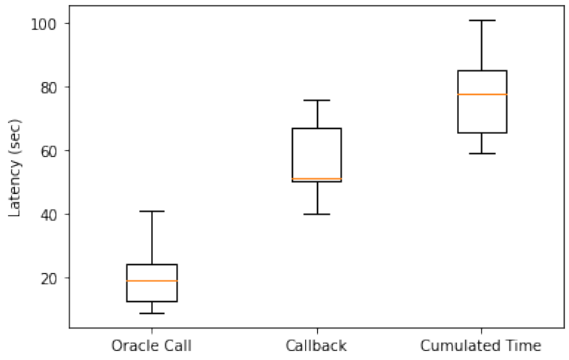


Figure 5.6: Smart contract resource-binding latency

51 sec. The median cumulated time is 78 sec. The cumulated time boxplot is left-skewed. Most requests take between 78 sec (the median) and 85 sec (the upper quartile), but some calls answer more rapidly. Overall, the oracle calls are shorter than callback times. Oracle calls are less spread than callback response times. The network capacity and the oracle-related block validation time may explain the disparity between oracle calls and callbacks. At runtime, the resource-binding cost is 0.47 ETH and takes one minute and a half to process on average. Depending on the targeted market, the gain in trust and transparency could justify these costs.

In summary, smart contract deployment costs depend on (1) the number of resources initially registered and (2) the number of binding parameters to be analyzed. The fees required for a smart contract binding request correspond to the oracle services. Hence, a resource-binding request is relatively expensive compared to traditional Ethereum transactions in our experiment, but is highly dependent on the oracle service triggered. Finally, matching time are highly impacted by the oracle response time. Hence this mechanism should not be considered for time-dependent matching applications.

Theory generation

We propose three nascent principles stemming from the prototype development cycles that support a successful artifact development [175]. We then discuss these findings in the light of other design science research papers.

Table 5.4 presents the three emerging design principles detailed hereinafter. First, **understandability** of the smart contract shipper-carrier allocation is needed to encourage non-technical users' adoption (DP1). Delegating the allocation process to the blockchain triggered some concerns during the user testing stage. More precisely, the use of an oracle raised concerns related to integrity gains. Having no access to the decision protocol, several users were puzzled regarding the result, as they did not have access

Table 5.4: Blockchain-based FTSP mapping nascent design principles

Understandability	The smart contract shipper-carrier allocation protocol must be displayed to the shipper
Automation	The FTSP process must be carried on in an end-to-end fashion by the smart contract
QoS-metrics privacy	The allocation mechanism must keep sensitive data off-chain in competitive markets

to the smart contract reasoning for such allocation. Thus the FTSP platform should display the decision protocol alongside the data output to avoid the technological black-box effect. Blockchain architecture requires strong technical skills to understand its benefits (decentralized governance, tamper-proof data, autonomous scripts). For non-expert users, the prototype thus consists of a box where data comes in (e.g., their delivery request) and out (e.g., a shipper/carrier mapping) without understanding what happens inside the box. We propose to display the shipper computation outputs for each profile to counter the black-box issue. Moreover, **automation** of the FTSP process is necessary (DP2): the FTSP process must be carried on in an end-to-end fashion by the smart contract. Indeed, process optimization is one of the significant benefits underlined by both testers and experts. Experts have made suggestions for enriched process automation. More precisely, they foresee blockchain-based FTSP mapping as autonomous agents managing request bundling, re-pricing, or matching. To integrate automation into our artifact, we merge the service request and contractualization stages into one step. The service status is tracked and managed within the blockchain for integrity and efficiency. Nonetheless, oracle latency is at stake regarding allocation queries, as it can take up to several minutes to process a request. According to one expert, *'latency variability seems acceptable for long-term markets where contracts can be set for a year or so [...] In spot markets, the service provider could become unavailable before the end of the query.'* Thus, the automation of the FTSP mapping should take into account allocation latency. Finally, **metrics privacy** should be considered (DP3). The allocation mechanism must keep sensitive data off-chain in competitive markets. On the evaluation side, the disjunction between private and public data appears in several testimonies. Such sensitivity needs to adjust to the market at stake. Price and capacity data are sensitive and should remain private to other carriers' eyes. Though our prototype saves QoS metrics to the blockchain, a better option is to encrypt the price and volume metrics. We leave it for future work.

These nascent design principles resonate with several design principles proposed in the design science research literature. Accessibility of blockchain-

based applications to non-technical users is one of the design principles of [143]. The design principle (DP1) stands on the necessity to explain the smart contract decision-making process adequately to users. Though blockchain smart contracts provide automation and transparency, protocol and data trust come if users understand the mechanisms motivating a decision (a delivery request allocation here). (DP1) is thus reinforced as a design principle. Moreover, allocation automation (DP2) resonates with the need for digitizing chapter-based processes in [143]. The issuance of CMR contracts implies the production of several paperwork copies that need to be approved by both shipper and carrier at the start of the delivery. Consequently, the paperwork slows the delivery process, reduces the overall efficiency, and implies tampering risks. (DP2) also resonates with the potential for smart contracts to manage processes presented in [3]. (DP2) is thus also reinforced as a design principle. Lastly, the privacy of allocation metrics (DP3) resonates with the need to manage in an off-chain fashion private and sensitive data. These needs were underlined in [5] that performed a design science research study in the trucking industry: private carrier and shipper information should remain off-chain. The need for sensitive data privacy is also underlined as a design principle in [187] and [119]. QoS metrics act as private and sensitive data in our setting. In the future, the QoS metrics of our proposed mechanism should consequently be ciphered so that they cannot be understandable by other competitors or kept off-chain. Thus, (DP1-3) can be considered as design principles for developing a blockchain-based FTSP solution.

The contribution of this experiment to theory lies in investigating the use of the blockchain in the logistics industry concerning coopetition [20]. The current FTSP-related literature mostly focuses on the traditional carrier-to-shipper market. Little attention has been paid to the carrier-to-carrier market, and the application of blockchain and smart contracts in such a market is still a new research topic. The FTSP process is applied in a cooperative/coopetive environment as (1) competing carriers agree to use the same platform to gain a service auction, and (2) competing shippers agree to use the same platform to find an adequate carrier. Both shipper and carrier participants have a shared interest in using the blockchain-based FTSP platform as the system has the potential to reduce operational costs, increase the visibility of the carriers, and transparency and integrity of the deliveries.

5.6 Conclusion

This chapter proposes two mechanisms to add flexibility to business-based business process management systems.

We present a trustworthy change propagation mechanism comprising three steps (change introduction, negotiation, and propagation) for declarative

choreography instances. Through this contribution, we aim at addressing Objective 3. Regarding change introduction, a partner in a running DCR choreography instance defines changes in its private process. Here, we declare rules that specify the allowed and prohibited changes to ensure that no deadlocks nor livelocks occur after the change. In other words, these rules provide a correct-by-construction DCR choreography. This ensures that the DCR choreography is safe and terminates in an acceptable state, i.e., deadlock-free after the change. The solution manages local changes off-chain. Change requests impacting an interaction trigger the on-chain negotiation phase. If the negotiation succeeds, the smart contract propagates the change effect to the partners affected by the change. We suppose that all partners project trustfully the updated DCR choreography. The smart contract records partners' involvement in a tamper-proof fashion during the change negotiation and propagation stages. If misbehavior occurs, the blockchain logs can be used as a shared source of truth.

We also propose a transparent and autonomous resource-binding protocol, hence answering Objective 4. To do so, we leverage smart contracts as a matching and binding mechanism. The smart contract (i) matches a contractor with the best service provider and (ii) generates a digital agreement stored in the blockchain to keep track of the service fulfillment. Transparency appears at the data management level as the QoS of all available service providers is stored in the blockchain. Transparency also occurs at the process level as smart contracts objectively manage the binding.

Hence, through these research works, we address our second research question (RQ2), which focused on the need to increase the flexibility of cross-organizational processes managed on-chain. Moreover, we address the trust challenge through smart contracts enforcing an agreed-upon binding protocol and through the blockchain ledger offering a tamper-proof history of services. In both approaches, blockchain addresses service providers' and issuers' collusion risks by design, as no single entity controls the cross-organizational business process management platform. We also address the need for process automation through management through smart contract hiring, contractualization, and settlement stages. Our experiments finally confirm the feasibility of these mechanisms, hence validating that we addressed (RQ2), as well as Objectives 3 and 4.

The work presented in Section 5.3 has been published in the International Conference on Business Process Management[27]. The work presented in Section 5.4 has been published in the Hawaii International Conference on System Sciences [90] and in the IEEE International Conference on Services Computing[87].

In the following, we investigate the need for privacy of sensitive metrics such as bidding content or payment transaction amount underlined by (DP3).

Chapter 6

Sealed-bid Auctions and Privacy-preserving Payment

Contents

6.1 Introduction

In the previous chapter, we proposed a dedicated allocation system that maps delivery requests to the best available carrier. Smart contracts take action

regarding the bids evaluation, service completion, and payment without requiring a trusted third party. Such a system can, for accounting and compliance purposes, (1) record all settled transactions in a tamper-resistant fashion and (2) keep track of delivery information provided by different oracles and tracking sensors. This configuration ensures information transparency as the involved members can access past allocation history stored in the ledger and access the tamper-proof delivery information.

In this context, auditability of the protocol and decision-making should be accessible to all participants to ensure trust in the system. If a claim occurs, participants must be able to rely on a tamper-proof database storing the history of services. Moreover, bid privacy should be enforced to limit information asymmetries and collusion risks between actors [7], as bids may comprise differentiating criteria. Finally, once the service is fulfilled by the service provider, the payment should remain private to preserve the confidentiality of the exchange, e.g., if the price for a service varies from one partner to another based on negotiation outcomes.

From the related work analysis presented in Section 3.5.2, dealing with data privacy in a blockchain framework raises challenges regarding access control enforcement and processing of sensitive data [170, 46, 208]. *Regarding sealed-bid auctions*, despite several publications of blockchain-based allocation protocols to map services and their procurement, there is a research gap in allocating services based on multi-objective sorting in a privacy-preserving fashion [132]. *Regarding payment on blockchain systems*, a middle-ground trust hypothesis considers banks as reliable proxies for payments to palliate a complex blockchain and key management setup scheme. Hence, in [79, 34], banks are leveraged as trustworthy intermediates to carry on on-chain payments. Cryptography technics such as zero-knowledge proof can moreover ensure privacy [34]. Nonetheless, in both approaches, auditability is not directly addressed.

Hence, this chapter sets out to answer the following questions:

1. *How to manage and compute numeric information in a privacy-preserving fashion using fully homomorphic encryption?* (c.f., Section 1.3, RQ3.2)

2. *How to ensure payment privacy in a cross-organizational process that preserves privacy while offering public auditability?* (c.f., Section 1.3, RQ3.3). In light of the related work, we refine this question as follows: *How to implement a decentralized token payment system building on banks as trustworthy proxies that preserves privacy while offering public auditability?*

In this chapter, building on the motivating example presented in Section 6.2, we propose two main contributions aiming at answering the aforementioned research questions:

1. First, we propose in Section 6.3 a solution to leverage FHE in a blockchain context while preventing deciphering and collusion issues on the competitors' side: we do so by combining the hybrid RSA/AES ciphering technic with smart contracts and oracles to compare ciphered vectors: (1) FHE encryption is used to compare bids that have been previously ciphered; (2) the hybrid RSA/AES encryption scheme is used to transfer ciphered bids confidentially in the blockchain, avoiding deciphering attempts of the FHE layer on behalf of competitors. Our approach differs from other literature approaches by using an oracle to delegate FHE comparisons while preventing smart contracts' rising transaction costs.

2. We propose in Section 6.4 a solution that uses a bank and a per-collaboration payment token linked to a random value to address this research question. Parties can use per-collaboration tokens to proceed to multiple payments while preserving the values' privacy. Token payment can be programmed to verify conditions coded in a smart contract, put into escrow, and carry partial payment. Additionally, external peers can audit trust-worthily token transactions as they are stored on-chain.

Finally, we evaluate both contributions in Section 6.5.

6.2 Motivating example

In this section, we build on the motivating example of the flower delivery business process presented in the introduction.

Figure 6.1 illustrates a sequence diagram for the blockchain-based flower delivery service, where the blockchain acts as a trustworthy and autonomous allocation service and settlement.

The florist uses the resource-binding smart contract to hire a carrier for the shipment (Figure 6.1, step 1). The carrier choice is paramount as the quality of the delivery directly impacts customer satisfaction and may have significant financial consequences for the florist's business. Hence, the florist sets a maximum price threshold and a minimum capacity for offers to be eligible: the truck capacity should be at least 5m3, and the service price below 20$/m3/km. Additionally, if two or more bids are suitable, the offer must be allocated to the carrier with the optimal bid.

Previously registered on the smart contract, four carriers, A_1, A_2, A_3, and A_4, compete for the service. Table 6.1 presents the candidates' bids with a normalized truck capacity. The resource-binding smart contract compares the QoS of competitors and allocates the delivery service to one of the carriers (Figure 6.1, step 2). The underlying allocation mechanism and data processed should be trustworthy to discourage actors' collusion and data tampering.

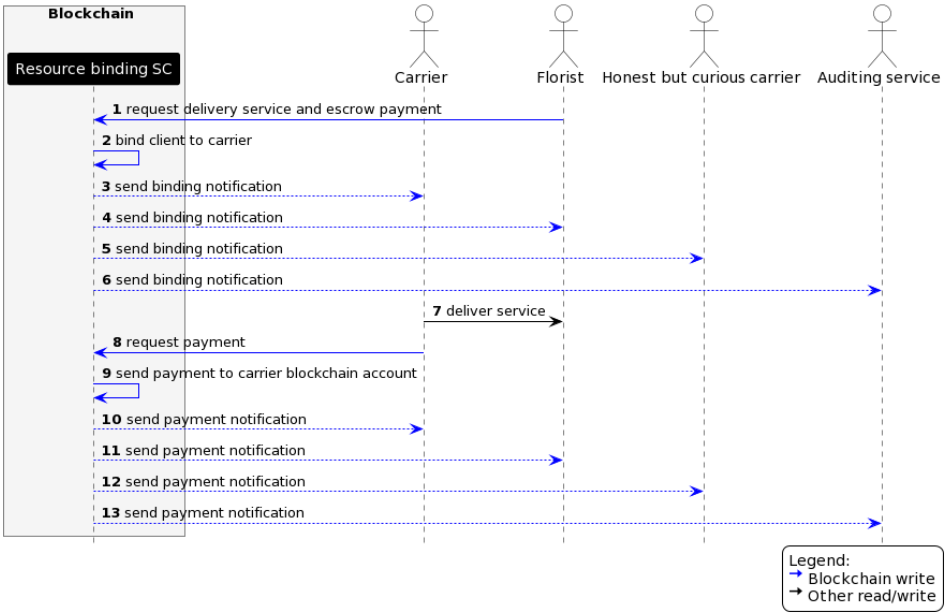


Figure 6.1: Sequence diagram of a blockchain-based service payment

Additionally, sensitive information encapsulated into carriers' offers should remain confidential (i.e., not accessible to other carriers). Hence, the following question arises: *how may smart contracts be leveraged to allocate the best carrier to a service request while ensuring the confidentiality of price and capacity?* The resource-binding smart contract sends an event notification to notify participants of the binding. Participants listening to the event will thus be notified (Figure 6.1, steps 3-6). The delivery service occurs off-chain (Figure 6.1, step 7), and the carrier requests his pay (Figure 6.1, step 8).

The smart contract reallocates the service funds put in escrow to the carrier blockchain account (Figure 6.1, step 9). It sends a payment notification

Table 6.1: Competing carriers offers

Carrier	Availability	Location	Price[1]	Capacity[12]
A_1	D1	ϕ	10$/m3/km	10
A_2	D1	ϕ	5$/m3/km	5
A_3	D2	ϕ	10$/m3/km	5
A_4	D2	ϕ	10$/m3/km	8

[a] Sensitive metrics that should remain private.
[b] Truck capacity is normalized to scale 10.

(e.g., under the shape of an event (Figure 6.1, steps 10-13)). The payment is recorded as a transaction in the ledger. Hence, an auditing service can access the payment transaction, e.g., assess compliance with regulation laws or ease claim resolution.

Nonetheless, the payment value is accessible to carrier competitors (step 12). For example, competitors could use this information to renegotiate contract terms with the client, which would hamper the adoption of a blockchain-based solution. Consequently, the payment should remain confidential to avoid any privacy leakage in such a competitive situation. Hence a second question arises: *how to carry on a privacy-preserving payment while ensuring the auditability of the payment transactions?*

6.3 Sealed-bid auctions

This section presents the trustworthy sealed-bid auctions protocol aiming at preserving the privacy of sealed bid offers while preserving the trustworthiness and auditing functionalities of blockchain. The solution leverages AES, RSA, and FHE encryption algorithms. We first introduce basic concepts related to these algorithms in Section 6.3.1. We then provide an overview of the approach in Section 6.3.2. We finally dive into the details of each protocol stage in the remaining subsections: first key initialization in Section 6.3.3, offers ciphering and forwarding in Section 6.3.4, and finally ciphered offers comparison in Section 6.3.5.

6.3.1 Basic concepts on encryption technics

(Fully) Homomorphic encryption

Homomorphic encryption, first proposed by Rivest et al. in 1978 [179] belongs to the family of symmetric encryption: the same key is used to cipher and decipher a number. This encryption protocol often helps address the millionaire problem where two millionaires want to compare their wealth without disclosing the exact amount of money they own [118]. The term *homomorphism* refers to a homomorphism between plain and ciphertext spaces. Indeed, this method preserves algebraic operations: one can carry computations on encrypted data without requiring a decryption key.

Partial homomorphic encryption algorithms exist where only one type of algebraic operation is possible (e.g., Paillier algorithm [160]). An FHE scheme extends partial homomorphic encryption as it supports all algebraic operations. The scheme was proposed by Craig Gentry in 2009 [75].

Figure 6.2 illustrates the FHE scheme. Let's consider a number x and its ciphered version c_x. Computing an algebraic operation g on x will lead to g(x)=y. Computing this operation on c_x will lead to c'_x. If one tries to decipher c'_x, the result will be y. Hence, one can carry on an algebraic

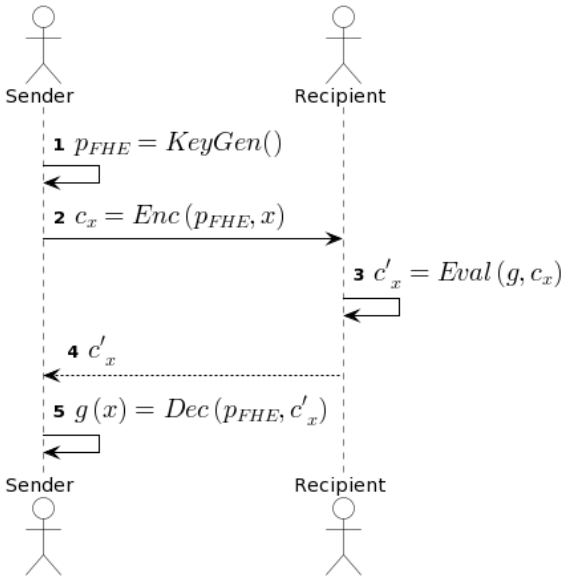


Figure 6.2: FHE Scheme

operation on a ciphertext with homomorphic encryption without having access to plaintext.

In more detail, the encryption first consists of translating plaintext into a binary. Afterward, the binary goes through a set of Boolean circuits composed of NAND gates as it is the only gate from which it can generate the AND, OR, and NOR gates. Each circuit corresponds to an operation (addition, multiplication, etc.). Among examples of usage of FHE stand the comparison of two [25] or more [24, 200] numbers. It is to note that successive operations may trigger noise; hence the deciphered output of the computation may not be exact. To circumvent this issue, one may use bootstrapping operations to reduce the noise [38].

Hybrid RSA/AES encryption

RSA stands for Rivest Shamir Adleman; it refers to the name of its three inventors. RSA is an asymmetric cryptography algorithm [180]: two keys are necessary to encrypt and decipher messages (c.f., Figure 6.3). It has two uses, signature and ciphering. Anyone wishing to cipher a plaintext uses the public key. The entity generating the two keys owns the private key, which is used to decipher incoming ciphertext.

AES stands for Advanced Encryption Standard. It refers to the encryption algorithm also known as Reijindael encryption [45]. It consists into a symmetric encryption algorithm leveraging (1) a plaintext to be ciphered, and (2) variable key lengths.

The RSA/AES hybrid encryption protocol, first proposed in [129],

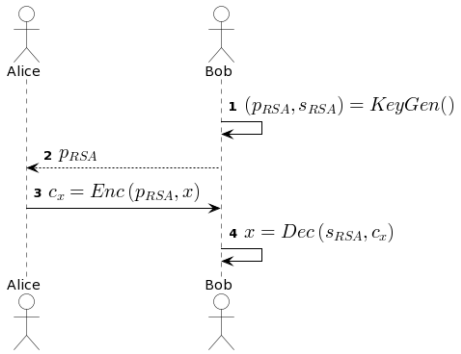


Figure 6.3: RSA Scheme

leverages RSA and AES, which are respectively asymmetric and symmetric. They are combined to cipher and decipher long ciphertexts. The sender, Alice, wishes to send a plaintext of a consequent length in a secured fashion to a recipient Bob. She will proceed as follows: (1) Bob generates a private/public key pair for the RSA algorithm, (2) Bob sends to Alice its public RSA key, (3) Alice generates an AES key randomly and ciphers its plaintext with the symmetric AES algorithm, (4) Alice ciphers the AES key with the RSA encryption protocol, using Bob's RSA public key, (5) Alice sends to Bob her ciphertext ciphered with AES and her ciphered AES key. Hence, only Bob can decipher Alice's ciphertext.

6.3.2 Overall approach

Figure 6.4 presents the main stages of our sealed-bid auctions system. First, we initialize the ciphering keys for the AES, RSA, and FHE algorithms. To preserve the privacy of sensitive offers during the allocation protocol, we separate the management of the encryption keys between five stakeholders. The second stage consists of the smart contract gathering eligible candidates' ciphered sensitive information and the RSA-ciphered AES key. The third stage consists of deciding on the best offer: the oracle receives the ciphered keys and bids, deciphers the key, then uses the deciphered key to access FHE-ciphered offers, and finally computes the best offer by comparing the FHE-ciphered data. The binding is then enacted, and service management can begin[87]. We describe each stage in more detail hereinafter.

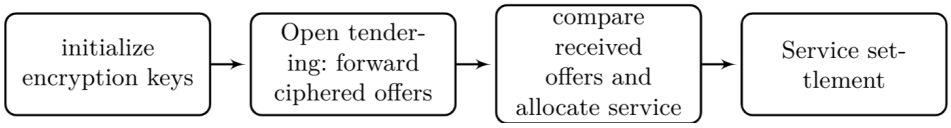


Figure 6.4: Privacy-preserving allocation and settlement stages

The privacy-preserving allocation smart-contract holds a list of registered service requests. Each registered service request comprises the blockchain addresses of the service requester and allocated service provider (null at first). It also comprises the open tendering and the service settling status. The open tendering comprises the list of submitted bids, with each bid attached to its issuer blockchain address, and the open tendering status. The latter can be open: the smart-contract accepts new bids. It can be pending: the smart-contract does not accept bids anymore and proceeds to the bids comparison. It can finally be closed: the comparison has terminated, and the service has been allocated. In this case, the blockchain address of the service provider is populated with the address of the winning bid issuer. The smart-contract also comprises three functions to manage the mechanism:

- a function registers or updates the RSA public key. The RSA public key is updated by the oracle service only, the RSA public key value is necessary for bidders to cipher their bids (c.f., Section 6.3.3);
- a function registers new bid offers: the bidder must forward the hash of the ciphered offer and the hash of the ciphered AES key (c.f., Section 6.3.4);
- a function closes the open tendering and launches the offer comparison by triggering an oracle service (c.f., Section 6.3.5);

Used formalism

- We refer to k_ as the ciphering key for the symmetric encryption of type _.
- We refer to $(s\_, p\_)$ as the couple of respectively secret and public keys for the asymmetric encryption algorithm of type _.
- With x a natural number, we write C_ the ciphering function of type _. With x a natural number, then $c = C\_(x, p\_)$ is the ciphered version of x obtained after applying the ciphering protocol of type _. With x a natural number, we write D_ the ciphering function of type _, and $x = D(c, s\_)$. For symmetric encryption, k_ is used for both ciphering and deciphering.
- We define a service provider offer O as a vector of offers, where $O[i]$ is an array comprising sensitive metrics to evaluate.
- We refer to O_{enc} as the array where each element of O has been ciphered with the public key of the encryption algorithm of type _: with i $\in [0, size(O) - 1]$, $O_{enc}[i]$=$C(O[i], p\_)$.
- We refer to L as the list of ciphered aggregated offers.

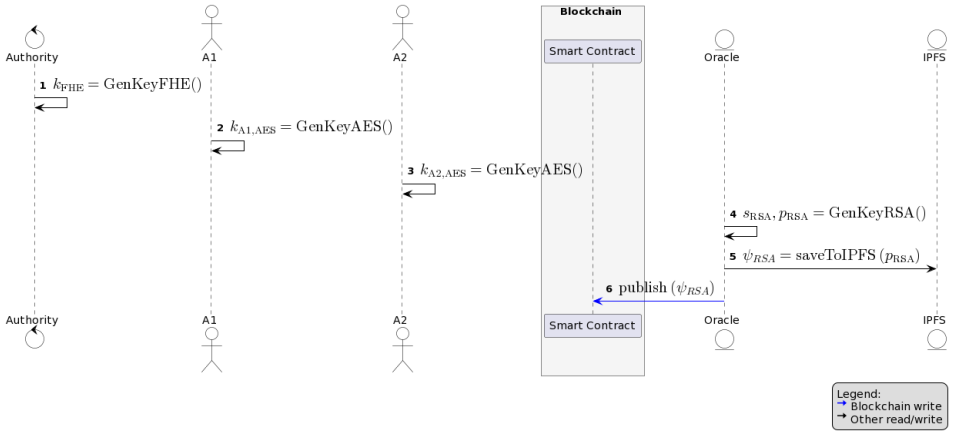


Figure 6.5: Initialization of cipher keys

6.3.3 Key initialization

Our approach uses three encryption algorithms: FHE, AES, and RSA. FHE is used to do blind calculations by comparing ciphered offers. Service providers can decipher each other's data using the same key. An asymmetric encryption layer is thus necessary to preserve offers confidentiality. A standard asymmetric encryption algorithm is RSA. Nonetheless, the FHE ciphertext length is too large for RSA but not for AES. As AES is symmetric, the issue of confidentiality rises again, and it must be combined with an asymmetric scheme. To do so, we use the hybrid AES/RSA encryption scheme presented in [129].

Figure 6.5 presents the sequence diagram of the initialization of the three RSA, AES, and FHE ciphering keys. In this approach, three separate entities manage the RSA, AES, and FHE key generation to avoid any deciphering attempt by one of the entities.

- The service providers' authority generates the FHE key k_{FHE} (step 1) that will be forwarded to service providers A_1 and A_2 once the auction time finishes. The service providers will use this key to cipher their offers.
- Each service provider generates its own AES encryption key (step2-3). Only one key is generated due to the symmetric behavior of AES encryption.
- The oracle generates the pair of private and public RSA keys s_{RSA} and p_{RSA} used to cipher and decipher messages with the RSA algorithm (step 4). The private key s_{RSA} remains secret for the oracle to retrieve offers ciphered with the RSA algorithm. The public key p_{RSA} is saved in IPFS to reduce storage costs in the smart contract (step 5). The

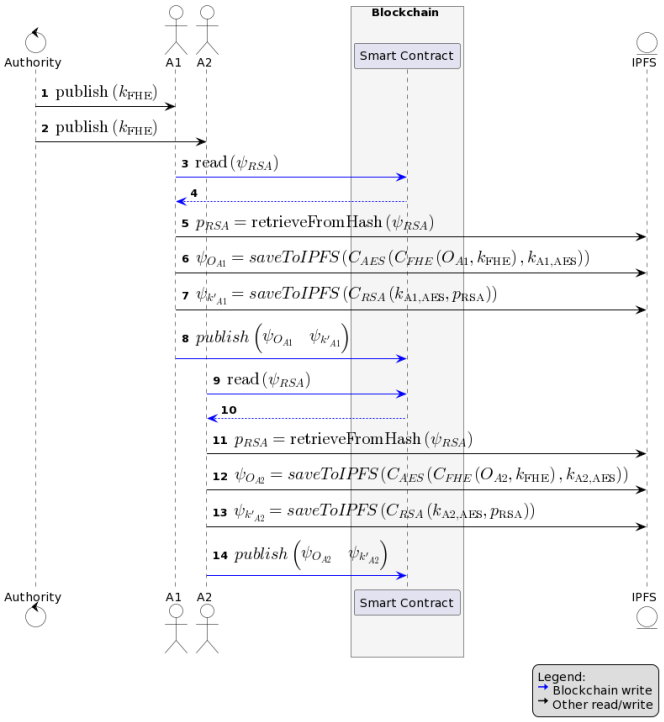


Figure 6.6: Ciphering and gathering offers

IPFS hash used to retrieve the public key on the IPFS network, ψ_{RSA}, is forwarded and stored in the smart contract (step 6).

Let's consider the delivering procurement allocation of our motivating example. The carriers' authority generates k_{FHE}. The four bidders of our motivating example, A_1, A_2, A_3, and A_4, all subscribe to the smart contract to participate in the bid. They each generate a private AES encryption key k_{AES}. The oracle generates the keyset $\{s_{RSA}, p_{RSA}\}$ and publishes the public RSA key to IPFS. It then publishes the hash of p_{RSA} to the smart contract.

6.3.4 Generating and forwarding FHE-ciphered offers to the smart contract

Figure 6.6 depicts the generation of ciphered offers by each competitor and the gathering of offers into the smart contract.

The service providers' authority forwards the FHE key to each available candidate via private channels (steps 1 and 2). In our motivating example, the service providers' authority sends k_{FHE} to A_1 and A_2 via private channels.

Afterward, service providers cipher their offers. Let's consider service provider A_1 in Figure 6.6. She triggers the smart contract to retrieve the IPFS

hash of p_{RSA} (step 3). The smart contract returns the hash to A_1 (step 4). A_1 connects to IPFS to retrieve p_{RSA} (step 5). Afterward, A_1 ciphers her offer O_{A1} twice using FHE encryption first and then AES encryption. She applies the following ciphering algorithms: $C_{AES}(C_{FHE}(O_{A1}, k_{FHE}), k_{A1,AES})$ using her personal AES key $k_{A1,AES}$.

Afterward, service provider A_1 forwards her offer to the smart contract. To do so, she first saves the content of the offer in IPFS (step 6). She then ciphers $k_{A1,AES}$ using the RSA algorithm and forwards it to IPFS (step 7). She finally publishes the hashes of her ciphered offer and key to the smart contract (step 8).

Service provider A_2 proceeds to the same steps in parallel (steps 9-14).

In our motivating example, delivering driver A_1 computes her ciphered AES key is expressed as $C_{RSA}(k_{A1,AES}, p_{RSA})$.

She also processes her ciphered offer as follows: $O_{A1_{enc}} = \begin{bmatrix} C_{AES}(C_{FHE}(price, k_{FHE}), k_{A1,AES}) \\ C_{AES}(C_{FHE}(capacity, k_{FHE}), k_{A1,AES}) \end{bmatrix}$.

6.3.5 Compare and allocate the service to the best offer

Once all offers are forwarded, the smart contract delegates the offers' comparison to the oracle. Delegation occurs to avoid intensive computations carried on the blockchain network. Indeed, the more calculation on the blockchain, the more expensive the transaction's time and transaction fees.

Shuffling and forwarding ciphered offers The smart contract interchanges the randomly received ciphered offers (Figure6.7, step 1). By so doing, we prevent an honest but curious oracle behavior that could lead to an information leakage on the order of candidates' submissions and hence on the winning offer. The smart contract then forwards the interchanged offers to the oracle for an argmax computation (Figure6.7, step 2).

To do so, the smart contract sends an API request to the oracle, specifying the ID of the bid comparison to analyze. Meanwhile, the smart contract sends an event comprising the list of hashes of the ciphered offers and AES keys.

In the following, we refer to the interchanged offers as $\Pi_{O_{A1}}$ and $\Pi_{O_{A2}}$.

In our motivating example, the smart contract interchanges the received ciphered offers $[O_{A1_{enc}}, O_{A2_{enc}}]$ randomly and forwards the interchanged offers to the oracle for an argmax computation.

Retrieving FHE offers and computing the mean The oracle first retrieves the list of ciphered offers and keys hashes specified in the comparison request event. It then connects to IPFS to fetch the offers and keys (Figure6.7, steps 3-4). The oracle uses the private key s_{RSA} to decipher each AES key of A_1 and A_2 (Figure6.7, step 5). It then uses each AES key to process the

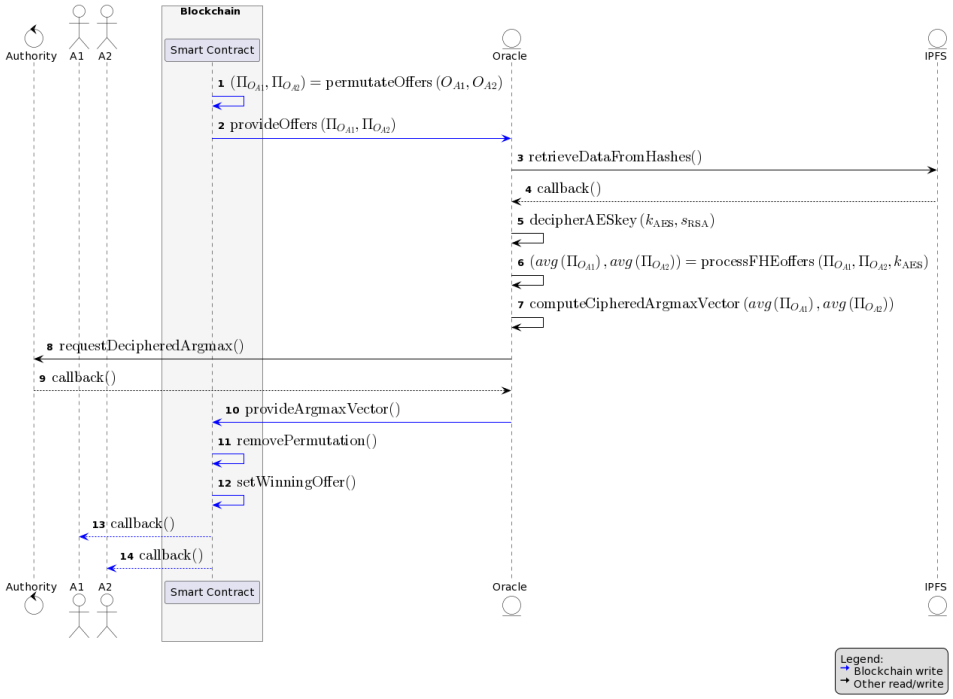


Figure 6.7: Ciphered comparison and allocation

corresponding FHE offers (Figure6.7, step 6): it will first decipher the offers to retrieve the FHE offers and then compute each offer's ciphered mean.

In our motivating example, the oracle deciphers the RSA layer of the AES key of A_1. It will then use this AES key to remove the AES encryption layer of $\Pi_{O_{A1}}$. The oracle then computes the mean for the offer submitted by A_1: it obtains $avg(\Pi_{O_{A1}}) = (C_{FHE}(capacityA1, k_{FHE} + C_{FHE}(priceA1, k_{FHE})/2$. The computation of $avg(\Pi_{O_{A1}})$ gives $C_{FHE}(8, k_{FHE})$. The computation of $avg(\Pi_{O_{A2}})$ gives $C_{FHE}(5.5, k_{FHE})$.

As a side note, we chose a simple average as possible use. Alternative computations can be carried on to aggregate offer metrics. For example, Pareto optima could be computed or weighted mean with weighting factors defined in the smart contract in the service request.

Comparing offers The oracle then proceeds to the pairwise comparison of the processed offers (Figure6.7, step 7).

In FHE, the maximum between two ciphered numbers is also ciphered [200]. Due to noise addition during ciphering and comparison, it is not possible to reuse this result for another comparison without deciphering it. Hence, it is not possible to directly compare more than two ciphered numbers.

We propose in Algorithm 6 a way to circumvent this issue, using pairwise

Algorithm 6: Ciphered numbers comparison algorithm

Data: L the list of ciphered aggregated offers, c_1 and c_n ciphered 1 and n, and p a prime number large enough to fulfill the condition $p > max(L)$

Result: B a vector of size len(L) comprising the argmax of L

```
1  Function cipherCompare(O):
2      var B ← [];
3      for i ∈ [0, length(L)] do
4          for j ∈ [0, length(L)] do
5              if i != j then
6                  var m ← c_max(L[i], L[j]);
7                  var b ← testEquality(L[i], m, c_1, c_n, p);
8                  B[i] ← B[i] + b;
9      return B
10 End Function
```

Algorithm 7: Function *testEquality* performing the equality testing of numbers ciphered with FHE.

Data: p a large prime number, c_{one} a ciphered one, c_n a ciphered number where n > 1, c_x and c_y two ciphered numbers where x and y are non divisible by p.

Result: Returns c_0 if x != y, and c_n otherwise

```
1  Function testEquality(c_x, c_y, c_one, c_n, p):
2      var t ← c_x − c_y;
3      for i ∈ [0, p − 1] do
4          t ← t ∗ (c_x − c_y)
5      t ← c_n(c_one − t) b
6  End Function
```

comparisons between offers as suggested in [200], and equality testing using the Fermat little theorem. B will store the argmax of the ciphered offers (Algorithm 6, line 2). A pairwise comparison is launched on L (Algorithm 6, line 3-11). More precisely, the maximum c_{max} between two elements $L[i]$ and $L[j]$ is assessed (Algorithm 6, line 6). An equality assessment between $L[i]$ and c_{max} is then determined (Algorithm 6, line 7): if $L[i]$ is maximum, the function *testEquality* returns a number c_n where $n > 1$. Otherwise, *testEquality* returns c_0. We increment the argmax $B[i]$ with the output of each comparison (Algorithm 6, line 8). Once all pairwise comparisons are performed, B comprises the ciphered argmax of L. The argmax will be the index with the maximum value.

Algorithm 7 presents *testEquality* in more details. We build on the

Fermat little theorem, as in [225, 219]: if p is a prime, and if $x \not\equiv p$, then $x^{p-1} \equiv 1 \mod p$. The function takes as argument two ciphered integers c_x and c_y to test for equality, and a prime number p verifying $p > x$ and $p > y$. We compute $(c_x - c_y)^{p-1}$ to test the equality between x and y (Algorithm 7, l.2-5). We then compute $C(1)$-$(c_x - c_y)^{p-1}$ and multiply the result by an integer $c_n > 1$ (Algorithm 7, l.6). We take $c_n > 1$ to differentiate equality (c_n) from non equality (c_0). Indeed, each computation on ciphered numbers generates noise in FHE. Hence the variation between 1 and 0, as proposed in the original Fermat little theorem, is hard to detect after deciphering.

In our motivating example, the forwarded ciphered offers are $[\Pi_{O_{A1}}, \Pi_{O_{A2}}]$. The respective indexes of $\Pi_{O_{A1}}$ and $\Pi_{O_{A2}}$ are 0 and 1. We suppose n=10. The oracle computes the comparison for $[mean(\Pi_{O_{A1}}), mean(\Pi_{O_{A2}})]$ and $[mean(\Pi_{O_{A2}}),mean(\Pi_{O_{A1}})]$. The argmax of the comparisons is B=$[c_0,c_{10}]$.

Deciphering the argmax and retrieving the best offer After the argmax computation, the oracle asks the service provider authority to decipher the argmax (Figure 6.7, step 8). The service provider authority uses its FHE key to decrypt the argmax. It then forwards the deciphered argmax to the oracle (Figure 6.7, step 9). The oracle then transfers the argmax vector to summarize the pairwise comparisons as an array to the smart contract (Figure 6.7, step 10). The smart contract reverts the shuffling applied in step 1 on the received array (Figure 6.7, step 11). The smart contract sets the winning offer (Figure 6.7, step 12). If there is only one maximum, the winner is the service provider whose index is mapped to the array's maximum. If several offers are equal, the winner is the service provider that first submitted its proposal. Such information comes from the blockchain logs generated when submitting the ciphered offers (c.f., Figure 6.6, steps 8 and 14). The smart contract emits an event to notify service providers of the auction's output and sets the service provider's blockchain address with the winning bid issuer (Figure 6.7, step 13-14).

In our motivating example, the oracle sends the ciphered argmax to the carriers' authority for deciphering. The carrier authority deciphers the argmax and obtains L_{dec}=[10,0]. She forwards L_{dec} to the oracle, which sends L_{dec} to the smart contract. The smart contract reverts the shuffling to the original order on the deciphered argmax: it finds that A_2 is the winning offer. The smart contract sets the blockchain address of the service provider with the address of the winning bid issuer. It also sends a notification to A_1 to inform her that her bid has not been successful.

In summary, we have combined three ciphering algorithms to carry on a privacy-preserving comparison on multi-objective offers. Using FHE instead of partial homomorphic encryption enables the comparison of multi-objective offers. Meanwhile, the hybrid RSA/AES algorithm enables a confidential transfer of FHE-ciphered offers in the blockchain. The content of offers

is tamper-proof and cannot be read by other competitors as they do not hold the AES key. The only participant able to decipher the offers is the comparison oracle, which holds the RSA secret key. Moreover, the content of offers remains confidential, even for the comparison oracle, as it is ciphered with FHE and can only be deciphered by FHE key holders.

Our approach, as in [192, 103] preserves bids' privacy thanks to FHE technics, bidders' privacy thanks to blockchain pseudo-anonymity, and bidders' non-interactivity. Our approach interest comes from using FHE technics to carry on bid comparison in such an ecosystem: any calculation is theoretically accessible to compare offers without using trusted hardware having access to cleared data. Additionally, the system offers a distributed authority due to several entities managing bids comparison (the service providers' authority, the blockchain, and the ciphered computation oracle). Finally, it provides a public auditability as the following information is available on the ledger: IPFS hashes containing ciphered offers, bidders' participation, and the winning result blockchain address.

6.4 Privacy-preserving token payment

After the contractualization stage, the client must pay the service provider once the service is fulfilled. Nonetheless, participants may not be willing to reveal the value of the accepted bid to other competitors. This section proposes to leverage smart contracts and a bank to manage per-collaboration payment tokens. The tokens are backed with fiat money with a conversion rate that is kept secret between payment partners and the bank. Transactions are stored in the ledger, hence offering auditability. Nonetheless, the real payment value is not accessible on-chain, hence providing privacy.

In the following, we start by providing an overview of our approach (Section 6.4.1), we then present the four stages composing our approach in the following four sections. Section 6.4.2 presents the first step: the payment token smart contract initialization. Section 6.4.3 presents the second step: the payment tokens allocation to participants upon request. Section 6.4.4 describes the third step: a service payment using tokens. Finally, Section 6.4.5 presents the last step: the mechanism employed for collaboration settlement.

6.4.1 Overall approach

Figure 6.8 presents the four stages composing our approach. The first stage consists of initializing the single-use payment token smart contract. The value assigned to the payment token is set randomly and provided off-chain to payment participants during this stage. The second step consists of giving payment tokens to the payment sender willing to pay the payment receiver confidentially for a service. The third stage consists of the privacy-preserving service payment using payment tokens put into escrow in the smart contract.

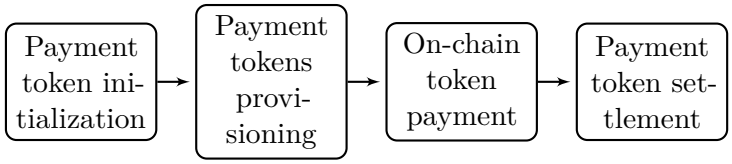


Figure 6.8: Privacy-preserving smart contract payments with a bank: main stages

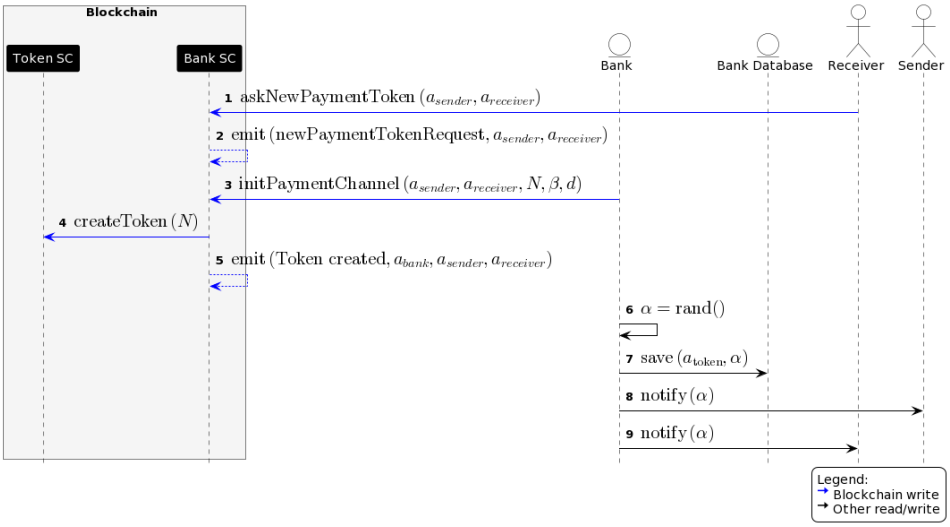


Figure 6.9: Sequence diagram of the payment token smart contract initialization (SC=Smart contract)

The last step consists of the payment token smart contract settlement once the collaboration between the payment sender and the receiver terminates.

6.4.2 Payment token smart contract initialization

Figure 6.9 shows the sequence diagram of the payment token smart contract initialization.

First, the payment receiver asks the smart contract manager, referred to as "Bank Contract" for a new collaboration token (Figure 6.9, step 1). The transaction contains the blockchain public key of both payment sender and receiver (respectively a_{sender} and $a_{receiver}$), which serves as a unique identifier for keeping track of the payment history.

Bank Contract emits an event stipulating that a new payment token should be issued between the payment sender and payment receiver (Figure 6.9, step 2). Participants who are listening to the smart contract (namely, the bank, the payment receiver, and the payment sender) will thus

be notified of this event.

Algorithm 8: Payment Channel h elements

```
1 Struct h contains
      // declare participants and token addresses
2     address a_sender;
3     address a_receiver;
4     address a_token;
      // declare payment channel parameters
5     uint N // token supply
6     uint β // claim ratio
      // declare payment
7     serviceStatus status // { INIT, DONE, CLAIM, PAYED }
8     uint t // target block
9     uint q // q
```

Algorithm 8 presents the structure of the payment channel h declared into the smart contract. h is defined as a struct comprising the following elements: (i) the service status (*INIT* when the channel is initialized, *DONE* when the service is done and waits to be paid, *CLAIM* if a claim occurs regarding the service delivery, and *PAYED* once payment has been fulfilled), (ii) the payment sender address a_{sender}, the payment receiver address $a_{receiver}$, and the token contract blockchain address a_{token}, (iii) the token supply N, i.e., the total number of tokens to generate, (iv) the claim ratio used to reallocate tokens put into escrow if a claim occurs β, (v) the target block t, that is, the delay given to the payment sender to trigger a claim using the smart contract, and (vi) the token price q, set upon a service request trigger, which corresponds to the price of a service paid using tokens.

Upon receiving the event asking for a payment channel creation, the bank initializes the new payment channel (Figure 6.9, step 3), deploys the payment token smart contract on-chain (Figure 6.9, step 4) and emits an event to confirm the creation of the token smart contract (Figure 6.9, step 5). Algorithm 9 presents the initPaymentChannel function which comprises steps 3-5 of Figure 6.9. The smart contract first verifies that the identity of the transaction sender corresponds to a_{bank} (line 2). It then creates a payment channel struct (line 3) and initializes each parameter of h (lines 4-12). Service status is set to *INIT* (line 8). The target block t is computed based on the claim time d forwarded to the bank contract (line 9). The smart contract triggers the token contract factory to generate a new token contract V_{token} (line 11): it does so by specifying the owner of the tokens (a_{bank}), as well as the token supply N. The address of the generated token contract is saved into h (line 12). The smart contract then adds h to the list of registered payment channels (line 13) and notifies with an event that the

Algorithm 9: On-chain initialization of a new payment channel

Data: H list of payment channels

1 **Function** `initPaymentChannel`$(a_{sender}, a_{receiver}, N, \beta, d)$**:**

    `// VERIFY SENDER IDENTITY`

2 require$(a_{sender} == a_{bank})$;

    `// GENERATE NEW PAYMENT TOKEN`

3 $new\ h$;

4 $h.a_{sender} \leftarrow a_{sender}$;

5 $h.a_{receiver} \leftarrow a_{receiver}$;

6 $h.N \leftarrow N$;

7 $h.\beta \leftarrow \beta$;

8 $h.status \leftarrow serviceStatus.INIT$;

9 $h.t \leftarrow block.number + d$;

10 $h.q \leftarrow 0$;

11 $Token\ V_{token} \leftarrow new\ Token(a_{bank}, N)$ `// generate token contract and fetch address`

12 $h.a_{token} \leftarrow V_{token}.a$ `// set token address`

13 $H.push(h)$ `// save channel`

14 emit event("token created", a_{bank}, a_{sender}, $a_{receiver}$);

15 **End Function**

token creation succeeded (line 14).

Afterward, the bank generates a random value α that is confidentially assigned to the payment token value (Figure 6.9, step 6). The payment token will be used to issue payment tokens to interested payment senders and manage payment token transactions; token smart contract payment tokens will be used as stable coins following the exchange rate 1T=α. Among token smart contract payment token transactions are (i) putting payment tokens into escrow when service is ongoing and (ii) allocating them to the interested participants when the service terminates.

The bank database references the random value α and the associated token smart contract payment token (Figure 6.9, step 7). Afterward, the bank notifies the involved participants of the token smart contract deployment and its associated random value (Figure 6.9, step 8-9).

As a side note, participants can initiate several payment tokens: tokens will be referenced based on the public address of the token smart contract a_{token}.

In our motivating example, we suppose $\alpha = 0.5€$. The bank deploys the token smart contract and generates a total supply of 50 payment tokens. As a side note, the total amount of tokens is set to provide enough tokens during the payment process. It is set independently by the bank in this use case, but it could be set based upon negotiation between the payment sender and the

payment receiver. The bank will save the token smart contract conversion rate coin value off-chain in its database. The participants involved in the confidential transaction, here the carrier, and the logistician, are noticed off-chain of the payment token value 1T=0.5€.

In this approach, payment token transactions are publicly accessible as they consist of public transactions stored on-chain. Nonetheless, their value remains confidential as it is linked to a random number by the bank. The tokens are backed with fiat money with a secret conversion rate: only the bank and participants will know the value of the tokens exchanged.

6.4.3 Request payment tokens

The next step consists of payment token issuance to the participant wishing to pay the payment receiver for a service.

First, the payment sender pays off-chain the bank in fiduciary money while specifying a_{token}, in exchange for token smart contract payment tokens.The bank generates the conversion between the provided amount in fiduciary money and the payment token value using the payment token value stored in the bank database referenced using the token payment on-chain address a_{token}. It obtains a quantity of m payment tokens. The bank asks the payment token to transfer these m payment tokens to the payment sender blockchain address.

In our motivating example, if we suppose the sum paid to the bank equals 10€, and $\alpha = 0.5$€, then the bank will ask the payment token smart contract to transfer 20 payment tokens to the blockchain address of the logistician.

6.4.4 Service payment

Let q be the token equivalent to the price of the service agreed upon by the payment receiver and the payment sender.

Figure 6.10 presents the sequence diagram of the service payment. Once the payment tokens are transferred (c.f. subsection 6.4.3), the payment sender launches the payment transaction request (Figure 6.10, step 1). It does so by specifying the number of payment tokens q to be escrowed by the smart contract. The bank smart contract will transfer these tokens to the payment receiver once the service terminates. The payment sender first authorizes the transfer of q from its token balance to the bank contract balance using the allocate function (Figure 6.10, step 2). The bank contract then verifies whether the payment sender has enough tokens (e.g., whether $q \preceq n$) using the allowance function (Figure 6.10, step 3). If there are not enough payment tokens, the transaction is reverted (Figure 6.10, step 4).

Else, the transaction proceeds: the bank contract asks the token smart contract to escrow q from the payment sender balance (Figure 6.10, step 5). Upon service completion, the payment receiver notifies the bank contract

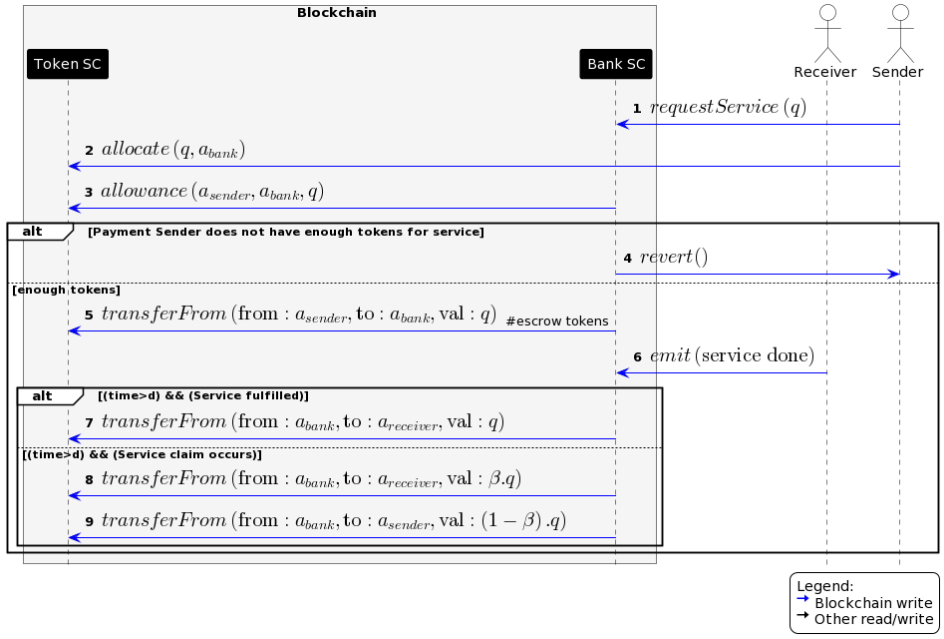


Figure 6.10: Service fulfillment

of the contract fulfillment (Figure 6.10, step 6). The payment sender can trigger a claim during the claim time. If the service is well fulfilled, the bank smart contract asks the token smart contract to transfer entirely q to the payment receiver balance (Figure 6.10, step 7). If a claim occurs during claim time, the bank contract proceeds to a partial transfer following the penalty factor $\beta < 1$ defined at the initialization of the payment channel. If a claim occurs, the penalty factor is applied to the number of payment tokens transferred: e.g., if β=0.4, then only 40 percent of the q payment tokens are transferred to the payment receiver (Figure 6.10, step 8). Remaining payment tokens e.g., (1-β)x = 0.6x are transferred back to the payment sender balance (Figure 6.10, step 9).

As a side note, several payment token transactions can be executed between the payment receiver and the payment sender in a privacy-preserving fashion. Each payment transaction provides to auditors the transaction timestamp (when), the number of tokens exchanged (what), and the sender and receiver pseudonymous addresses (who). By so doing, a compromise is reached between full confidentiality (e.g., by losing the track of transactions between payments), and total traceability which implies revealing who completed the transaction, what amount, and when it took place.

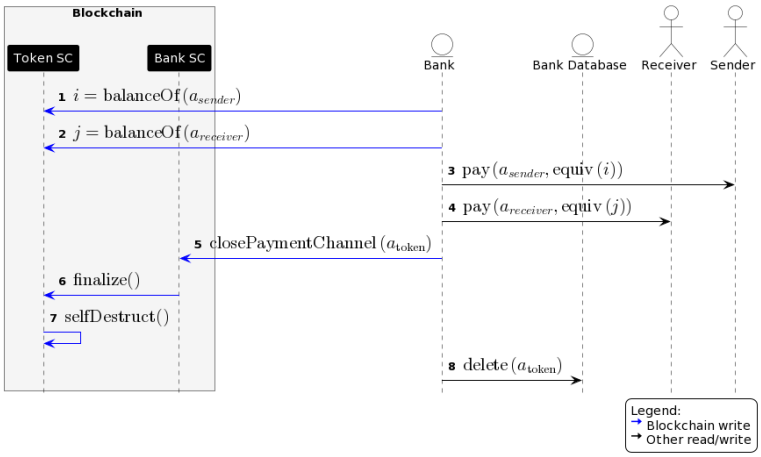


Figure 6.11: Settlement of the payment channel and token deactivation

6.4.5 Collaboration settlement and payment tokens deactivation

After all payment transactions have been carried on between the payment receiver and the payment sender, the bank smart contract can deactivate the payment tokens. Figure 6.11 presents the steps for settling the collaboration between the payment sender and the payment receiver.

The bank checks the payment token balance of the payment sender and payment receiver (Figure 6.11, step 1-2). Afterward, the bank converts each payment token balance into fiduciary money, following the conversion rate α, and pays back in an off-chain channel to the payment receiver and the payment sender (Figure 6.11, step 3-4). Then, the bank asks for the bank contract to close the payment channel (Figure 6.11, step 5). This function triggers the token smart contract (Figure 6.11, step 6) to ask it to self-destruct (Figure 6.11, step 7). The remaining tokens are set to null. Finally, the bank deletes the token value from its database (Figure 6.11, step 8).

6.5 Implementation and evaluation

We now investigate the feasibility and validity of the two privacy-preserving mechanisms dedicated to auctions and payments. In Section 6.5.1, we build a sealed-bid auctions mechanism and investigate (1) the ciphering file sizes, (2) the smart contract transaction costs, and (3) the ciphered comparison processing time required to enact our motivating example sealed-bids allocation. These metrics are important because they provide insights regarding the latency and scalability of the solutions. In Section 6.5.2, we build a privacy-preserving payment service leveraging random-value tokens

using Solidity smart contracts, and evaluate this mechanism using gas costs necessary for the payment stage of our motivating example, as a way to investigate the scalability of the solution.

6.5.1 Sealed-bid auctions

Implementation To demonstrate the approach's feasibility of the blockchain-based sealed-bid auctions, we build a C++ API that (i) gathers ciphered offers hashes, (ii) retrieves IPFS hashes, and (iii) compares ciphered offers using FHE[1]. We leverage the TFHE C++ library [41] that implements FHE methods and the Cryptopp C++ encryption library[2] to provide RSA and AES ciphering facilities.

The C++ API holds the following functions: FHE, RSA, and AES key generation, registration in IPFS, offers registration that populates a JSON file comprising offers holding the hash of the AES+FHE ciphered offer and the ciphered AES public key, and offers comparison using the content of the provided JSON file. We leverage the C++ API to implement the proposed mechanism in the Ethereum network. We deploy our smart contract on a local test network to assess the approach. We use Infura[3] as our API gateway to IPFS. By this means, we can store the ciphered offers and keys into IPFS and recover them using IPFS hashes. The comparison API is managed with a personal computer with an Intel i5 core CPU and 4GB of RAM.

This testing prototype comprises two parts. The first part demonstrates the offers registration part of the approach using Ethereum and Infura. It covers RSA, AES, and FHE key generation. To mimic real-life behavior, we initially generate FHE and RSA key pairs to mimic the key creation stage and provide private keys to the service providers' authority and oracle, respectively. We publish the RSA public key to the smart contract to simulate the initial oracle behavior and transfer the FHE key to carriers (Figure 6.12, step 1).

Our prototype covers bidders' local ciphering of offers: each bidder asks the RSA key to the smart contract (Figure 6.12, step 2.a.), generates the AES key pair locally, then ciphers the offer using FHE and AES private keys, and the AES public key using the RSA public key. Bidders then publish the offers hash into IPFS using Infura (Figure 6.12, step 2.b.). They then register their offer in the smart contract (Figure 6.12, step 2.c.). The smart contract takes care of the registration of competing offers. For bidders to interact with the smart contract, the smart contract implements the following

[1]Code is accessible here: `https://archive.softwareheritage.org/swh:1:dir:e08b491a6240075052af4c13709f95a83ef52ae6;origin=https://github.com/tiphainehenry/fhe_oracle;visit=swh:1:snp:05d0f85de663d57daabd74fe5fbc38a83cb2c953;anchor=swh:1:rev:8666576002ad5f375f9550798630a468681f29c6`.

[2]https://cryptopp.com/

[3]https://infura.io/docs/ipfs

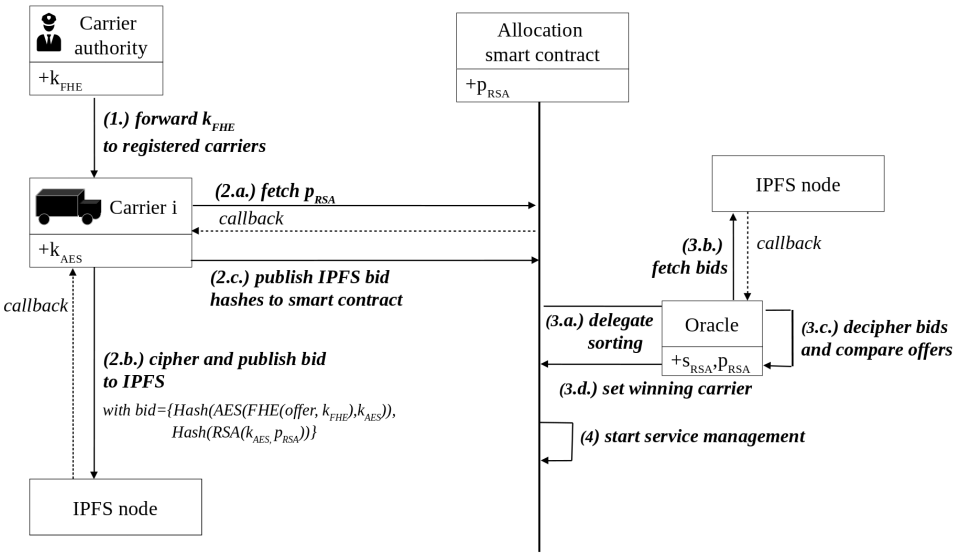


Figure 6.12: Protocol and key holders

functionalities: bid initialization, offer registration, and comparison launch.

The second part of the mechanism demonstrates the comparison part of the approach, using a toy array of ciphered offers stored in IPFS. It mimics the smart contract, the oracle, and the service providers' authority interactions on the reception of a JSON file populated with ciphered submissions. The API receives the IPFS hashes of ciphered offers and fetches the data (Figure 6.12, step 3.b.). Then, it proceeds to bids deciphering and comparison following the strategy presented in section 6.3.5 (Figure 6.12, step 3.c.). The winning bidder is allocated to the service request (Figure 6.12, step 3.d.), and the service management and later on, the settlement can proceed (Figure 6.12, step 4)

Evaluation We evaluate the protocol using three experiments that aim respectively to measure (1) the ciphering file sizes, (2) the smart contract transaction costs, and (3) the ciphered comparison processing time. These metrics are important because they provide insights regarding the latency and scalability of the solutions.

Table 6.2 gathers the size of the files generated during the execution of the motivating example scenario. The heaviest file is the FHE key file k_{FHE} (109MB). The size depends on the initial parameters used to generate the FHE key, which impacts the multiplicative depth chosen to carry on bootstrapping operations, and noise reduction [41, 23]. In our approach, the size of the FHE key is not a bottleneck, as it is forwarded by the service providers' authority to the bidders in private channels. The files to be stored on IPFS are (1) the RSA public key, (2) offers ciphered content, and (3)

Table 6.2: Size of files generated during the protocol (1 Mbit = 125 KB). Acronyms: IS= competitors' information system

Context	Participant	Storage	Data type	File size
Initialization	FHE key auth.	IS	k_{FHE}	109MB
	FHE key auth.	IS	FHE parameters	418B
	Tender Initiator	IPFS	p_{RSA}	160B
	Tender Initiator	Oracle	s_{RSA}	634B
Bid ciphering	Carrier	Carrier	k_{AES}	128B
	Carrier	IPFS	IV	16B
	Carrier	IPFS	$C_{AES}(C_{FHE}(offer))$	40KB

offers IV. The AES key file and IV file sizes are 128 and 16B. The ciphered offer file is heavier with 40KB. It is to note that the length of ciphertext is around a hundred times the plaintext size. Nonetheless, these file sizes range in the file sizes accepted to be stored in IPFS (100MB per request with Infura at the time of writing).

Table 6.3: Smart contract transaction costs.

Context	Participant	ETH Tx. fee
RSA Key IPFS hash storage	Tender Initiator	0.00265454ETH
Offer creation	Tender Initiator	0.00350124ETH
Offer registration IPFS hash storage	Competitor[1]	0.00417978ETH
Comparison request	Tender Initiator	0.152ETH

[a]Average transaction fees for competitors registration, where competitors offers are described in the motivating example (see Section 6.2)

Table 6.3 presents the transaction costs required to perform smart contract executions on the Ethereum blockchain. The transaction costs needed to store the RSA key on the blockchain are worth 0.00265454ETH (4.4€). They are worth 0.00350124ETH (5.8€) for the offer creation. The average cost paid by each competitor to register the IPFS hash of the ciphered offer is worth 0.00488606ETH (8.1€). Finally, the most expansive transaction consists of the comparison request performed by the tender initiator when the auction terminates. It is worth 0.152ETH (252.9€). This transaction fee does not depend on the number of offers to be compared, as offer content is sent via an event to the oracle, and event triggers do not require transaction fees. Instead, the price is fixed by the oracle, here Provable, and derives from the need to compensate the oracle for its computing power.

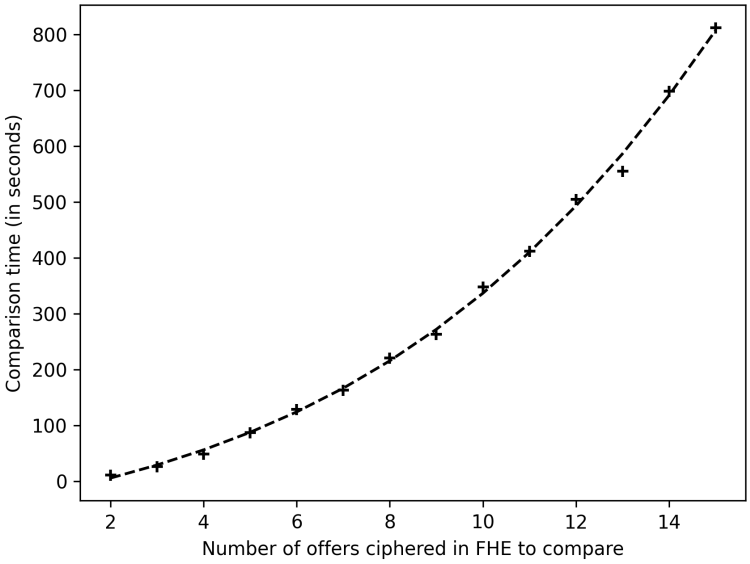


Figure 6.13: Comparison time according to the number of ciphered FHE offers submitted

Finally, we investigate the time taken to compare integers ciphered following the proposed encryption approach. To do so, we generate random numbers between 0 and 9. We cipher each using FHE and then launch the comparison algorithm. We measure the time needed to perform the pairwise FHE comparison. Figure 6.13 depicts the comparison times recorded for arrays of ciphered offers of increasing size. The time necessary to compare offers depends on the number of offers at stake. The dotted line in the figure represents a polynomial of degree 3 and illustrates the complexity in x^3 of our algorithm.

In summary, the size of files generated for the ciphering protocol implies adapting the distributed storage facility such as IPFS in case large batches of offers are compared. The gas fees required to leverage smart contracts as trustworthy auctioning third parties are only slightly impacted by the ciphering layers as only the hashes of the offers are stored on-chain. Instead, gas fees depend more importantly on the service price stated by the oracle service. Finally, the comparison time increases with the number of offers, and this limitation calls for further scalability studies.

6.5.2 Privacy-preserving payment

We now study the feasibility of the privacy-preserving payment mechanism. To do so, we build and evaluate a privacy-preserving payment service

leveraging random-value tokens[4]. We then evaluate the prototype using our motivating example use case.

Implementation Figure 6.14 illustrates the interaction between the actors and the contracts and between the smart contracts themselves. The main smart contract (the Bank contract) is deployed on the Ropsten network. The history of transactions related to the smart contract can be accessed using the smart contract address[5] using Etherscan (`https://ropsten.etherscan.io/address/`). The smart contract manages payment channels between payment senders and payment receivers. It is a trustworthy interface between payment senders, receivers, and the bank. The bank contract generates and interacts with a set of payment tokens defined according to the Ethereum ERC20 standard for creating tokens. Additionally, payment receivers and senders interact with the bank off-chain to manage fiduciary money and the random token value. These interactions are not displayed in Figure 6.14 for readability.

The Bank contract interface comprises five methods. The method *initPaymentChannel* initializes a new payment channel. Then, *getPayment* gets the payment sender address, the payment receiver address, the ERC20 contract address, and the token supply of a given payment channel. *serviceDone* allows the payment receiver to signal that he provided the service. With *serviceClaim*, the payment sender can raise a claim if she is in the claiming time window. Finally, *closePaymentChannel* will close a payment channel.

Generated tokens implement the ERC20 standard, which comprises the following methods: (i) *balanceOf* gets the balance of an Ethereum address, (ii) *transfer* allows the caller to transfer tokens; (iii) *approve* is used by the caller to allow an address to spend a certain amount of tokens; (iv) *transferFrom* is used by the caller to spend the tokens of a spender that allowed her to spend; and (v) *allowance* is used to get the allowance of an address to another one.

The deployment of the bank contract on the test network uses 2,353,462 gas (117.5€).

Evaluation We now evaluate the feasibility of our approach by enacting the $Pay(Customer \longrightarrow Florist)$ activity of our motivating example. In our testing protocol, we suppose that Customer has paid the bank to obtain a

[4]Code of the implemented prototype is available at `https://archive.softwareheritage.org/swh:1:dir:60577355b219ab188003bdaa624a31cf564f2b98;origin=https://github.com/tiphainehenry/random-value-token-payment;visit=swh:1:snp:dd7f3b87913aef5990d76d4c7fc3e11998cb0a8d;anchor=swh:1:rev:e2ad349e7d65c6a3dd735d48ce5a9e73ff9541c0`

[5]0x5F943a16Ba1Ea8E3E2FA87e8162181e3c5A6d2C0

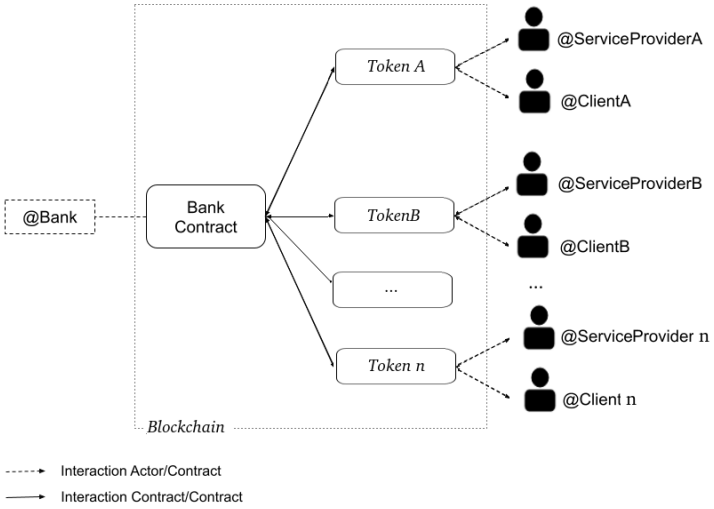


Figure 6.14: Interaction Contract/Contract & Actor/Contract

Table 6.4: Gas measurement for deploying 5 payment token smart contracts (SC).

	SC1	SC2	SC3	SC4	SC5
Gas	1,197,421	1,075,621	1,075,621	1,075,621	1,075,621
ETH	0.036	0.032	0.032	0.032	0.032
Cost in €	59.9	53.2	53.2	53.2	53.2

set of unique tokens assigned to a confidential random value. We evaluate the gas consumed by the method calls required to go through the payment process. We repeat the experiments with 100 tokens to assess whether the number of tokens impacts gas. In the following, Customer is referred to as the payment sender and Florist as the payment receiver.

We first investigate the payment token deployment costs. Table 6.4 presents the gas consumption according to the number of payment token smart contracts deployed. The deployment of the first ERC20 token uses 1,197,421 gas (59.8€). Afterward, for each new token creation (we deployed 100 ERC20 contract deployments), the gas used stays constant at 1,075,621. Hence, we only present the measurements for the first five smart contracts. The first deployment is more expensive than the others. One reason is that the bank contract must initialize the memory used for the first ERC20 contract deployment. The bank contract adds to the existing memory for the other token creations without initialization. As a side note, the initial bank contract deployment uses two times more gas than deploying a token

Table 6.5: Gas measurement for the Bank Contract Methods during payment and settlement stages.

	Payment					Settlement
	Transfer	Approve	Request	Done	Pay	Close
Gas	51,156	44,091	47,562	43,324	57,991	37,213
ETH	0.0015	0.0013	0.0014	0.0013	0.0017	0.0011
Cost in €	2.5	2.2	2.3	2.16	2.8	1.8

contract. This behavior can be explained in terms of code volume. Indeed, the bank contract has to embed the ERC20 contract to deploy a new ERC20 contract.

We now investigate the payment and settlement stages' transaction fees. Table 6.5 presents the gas consumption of the functions available in the Bank contract after creating one token. Experiments for 100 token creations show that the number of payment channels created does not impact the gas required to process or settle a payment. We detail below the gas consumption for these stages. Regarding the payment stage, the payment sender can claim a dispute if she is unhappy with the service provided. There are two scenarios. Without a claim, the payment receiver receives the entire token amount, consuming 57,991 gas. If the payment sender claims a dispute, tokens are distributed using a given ratio (i.e., 60% goes to the payment receiver and 40% goes to the payment sender). Claiming a dispute will consume 28,346 gas.

It is to note that the claiming ratio value does not impact the gas. The gas consumed to transfer tokens to the actors stays at 57,908 gas. Hence, the token redistribution will consume 86,254 gas which takes 28,263 more than without claim.

Regarding the settlement stage, closing a payment channel (destroying the ERC20 contract and deleting the entry stored in the smart contract) requires 37,213 gas, which is 0.00111639 ETH (1.8€). The smart contract deletion consumes less gas than creating a new one: 37,213 gas versus 1,075,621 gas.

In summary, the token deployment gas fee is constant: hence, the solution is scalable to collaboration-intensive payments. Moreover, claiming a dispute consumes additional gas, hence this behavior could be used as an additional incentive for participants to carry on the services as agreed upon in the contract. Finally, the payment collaboration settlement implies additional gas fees related to the token contract deactivation. The perspective of these additional gas fees could be less discouraging regarding adoption if participants use this protocol for several payments.

6.6 Conclusion

In this chapter, we combine the blockchain with an ecosystem comprising a computation oracle and a service provider authority managing encryption keys to carry on sealed-bid auctions on-chain. Smart contracts manage autonomously and transparently keys attribution, offers registration, and bids allocation. This distributed authority preserves bids' privacy and bidders' non-interactivity. The pseudo-anonymity of blockchain users also ensures bidders' privacy requirements. The comparison of offers takes place on FHE-ciphered data. FHE enables the oracle to compare offers without having access to the cleared data. To ensure data privacy end-to-end, we propose a mechanism leveraging two complementary encryption algorithms: RSA and AES. The hybrid RSA/AES encryption is used to transfer data in the blockchain without directly revealing the FHE-ciphered offers to other participants. Additionally, we ensure public auditability of the sealed-bid auction as the following information is available on the ledger: IPFS hashes containing ciphered offers, bidders' participation, and the winning bidder's blockchain address. We focused on sealed-bid auctions as in such auctions, bidders explicitly do not have access to bids from other competitors, and are requested to submit only one bid per auction. However, the approach could be studied in future work for other type of auctions such dutch auctions.

In this chapter, we also propose payment tokens assigned to a random value stored off-chain to ensure a privacy-preserving service collaboration payment. A bank contract manages collaboration settlement interactions by issuing an on-demand token smart contract assigned to a random value. Hence, payment receivers and payment senders can exchange tokens and settle their services on-chain while ensuring (1) tamper-proof records of payments for auditing purposes and (2) privacy of the fiduciary payment value. One payment token smart contract can be used for multiple payments between two actors, similar to a payment channel, but carried on the main chain. Only the bank, the payment receiver, and the payment sender know the exchange rate of tokens to fiduciary money. Banks do not need to manage accounting entries: all accounting entries are collected on-chain. The ledger keeps track in a tamper-proof fashion of the number of tokens exchanged and the blockchain pseudo-identity of the payment issuers and receivers, which can be used for auditing and claim resolution. An advantage of this approach is the easy setup as no encryption key management is necessary: only the bank manages token exchange rates. All transactions can take place using a classical token smart contract payment token.

Through these contributions, we hence answer the research questions (RQ3.2) and (RQ3.3), and validate the remaining objectives 5 and 6 presented in Section 1.4.1. The experiments finally prove the feasibility of these mechanisms, and thereby confirm the completion of these objectives.

The work presented in Section 6.3 has been submitted to the Journal

of Banking and Financial Technology [89], and the work presented in Section 6.4 has been published in the International Conference on Cooperative Information Systems [88].

Chapter 7

Summary, Discussion and Future Work

This chapter summarizes the research conducted in this thesis. We present how our contributions answer the research questions (RQ1-3). We discuss each contribution before describing future work stemming from this thesis research outputs.

7.1 Summary

The thesis investigates the potential for trustworthy, flexible, and privacy-preserving peer-to-peer business process management systems.

To address this challenge, we propose the three following research questions: (RQ1) How to leverage smart contracts as a trustworthy distributed tool for coordination and decision-making in cross-organizational processes? (RQ2) How to deploy and execute in a flexible fashion cross-organizational processes managed on-chain? (RQ3) How to ensure the privacy of sensitive data processed on-chain while preserving blockchain systems' integrity and verifiability properties?

To foster the use of smart contracts as a trustworthy distributed tool, we proposed in Chapter 4 to focus on the deployment and execution of blockchain-based DCR choreographies. This mechanism offers deployment trustworthiness while preserving the separation of concerns, two main challenges related to the property of transparency of blockchain systems. The first contribution is a mechanism for the hybrid on/off-chain deployment of DCR global choreographies. Participants build the global process incrementally from a public view stored in a smart contract during deployment. Each participant will compute off-chain its role projection comprising public events where she is involved, and private events are kept off-chain for privacy concerns. This way, private control flows remain in the participants' process engines, while blockchain systems ensure a tamper-proof

public view. The blockchain cannot access private events; aggregating all role projections will render the global process. The second contribution is a hybrid on/off-chain mechanism for executing cross-organizational choreographies. The roles complete their internal tasks off-chain in their local process execution engine. Meanwhile, a smart contract manages public interactions. When the smart contract receives an interaction request initiated from one of the roles (sender or receiver(s)), it executes the task and communicates its state back. The roles update their private states accordingly. Hence, we achieve a trustworthy separation of concerns preserving partners' private processes' privacy. Hence, through these contributions, we answer (RQ1).

We investigate in Chapter 5 the notion of flexibility through two prisms: control-flow changes and partnerships changes. To improve control-flow changes, we propose a protocol that extends the DCR choreography management presented in Chapter 4. This protocol allows (i) partners first to negotiate the change on-chain, (ii) then to dynamically update the choreography process instance managed by the smart contract with the new process change information, and (iii) finally propagate this information across partners' processes affected by the change. Regarding the flexibility of partnerships, we propose an algorithm leveraging smart contracts to filter and sort a set of available candidates based on their past blockchain history. Such matching, done autonomously, generates a digital agreement that can be linked to a BPMS to manage the settlement of the allocated services. We thus provide a transparent and reliable protocol computing a set of tamper-proof QoS ratings. In so doing, we contribute to the literature by addressing the lack of research regarding the fair (i.e., trustworthy and reliable) enactment of QoS allocations. Hence, through these research works, we address our second research question focusing on increasing the flexibility of cross-organizational processes managed on-chain. Moreover, we address the trust challenge through smart contracts enforcing an agreed-upon binding protocol and through the blockchain ledger offering a tamper-proof history of services. In both approaches, blockchain addresses service providers' and issuers' collusion risks by design, as no single entity controls the cross-organizational business process management platform. We also address the need for process automation through management through smart contract hiring, contractualization, and settlement stages.

Finally, in this manuscript, we focus on two cross-organizational activities prone to both privacy and auditability requirements: auctions and payments. Regarding auctions, we focused on sealed-bid auctions as in such auctions, bidders explicitly do not have access to bids from other competitors, and are requested to submit only one bid per auction. We propose a solution to leverage FHE in a blockchain context while preventing deciphering and collusion issues on the competitor's side: we do so by combining the hybrid RSA/AES ciphering mechanism with smart contracts and oracles to compare ciphered vectors. The comparison of offers takes place on FHE-ciphered

data. FHE enables the oracle to compare offers without having access to the cleared data. Smart contracts manage autonomously and transparently keys attribution, offers registration, and bids allocation. This distributed authority preserves bids' privacy and bidders' non-interactivity. The pseudo-anonymity of blockchain users also ensures bidders' privacy requirements. Finally, we ensure public auditability of the sealed-bid auction as the following information is available on the ledger: IPFS hashes containing ciphered offers, bidders' participation, and the winning bidder's blockchain address. The approach could be studied in future work for other type of auctions such dutch auctions.Regarding payments, we propose a solution that uses a bank and a per-collaboration payment token linked to a random value to address this research question. Parties can use per-collaboration tokens to proceed to multiple payments while preserving the values' privacy. Token payment can be programmed to verify conditions coded in a smart contract, put into escrow, and carry partial payment. Additionally, external peers can audit trust-worthily token transactions as they are stored on-chain. Hence, under the prism of these contributions, we address (RQ3).

7.2 Discussions

7.2.1 Discussion on the DCR-choreography deployment and execution blockchain-based system

The hybrid on/off-chain deployment and execution approach represents a first effort to separate the public and private views of a declarative choreography and proceed with its hybrid off/on-chain management. Results confirm the advantages of separating public from private events to ensure privacy while leveraging the blockchain technology as a decentralized execution infrastructure. Moreover, the local execution of private events leads to time and economic gains.

In this approach, the border case of a fully private process (i.e., the process does not hold public events) is considered. Then, no public projection is generated, and the process remains in the private information system of the partner. Multi-instance choreographies are also possible: for each new instance, a workflow instance is added to the smart contract.

Besides, experiments on graphs of alternative complexity (be it the number of participants or activities) should confirm preliminary results regarding latency gains induced by the hybrid on/offchain separation of concerns. A limitation of our approach concerns the public/private exchange of information. In our setting, the information published in the smart contract is public. Consortium or private blockchains, coupled with off-chain oracles to exchange sensitive information with the smart contract, or ciphering technics such as the ones presented in Section 6 could answer privacy concerns.

Furthermore, we rely on the truthfulness of participants to execute their

private projections, and we do not ensure the correct enforcement of private processes. This concern, inherent to choreographies, is part of ongoing research efforts.

For future work, it would be interesting to study the use of side channels [163] to manage on-chain process instances to save transaction costs and reduce task execution latency. Only two blockchain transactions would be of need: one to instantiate the process execution channel and one to settle it. It would also be interesting to integrate this work within the eSourcing framework, as public views could be associated to the external-level contractual collaboration processes identified in the framework, while role projections would correspond to extended in-house processes where additional process details are hidden.

7.2.2 Discussion on the DCR-choreography control-flow and partners change mechanism

Control-flow flexibility prototype

The control-flow flexibility approach holds several limitations. In this approach, we only consider the compatibility checks between public DCR processes of partners as a correctness criterion. We are working on proving the consistency checks between one partner's private and public DCR processes. Moreover, we focus on the current instance of the process. Nonetheless, it is also interesting to consider the change at the process model level and that after the change is validated, all future instances follow the change. Additionally, the change initiator specifies the endorsers, which could impede trust in the governance. An alternative design could be as follows: choreography participants could alternatively agree on a pre-specified list of endorsers before starting the process instance [125]. This way, the change negotiation and propagation agreement can be placed off-chain. Meanwhile, an on-chain transaction stating the agreement would be stored in a multi-signed document in IPFS (this might require using a different blockchain platform). However, even with multi-sig mechanisms, the risk of private key loss remains, and recovery schemes such as using secured wallets should be investigated [220]. Alternatively, change negotiation could be delegated to decentralized autonomous organisations, where semantic rules and negotiation strategies could be integrated as smart contracts for each organization. Finally, governance should also be considered when choosing the access control setup. Not every endorser should necessarily run their full node for public blockchains to preserve the consensus. For permissionned blockchains, governance should be well shared between change endorsers to avoid tampering or transaction misuse.

Partnership flexibility prototype

Several limitations appear in the dynamic QoS assignment mechanism proposed in Chapter 5. First, smart contracts use oracles to aggregate blockchain-available QoS data. In return, oracles communicate with the computation system via API calls. Issues related to API tampering may occur. The oracle could call independent computation APIs and compare the results to ensure results integrity. Additionally, a party wishing to claim a computation failure may use blockchain data to verify computations. At last, the QoS computation protocol should be considered modular. The protocol should be adapted to the binding context: Pareto optima could be used to find the best QoS instead of normalized means; an exponential moving average could be used to update the QoS to put a heavier weight on the latest services.

To be close to the field's reality, we designed the artifact by extracting the main allocation metrics used in the freight transportation procurement process. For the user tests, we gathered a population without experience with blockchain. Thus we can assume that their answers are close to carriers discovering the technology for the first time. Furthermore, the focus groups comprise logistics field experts: we can assume that we qualitatively assessed the potential of blockchain-based freight transportation procurement process mapping in the logistics context. Regarding internal validity and results correctness, we tried to influence the least possible user testing experiments by letting the users be free to use the application. The focus group panel was also broad enough to underline the most salient benefits and challenges of using blockchain-based freight transportation procurement process mappings. However, we acknowledge that the logistics researchers and the industry experts are an estimator of what happens in the logistics services' procurement process. Moreover, we use a public blockchain to demonstrate the feasibility of the solution, which may lead to an inadequate balance of powers. To ensure an adequate balance of powers between shippers and carriers, a future solution would require a permissioned blockchain dedicated to the mapping process, whose smart contracts are generated by a consortium of shippers and carriers.

Finally, the design science research approach is anchored in a freight transportation procurement process context, and thus findings are not generalizable.

7.2.3 Discussion on the privacy-preserving auction and payment mechanisms

Privacy preserving auctions

Experiments show the feasibility of the trustworthy FHE-based sealed-bid auctions mechanism, though limitations arise regarding collusion risks,

transaction costs, system scalability, and latency of comparison requests.

Collusion risks may arise in our system. First, using oracles implies a re-centralization and requires trust in the output. Triangulation methods from several oracles partially answer the risk of data tampering from a malicious node. Additionally, a collusion risk may exist between the computation oracle and the service providers' authority. Indeed, the oracle possesses FHE offers, and the service providers' authority owns the decryption key. Hence, removing interactions between competitors comes at the expense of a centralized authority responsible for key issuance and argmax deciphering. We aim to limit information leakage using offer permutation on the smart contract size to prevent linking offer content to the pseudonymous identities of participants.

The transaction costs required to launch a comparison request depend on the oracle chosen, and the use of Provable may be prohibitive to perform a ciphered comparison. Additionally, the protocol shall be tuned depending on the use case regarding metrics aggregation and FHE parameter tuning for noise addition. Such tuning can impact the FHE key and ciphered offers size, as well as the comparison time, and shall be taken into account.

The computation time follows the comparison algorithm polynomial complexity of x^3, induced by the ciphered maximum computation of arrays: the comparison time is directly proportional to the cube's number of offers to be compared. We implement the comparison asynchronously, using smart contract events, as oracle requests have a maximum callback time often exceeded with the FHE comparison. Hence, this approach does not apply to time-dependent applications, especially if many offers are at stake. Ongoing research in the field of FHE may help solve this latency issue [35].

Finally, scalability issues arise. The number of comparisons is limited by the random access memory of the machine. In our testing configuration, with a personal computer with an Intel i5 core CPU and 4GB of RAM, the comparison works with a maximum of 15 offers. Additionally, public IPFS storage can be limited if offer sizes are consequent, with multiple metrics. To circumvent this issue, one could use a private or permissioned IPFS channel to store offers in a decentralized fashion.

Privacy preserving payments

The main limitation of the random-value token payment approach could be some centralization induced by the use of the bank to manage token payment exchange rates, even if banks are often considered trusted entities. To defuse the risk of privacy leakage, several banks could be used to manage each collaboration token. Then, each bank would only have partial knowledge of the token value. Furthermore, scalability issues may arise as we issue a new payment token value each time a new collaboration requires a confidential payment. This issue can be mitigated by using a sidechain [188] and by

regrouping transactions. We should address both issues in future work.

An extension to this approach consists of implementing smart contract-based periodic negotiations to review and update the token exchange rate. Additionally, payment senders could have the possibility to choose between public or private pricing alternatives. If public pricing occurs, then a classic payment is processed. Else, if privacy is required, a token smart contract is initialized.

7.2.4 Summary

In summary, the limitations of this thesis can be listed as follows:

- **Formal proofs of correctness**: we demonstrate the feasibility of the solutions through implemented prototypes. Nonetheless, formal proofs are needed to ensure the correctness of the proposed DCR-to-Solidity translation algorithms and consistency checks between one partner's private and public DCR processes.

- **Private processes enforcement**: the correct enforcement of private processes is not ensured in this work. This concern, inherent to choreographies, is part of ongoing research efforts.

- **Transactions scalability**: in current prototypes, the number of comparisons to process, the size of DCR graphs to be deployed, or the number of new token issuances is limited.

- **Oracle use and latency**: experiments demonstrate that latency increases when oracles are included in the design of proposed mechanisms, e.g., when processing comparisons, and especially FHE-based comparisons. For now, proposed approaches leveraging oracles do not apply to time-dependent applications, especially if many offers are at stake.

- **Collusion risks**: Collusion risks may appear in our proposed systems. First, the use of oracles for computation implies a re-centralization and requires trust in the output (issue illustrated for QoS computation (with or without FHE)). Moreover, when dealing with complex key management schemes (e.g., in Chapter 6), a balance between privacy and recentralization risks is at stake.

- **Evaluation scope**: Finally, the proposed mechanisms have been tested on a logistic use case, and this study should be extended to other use cases, with more complex business processes. Plus, we use a public blockchain to demonstrate the feasibility of the solution, which may lead to an inadequate balance of powers. To ensure an adequate balance of powers between end-users such as customers and service providers, a

future solution would require a permissioned blockchain, whose smart contracts are generated by a consortium of partners' representatives.

7.3 Future work

At last but not least, we list hereinafter the main technical and non-technical challenges related to this thesis, calling for future work.

Technical challenges

We identify four main technical challenges: scalability of the blockchain architecture, blockchain interoperability, security of the information system, and the broadening of privacy-preserving technics in coopetition contexts.

Blockchain aims at solving disputes securely, and they do not address efficiency requirements as, e.g., centralized cloud computing does. *Scalability* of blockchain architectures is one of the main technical challenges to make the system more viable and usable by the general public, e.g., to be competitive for real-world applications [133]. Hence, increasing transaction throughput while preserving enough decentralization for security purposes should be studied [31]. Various strategies have been proposed to increase the number of transactions per second. Some propose to reduce the block validation time (e.g., decreasing from 10 minutes in Bitcoin to 2,5 minutes in Litecoin), while others propose to increase the block size (e.g., increasing from 1 MB in Bitcoin to 8 MB in Bitcoin Cash). Another strategy consists of gathering off-chain transactions in a file (e.g., school diplomas [40, 166]), in a sidechain (e.g., Blockstream and Lightening Network on Bitcoin and Slock.it on Ethereum), or in a shard chain (Ethereum 2.0) and recording a unique on-chain transaction on the main chain. Some promising solutions for interoperability and scalability[1] should be further analyzed [17].

Additionally, the challenge of blockchain *interoperability* is to be addressed in different configurations to ease cross-organizational collaborations: companies could deploy permissioned blockchains for internal processes but wish to connect one of the internal processes with a cross-organizational one running on a public blockchain [84]. Several future works concern the interoperability of homogeneous and heterogeneous blockchain systems, and

[1]Examples are Aion, Chainlink, Cosmos (Tendermint), Overledger, Plasma, or Polkadot.

the interoperability of assets and tokens [84, 185, 102][2].

A field of research with particular focus remains on the *dependability and security* justification of blockchain-based business process management systems [171]. Future work in this thesis hence calls for formal analysis and model checking of the translation rules leading to smart contracts, as well as a formal analysis of security and soundness of the change propagation rules proposed in Chapter 5. Robust testing with real-world case studies (e.g., DCR graphs of different sizes) should confirm the interest in the approach illustrated in this thesis. Furthermore, translation work should be extended to other business model languages such as DMN [100] and CMMN [217] smart contract translators. Finally, a special focus on the legal aspects of blockchain-based business process management systems should also be investigated by leveraging work exposed in [60].

Transparency and immutability are not always relevant for business executions. *Privacy* of internal processes or competitive information should be ensured in blockchain-based business process management systems. To add a privacy layer to blockchain transactions, we proposed to leverage cryptography protocols such as FHE in the context of sealed-bid auctions. In future work, the FHE-based solution should be extended to more business-sensitive activities, and improvements regarding computation times should be investigated. Additionally, solutions should be analyzed for the privacy-preservation of files that must transit from one process activity to another. Finally, a standardized framework for using trusted execution environments (TEEs) in business process management would be beneficial for theory abstraction.

We also identify several non-blockchain challenges regarding adopting blockchain-based information systems, to be investigated in future work.

7.3.1 Contextual challenges

Non-technical challenges should also be considered for adopting the technology in a cross-organizational context.

The first challenge concerns the *security of the data entries* as one cannot guarantee good input data, for example, in traceability use cases. Encrypted RFID tags can identify wrong objects; hence, hardware oracles can input false data into the blockchain. This issue is often referred to as the grapes problem: how do we ensure the merchandise, and not only the package, is to

[2](i) interoperability of homogeneous blockchain systems, e.g., between a first Hyperledger Fabric blockchain managed by a first entity and a second Hyperledger Fabric blockchain managed by a second entity; (ii) interoperability of heterogeneous blockchain systems, e.g., between Ethereum blockchain and Hyperledger Besu blockchain; (iii) interoperability of cryptocurrencies and assets/tokens, e.g., between ERC20 token (Ethereum) and QRC20 token (Qtum); and (iv) interoperability of smart contracts deployed in heterogeneous systems; e.g., between Ethereum smart contract and another blockchain smart contract.

be trusted?

The second challenge concerns the *barriers to adoption*: blockchain network security increases with the number of users. Bad UI/UX of decentralized applications or unfamiliarity or a lack of education can impede the adoption of blockchain-based information systems. Hence, a comparative study on the appropriation of different business process languages in the context of a blockchain consortium should be carried out, as well as design science research led by corporations for each new use case.

Finally, the issue of *governance*, both social and technical, is at stake. Social governance refers to the number of entities managing the blockchain system (e.g., only Ripple Labs for Ripple [177], only IBM for Hyperledger Fabric [53]), the percentage and nationality of nodes, and, in the case of proof of work, of mining pools. Such social governance raises concerns regarding the decentralization of property and the independence of blockchain systems [81, 106]. Technical governance refers to how the community of developers agrees or disagrees on planned upgrades and reacts to unplanned upgrades, sometimes leading to the split of blockchains (e.g., Ethereum Classic vs. Ethereum after a disagreement on TheDao attack, or Bitcoin vs. Bitcoin Cash after a disagreement on the block size). Such reaction impacts the communities' strengths, and the trust in the system [16]. Hence, social and technical governance should be studied in real-world settings to foster the adoption of blockchain-based information systems.

Bibliography

[1] Dero.io white paper, https://github.com/deroproject/documentation/blob/master/WhitePaper.md

[2] van der Aalst, W.M., Weske, M.: The p2p approach to interorganizational workflows. In: CAISE. pp. 140–156. Springer (2001)

[3] Ahmed, S., Rahman, M.S., Rahaman, M.S., et al.: A blockchain-based architecture for integrated smart parking systems. In: 2019 IEEE international conference on pervasive computing and communications workshops (PerCom workshops). pp. 177–182. IEEE (2019)

[4] Al-Breiki, H., Rehman, M.H.U., Salah, K., Svetinovic, D.: Trustworthy blockchain oracles: review, comparison, and open research challenges. IEEE Access (Volume: 8) (2020)

[5] Alacam, S., Sencer, A.: Using blockchain technology to foster collaboration among shippers and carriers in the trucking industry: A design science research approach. Logistics **5**(2) (2021). https://doi.org/10.3390/logistics5020037, https://www.mdpi.com/2305-6290/5/2/37

[6] Alpern, B., Schneider, F.B.: Recognizing safety and liveness. Distributed computing **2**(3), 117–126 (1987)

[7] Alvarez, R., Nojoumian, M.: Comprehensive survey on privacy-preserving protocols for sealed-bid auctions. Computers & Security **88**, 101502 (2020)

[8] Andrews, K., Steinau, S., Reichert, M.: Dynamically switching execution context in data-centric bpm approaches. In: Enterprise, Business-Process and Information Systems Modeling. Springer (2020)

[9] Androulaki, E., Camenisch, J., Caro, A.D., Dubovitskaya, M., Elkhiyaoui, K., Tackmann, B.: Privacy-preserving auditable token payments in a permissioned blockchain system. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies. pp. 255–267 (2020)

[10] of Antwerp Newsroom, P.: Smart port with blockchain (2019), https://www.portofantwerp.com/en/news/smart-port-blockchain, Accessed on 2019-11-29

[11] Astigarraga, T., Chen, X., Chen, Y., Gu, J., Hull, R., Jiao, L., Li, Y., Novotny, P.: Empowering business-level blockchain users with a rules framework for smart contracts. In: International Conference on Service-Oriented Computing. pp. 111–128. Springer (2018)

[12] Bach, L., Mihaljevic, B., Zagar, M.: Comparative analysis of blockchain consensus algorithms. In: MIPRO. pp. 1545–1550. IEEE (2018)

[13] Bagozi, A., Bianchini, D., Antonellis, V.D., Garda, M., Melchiori, M.: A three-layered approach for designing smart contracts in collaborative processes. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 440–457. Springer (2019)

[14] Baranwal, P.R.: Blockchain based full privacy preserving public procurement. In: International Conference on Blockchain. pp. 3–17. Springer (2020)

[15] Bazarhanova, A., Magnusson, J., Lindman et al, J.: Blockchain-based electronic identification: cross-country comparison of six design choices. In: Proceedings of the 27th European Conference on Information Systems (ECIS), Stockholm and Uppsala, Sweden (2019)

[16] Beck, R., Weber, S., Gregory, R.W.: Theory-generating design science research. Information Systems Frontiers (2013)

[17] Belchior, R., Vasconcelos, A., Guerreiro, S., Correia, M.: A survey on blockchain interoperability: Past, present, and future trends. ACM Computing Surveys (CSUR) **54**(8), 1–41 (2021)

[18] Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: 23rd {USENIX} Security Symposium ({USENIX} Security 14). pp. 781–796 (2014)

[19] Benet, J.: Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561 (2014)

[20] Bengtsson, M., Eriksson, J., Wincent, J.: Coopetition: new ideas for a new paradigm. Coopetition: Winning strategies for the 21st century pp. 19–39 (2010)

[21] Bermbach, D., Maghsudi, S., Hasenburg, J., Pfandzelter, T.: Towards auction-based function placement in serverless fog platforms. In: 2020 IEEE International Conference on Fog Computing (ICFC). pp. 25–31. IEEE (2020)

[22] Blass, E.O., Kerschbaum, F.: Strain: A secure auction for blockchains. In: European Symposium on Research in Computer Security. pp. 87–110. Springer (2018)

[23] Bonnoron, G.: A journey towards practical fully homomorphic encryption. Ph.D. thesis, Ecole nationale supérieure Mines-Télécom Atlantique (2018)

[24] Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: NDSS. vol. 4324, p. 4325 (2015)

[25] Bourse, F., Sanders, O., Traoré, J.: Improved secure integer comparison via homomorphic encryption. In: Cryptographers' Track at the RSA Conference. pp. 391–416. Springer (2020)

[26] Boutin, C.: Nist selects winner of secure hash algorithm (sha-3) competition. Press release., October **2** (2012)

[27] Brahem, A., Henry, T., Bhiri, S., Devogele, T., Laga, N., Messai, N., Sam, Y., Gaaloul, W., Benatallah, B.: A trustworthy decentralized change propagation mechanism for declarative choreographies. In: International Conference on Business Process Management. pp. 418–435. Springer (2022)

[28] Brahem, A., Messai, N., Sam, Y., Bhiri, S., Devogele, T., Gaaloul, W.: Blockchain's fame reaches the execution of personalized touristic itineraries. In: 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). pp. 186–191. IEEE (2019)

[29] Buchanan, B., Naqvi, N.: Building the future of eu: Moving forward with international collaboration on blockchain. The Journal of The British Blockchain Association **1**(1), 3579 (2018)

[30] Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: Towards privacy in a smart contract world. In: International Conference on Financial Cryptography and Data Security. pp. 423–443. Springer (2020)

[31] Buterin, V., et al.: A next-generation smart contract and decentralized application platform. white paper **3**(37) (2014)

[32] Caldarelli, G.: Understanding the blockchain oracle problem: A call for action. Information **11**(11), 509 (2020)

[33] Casilli, A.A., Posada, J.: The Platformization of Labor and Society, pp. 293–306. Oxford University Press (2019). https://doi.org/10.1093/oso/9780198843498.003.0018

[34] Cecchetti, E., Zhang, F., Ji, Y., Kosba, A., Juels, A., Shi, E.: Solidus: Confidential distributed ledger transactions via pvorm. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 701–717 (2017)

[35] Chatterjee, A., Sengupta, I.: Sorting of fully homomorphic encrypted cloud data: Can partitioning be effective? IEEE Transactions on Services Computing **13**(3), 545–558 (2017)

[36] Chatzigiannis, P., Baldimtsi, F.: Miniledger: compact-sized anonymous and auditable distributed payments. In: European Symposium on Research in Computer Security. pp. 407–429. Springer (2021)

[37] Chebbi, I., Dustdar, S., Tata, S.: The view-based approach to dynamic inter-organizational workflow cooperation. Data & Knowledge Engineering (2006)

[38] Chen, H., Han, K.: Homomorphic lower digits removal and improved fhe bootstrapping. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 315–337. Springer (2018)

[39] Chen, Y., Ma, X., Tang, C., Au, M.H.: Pgc: decentralized confidential payment system with auditability. In: European Symposium on Research in Computer Security. pp. 591–610. Springer (2020)

[40] Cheng, J.C., Lee, N.Y., Chi, C., Chen, Y.H.: Blockchain and smart contract for digital certificate. In: 2018 IEEE international conference on applied system invention (ICASI). pp. 1046–1051. IEEE (2018)

[41] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)

[42] Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016). https://doi.org/10.1109/ACCESS.2016.2566339

[43] Christin, N.: Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In: Proceedings of the 22nd international conference on World Wide Web. pp. 213–224 (2013)

[44] CoinDesk: The world's largest shipping firm now tracks cargo on blockchain (2017), `https://www.coindesk.com/worlds-largest-shipping-company-tracking-cargo-blockchain`, Accessed on 2020-01-28

[45] Daemen, J., Rijmen, V.: Reijndael: The advanced encryption standard. Dr. Dobb's Journal: Software Tools for the Professional Programmer **26**(3), 137–139 (2001)

[46] Dasgupta, D., Shrein, J.M., Gupta, K.D.: A survey of blockchain from security perspective. Journal of Banking and Financial Technology **3**(1), 1–17 (2019)

[47] Debois, S., Hildebrandt, T.: The DCR Workbench: Declarative Choreographies for Collaborative Processes, pp. 99–124. River Publishers (2017)

[48] Debois, S., Hildebrandt, T., Slaats, T.: Safety, liveness and run-time refinement for modular process-aware information systems with dynamic sub processes. In: International Symposium on Formal Methods. pp. 143–160. Springer (2015)

[49] Debois, S., Hildebrandt, T.T., Slaats, T.: Replication, refinement & reachability: complexity in dynamic condition-response graphs. Acta Informatica **55**(6), 489–520 (2018)

[50] Decker, G., Weske, M.: Behavioral consistency for b2b process integration. In: CAISE. pp. 81–95. Springer (2007)

[51] Desai, H., Kantarcioglu, M., Kagal, L.: A hybrid blockchain architecture for privacy-enabled and accountable auctions. In: 2019 IEEE International Conference on Blockchain (Blockchain). pp. 34–43. IEEE (2019)

[52] Deval, V., Norta, A., Dai, P., Mahi, N., Earls, J.: Decentralized governance for smart contract platform enabling mobile lite wallets using a proof-of-stake consensus algorithm. In: Blockchain Technology and Innovations in Business Processes, pp. 67–93. Springer (2021)

[53] Dhillon, V., Metcalf, D., Hooper, M.: The hyperledger project. In: Blockchain enabled applications, pp. 139–149. Springer (2017)

[54] Di Ciccio, C., Cecconi, A., Dumas, M., García-Bañuelos, L., López-Pintado, O., Lu, Q., Mendling, J., Ponomarev, A., Binh Tran, A., Weber, I.: Blockchain support for collaborative business processes. Informatik Spektrum **42**(3), 182–190 (2019)

[55] Divya, M., Biradar, N.B.: Iota-next generation block chain. International Journal of Engineering and Computer Science **7**, 23823–23826 (2018)

[56] Duffield, E., Diaz, D.: Dash: A privacycentric cryptocurrency (2015)

[57] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Introduction to business process management. In: Fundamentals of business process management, pp. 1–33. Springer (2018)

[58] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013)

[59] Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.: Fundamentals of business process management. Springer Berlin Heidelberg (2018)

[60] Dwivedi, V., Norta, A.: Auto-generation of smart contracts from a domain-specific xml-based language. In: Intelligent Data Engineering and Analytics, pp. 549–564. Springer (2022)

[61] Enkhtaivan, B., Takenouchi, T., Sako, K.: A fair anonymous auction scheme utilizing trusted hardware and blockchain. In: 2019 17th International Conference on Privacy, Security and Trust (PST). pp. 1–5. IEEE (2019)

[62] Eshuis, R., Norta, A., Roulaux, R.: Evolving process views. Information and Software Technology **80**, 20–35 (2016)

[63] Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: The issue of understandability. In: Enterprise, Business-Process and Information Systems Modeling, pp. 353–366. Springer (2009)

[64] Falazi, G., Hahn, M., Breitenbücher, U., Leymann, F., Yussupov, V.: Process-based composition of permissioned and permissionless blockchain smart contracts. In: 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC). pp. 77–87. IEEE (2019)

[65] Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: A new design for anonymous cryptocurrencies. In: International conference on the theory and application of cryptology and information security. pp. 649–678. Springer (2019)

[66] Fdhila, W., Indiono, C., Rinderle-Ma, S., Reichert, M.: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information systems **49**, 1–24 (2015)

[67] Fdhila, W., Indiono, C., Rinderle-Ma, S., Vetschera, R.: Multi-criteria decision analysis for change negotiation in process collaborations. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC). pp. 175–183. IEEE (2017)

[68] Food, C.A.F.: la blockchain alimentaire (2020), https://actforfood.carrefour.fr/nos-actions/la-blockchain-alimentaire, Accessed on 2020-02-03

[69] Frantz, C.K., Nowostawski, M.: From institutions to code: Towards automated generation of smart contracts. In: 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W). pp. 210–215 (2016). https://doi.org/10.1109/FAS-W.2016.53

[70] Gaaloul, W.: La Découverte de WorkflowTransactionnel pour la Fiabilisation desExécutions. (Mining transaction workflow for execution reliability). Ph.D. thesis, Henri Poincaré University, Nancy, France (2007), https://tel.archives-ouvertes.fr/tel-00124083

[71] Galal, H.S., Youssef, A.M.: Verifiable sealed-bid auction on the ethereum blockchain. In: International Conference on Financial Cryptography and Data Security. pp. 265–278. Springer (2018)

[72] Galal, H.S., Youssef, A.M.: Trustee: full privacy preserving vickrey auction on top of ethereum. In: International Conference on Financial Cryptography and Data Security. pp. 190–207. Springer (2019)

[73] García-Bañuelos, L., Ponomarev, A., Dumas, M., Weber, I.: Optimized execution of business processes on blockchain. In: Business Process Management. pp. 130–146. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5-8, https://link.springer.com/chapter/10.1007/978-3-319-65000-5-8

[74] Garman, C., Green, M., Miers, I.: Accountable privacy for decentralized anonymous payments. In: International Conference on Financial Cryptography and Data Security. pp. 81–98. Springer (2016)

[75] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)

[76] Goedertier, S., Vanthienen, J., Caron, F.: Declarative business process modelling: principles and modelling languages. Enterprise Information Systems **9**(2), 161–185 (2015)

[77] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on computing **18**(1), 186–208 (1989)

[78] Goodrich, M.T., Mitzenmacher, M.: Privacy-preserving access of outsourced data via oblivious ram simulation. In: International

Colloquium on Automata, Languages, and Programming. pp. 576–587. Springer (2011)

[79] Green, M., Miers, I.: Bolt: Anonymous payment channels for decentralized currencies. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 473–489 (2017)

[80] Gürcan, Ö., Agenis-Nevers, M., Batany, Y.M., Elmtiri, M., Le Fevre, F., Tucci-Piergiovanni, S.: An industrial prototype of trusted energy performance contracts using blockchain technologies. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp. 1336–1343. IEEE (2018)

[81] Hacker, P.: Corporate governance for complex cryptocurrencies? a framework for stability and decision making in blockchain-based organizations. Oxford University Press (2019)

[82] Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the provop approach. J SOFTW MAINT EVOL-R **22**(6-7), 519–546 (2010)

[83] Hara, K., Adams, A., Milland, K., Savage, S., Callison-Burch, C., Bigham, J.P.: A data-driven analysis of workers' earnings on amazon mechanical turk. In: Proceedings of the 2018 CHI conference on human factors in computing systems. pp. 1–14 (2018)

[84] Hardjono, T., Lipton, A., Pentland, A.: Toward an interoperability architecture for blockchain autonomous systems. IEEE Transactions on Engineering Management **67**(4), 1298–1309 (2019)

[85] Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In: Network and Distributed System Security Symposium (2017)

[86] Henry, T., Beck, R., Laga, N., Gaaloul, W., Pan, S.: Decentralized procurement mechanisms for efficient logistics services mapping-a design science research approach. In: Proceedings of the Annual Hawaii International Conference on System Sciences. IEEE Computer Society Press (2022)

[87] Henry, T., Brahem, A., Laga, N., Hatin, J., Gaaloul, W., Benatallah, B.: Trustworthy cross-organizational collaborations with hybrid on/off-chain declarative choreographies. In: International Conference on Service-Oriented Computing. pp. 81–96. Springer (2021)

[88] Henry, T., Hatin, J., Kazmierczak, L., Laga, N., Gaaloul, W., Bertin, E.: Random-value payment tokens for on-chain privacy-preserving payments. Cooperative Information Systems pp. 223–241 (2022)

[89] Henry, T., Hatin, J., Laga, N., Gaaloul, W.: Towards trustworthy and privacy-preserving decentralized auctions. Preprint (2022)

[90] Henry, T., Laga, N., Hatin, J., Beck, R., Gaaloul, W.: Hire me fairly: towards dynamic resource-binding with smart contracts. In: 2021 IEEE International Conference on Services Computing (SCC). pp. 407–412. IEEE (2021)

[91] Henry, T., Laga, N., Hatin, J., Gaaloul, W., Boughzala, I.: Cross-collaboration processes based on blockchain and iot: a survey. In: HICSS (2021)

[92] Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. Electronic Proceedings in Theoretical Computer Science **69**, 59–73 (2011). https://doi.org/10.4204/EPTCS.69.5, `http://arxiv.org/abs/1110.4161`

[93] Hildebrandt, T.T., Slaats, T., López, H.A., Debois, S., Carbone, M.: Declarative choreographies and liveness. In: International Conference on Formal Techniques for Distributed Objects, Components, and Systems. pp. 129–147. Springer (2019)

[94] Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification. GitHub: San Francisco, CA, USA p. 1 (2016)

[95] Huang, H., Lin, J., Zheng, B., Zheng, Z., Bian, J.: When blockchain meets distributed file systems: An overview, challenges, and open issues. IEEE Access **8**, 50574–50586 (2020)

[96] Hull, R., Batra, V.S., Chen, Y.M., Deutsch, A., Heath III, F.F.T., Vianu, V.: Towards a shared ledger business collaboration language based on data-aware processes. In: International conference on service-oriented computing. pp. 18–36. Springer (2016)

[97] IBM: Food trust (2020), `https://www.ibm.com/blockchain/solutions/food-trust`, Accessed on 2020-02-13

[98] Ihde, S., Pufahl, L., Lin, M.B., Goel, A., Weske, M.: Optimized resource allocations in business process models. In: International Conference on Business Process Management. pp. 55–71. Springer (2019)

[99] Indiono, C., Rinderle-Ma, S.: Dynamic change propagation for process choreography instances. In: OTM Conferences. pp. 334–352. Springer (2017)

[100] Janssens, L., Bazhenova, E., De Smedt, J., Vanthienen, J., Denecker, M.: Consistent integration of decision (dmn) and process (bpmn) models. In: CAiSE forum. vol. 1612, pp. 121–128 (2016)

[101] Jia, Y., Sun, S., Zhang, Y., Zhang, Q., Ding, N., Liu, Z., Liu, J., Gu, D.: Pbt: A new privacy-preserving payment protocol for blockchain transactions. IEEE Transactions on Dependable and Secure Computing (2020)

[102] Johnson, S., Robinson, P., Brainard, J.: Sidechains and interoperability. arXiv preprint arXiv:1903.04077 (2019)

[103] Keizer, N.V., Ascigil, O., Psaras, I., Pavlou, G.: Flock: Fast, lightweight, and scalable allocation for decentralized services on blockchain. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–9. IEEE (2021)

[104] Kerry, C.F., Gallagher, P.D.: Digital signature standard (dss). FIPS PUB pp. 186–4 (2013)

[105] Khan, S.N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., Bani-Hani, A.: Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-peer Networking and Applications **14**(5), 2901–2925 (2021)

[106] Khoshavi, N., Francois, W., Sargolzaei, A., Chintakunta, H.: A survey on blockchain security. In: 2019 SoutheastCon. pp. 1–8. IEEE (2019)

[107] Klai, K., Tata, S., Desel, J.: Symbolic abstraction and deadlock-freeness verification of inter-enterprise processes. In: BPM (2009)

[108] Köpke, J., Franceschetti, M., Eder, J.: Balancing privity and enforceability of bpm-based smart contracts on blockchains. In: BPM (2019)

[109] Krejci, S., Sigwart, M., Schulte, S.: Blockchain-and ipfs-based data distribution for the internet of things. In: European conference on service-oriented and cloud computing. pp. 177–191. Springer (2020)

[110] Król, M., Sonnino, A., Tasiopoulos, A., Psaras, I., Rivière, E.: Pastrami: privacy-preserving, auditable, scalable & trustworthy auctions for multiple items. In: Proceedings of the 21st International Middleware Conference. pp. 296–310 (2020)

[111] Kurz, M., Schmidt, W., Fleischmann, A., Lederer, M.: Leveraging cmmn for acm: examining the applicability of a new omg standard for adaptive case management. In: Proceedings of the 7th international conference on subject-oriented business process management. pp. 1–9 (2015)

[112] Ladleif, J., Weske, M., Weber, I.: Modeling and enforcing blockchain-based choreographies. In: International Conference on Business Process Management. pp. 69–85. Springer (2019)

[113] Lafkihi, M., Pan, S., Ballot, E.: Freight transportation service procurement: A literature review and future research opportunities in omnichannel e-commerce. TRANSPORT RES E-LOG (2019)

[114] Laue, R., Kirchner, K.: Patterns for discussing and modelling variability in business processes. In: EuroPLoP. pp. 1–10 (2018)

[115] Leiding, B.: The M2X Economy-Concepts for Business Interactions, Transactions and Collaborations Among Autonomous Smart Devices. Ph.D. thesis, Georg-August-Universität Göttingen (2019)

[116] Li, C., Palanisamy, B.: Decentralized privacy-preserving timed execution in blockchain-based smart contract platforms. In: HiPC. pp. 265–274. IEEE (2018)

[117] Li, Z., Kang, J., Yu, R., Ye, D., Deng, Q., Zhang, Y.: Consortium blockchain for secure energy trading in industrial internet of things. IEEE TII (2017)

[118] Lin, H.Y., Tzeng, W.G.: An efficient solution to the millionaires' problem based on homomorphic encryption. In: International Conference on Applied Cryptography and Network Security. pp. 456–466. Springer (2005)

[119] Lockl, J., Schlatt, V., Schweizer, A., Urbach, N., Harth, N.: Toward trust in internet of things ecosystems: Design principles for blockchain-based iot applications. IEEE Transactions on Engineering Management (2020)

[120] López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Dynamic role binding in blockchain-based collaborative business processes. In: CAISE (2019)

[121] López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Controlled flexibility in blockchain-based collaborative business processes. IS (2020)

[122] López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I.: Caterpillar: A blockchain-based business process management system. BPM (Demos) **172** (2017)

[123] Loukil, F., Boukadi, K., Abed, M., Ghedira-Guegan, C.: Decentralized collaborative business process execution using blockchain. World Wide Web **24**(5), 1645–1663 (2021)

[124] Loukil, F., Ghedira-Guegan, C., Boukadi, K., Benharkat, A.N.: Privacy-preserving iot data aggregation based on blockchain and homomorphic encryption. Sensors **21**(7), 2452 (2021)

[125] López-Pintado, O., Dumas, M., García-Bañuelos, L., et al.: Controlled flexibility in blockchain-based collaborative business processes. Information Systems (2020)

[126] Ma, J., Qi, B., Lv, K.: Fully private auctions for the highest bid. In: Proceedings of the ACM Turing Celebration Conference-China. pp. 1–6 (2019)

[127] Madsen, M.F., Gaub, M., Høgnason, T., Kirkbro, M.E., Slaats, T., Debois, S.: Collaboration among adversaries: distributed workflow execution on a blockchain. In: Symposium on Foundations and Applications of Blockchain. p. 8 (2018)

[128] Magnani, M., Montesi, D.: Bpmn: How much does it cost? an incremental approach. In: BPM. pp. 80–87. Springer (2007)

[129] Mahalle, V.S., Shahade, A.K.: Enhancing the data security in cloud by implementing hybrid (rsa & aes) encryption algorithm. In: 2014 International Conference on Power, Automation and Communication (INPAC). pp. 146–149. IEEE (2014)

[130] Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M.: Silentwhispers: Enforcing security and privacy in decentralized credit networks. Cryptology ePrint Archive (2016)

[131] Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M., Ravi, S.: Concurrency and privacy with payment-channel networks. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 455–471 (2017)

[132] Mammadzada, K., Iqbal, M., Milani, F., García-Bañuelos, L., Matulevičius, R.: Blockchain oracles: A framework for blockchain-based applications. In: International Conference on Business Process Management. pp. 19–34. Springer (2020)

[133] Mechkaroska, D., Dimitrova, V., Popovska-Mitrovikj, A.: Analysis of the possibilities for improvement of blockchain technology. In: 2018 26th Telecommunications Forum (TELFOR). pp. 1–4. IEEE (2018)

[134] Mehandjiev, N., Grefen, P.: Dynamic business process formation for instant virtual enterprises, vol. 39. Springer (2010)

[135] Mendling, J., Weber, I., Aalst, W.V.D., Brocke, J.V., Cabanillas, C., Daniel, F., Debois, S., Ciccio, C.D., Dumas, M., Dustdar, S.,

et al.: Blockchains for business process management-challenges and opportunities. ACM Transactions on Management Information Systems (TMIS) **9**(1), 1–16 (2018)

[136] Merkle, R.C.: Secrecy, authentication, and public key systems. Stanford university (1979)

[137] Meroni, G., Plebani, P.: Combining artifact-driven monitoring with blockchain: Analysis and solutions. In: International Conference on Advanced Information Systems Engineering. pp. 103–114. Springer (2018)

[138] Meroni, G., Plebani, P., Vona, F.: Trusted artifact-driven process monitoring of multi-party business processes with blockchain. In: BPM (2019)

[139] Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy. pp. 397–411. IEEE (2013)

[140] Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., Qijun, C.: A review on consensus algorithm of blockchain. In: 2017 IEEE international conference on systems, man, and cybernetics (SMC). pp. 2567–2572. IEEE (2017)

[141] Mukkamala, R.R., Hildebrandt, T., Slaats, T.: Towards trustworthy adaptive case management with dynamic condition response graphs. In: 2013 17th IEEE International Enterprise Distributed Object Computing Conference. pp. 127–136. IEEE (2013)

[142] Myung, S., Lee, J.H.: Ethereum smart contract-based automated power trading algorithm in a microgrid environment. The Journal of Supercomputing (2020)

[143] Nærland, K., Müller-Bloch, C., Beck, R., Palmund, S.: Blockchain to rule the waves-nascent design principles for reducing risk and uncertainty in decentralized environments. In: ICIS (2017)

[144] Nahabedian, L., Braberman, V., D'Ippolito, N., Kramer, J., Uchitel, S.: Assured automatic dynamic reconfiguration of business processes. Information Systems **104**, 101850 (2022)

[145] Nahabedian, L., Braberman, V., D'ippolito, N., Kramer, J., Uchitel, S.: Dynamic reconfiguration of business processes. In: BPM. pp. 35–51. Springer (2019)

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

[147] Nardini, M., Helmer, S., El Ioini, N., Pahl, C.: A blockchain-based decentralized electronic marketplace for computing resources. SN Computer Science (2020)

[148] Narendra, N.C., Norta, A., Mahunnah, M., Ma, L., Maggi, F.M.: Sound conflict management and resolution for virtual-enterprise collaborations. Service Oriented Computing and Applications **10**(3), 233–251 (2016)

[149] Narula, N., Vasquez, W., Virza, M.: {zkLedger}:{Privacy-Preserving} auditing for distributed ledgers. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). pp. 65–80 (2018)

[150] Noether, S., Noether, S.: Monero is not that mysterious. Technical report (2014)

[151] Norta, A.: Creation of smart-contracting collaborations for decentralized autonomous organizations. In: International Conference on Business Informatics Research. pp. 3–17. Springer (2015)

[152] Norta, A.: Establishing distributed governance infrastructures for enacting cross-organization collaborations. In: International Conference on Service-Oriented Computing. pp. 24–35. Springer (2015)

[153] Norta, A.: Designing a smart-contract application layer for transacting decentralized autonomous organizations. In: International Conference on Advances in Computing and Data Sciences. pp. 595–604. Springer (2016)

[154] Norta, A., Kormiltsyn, A., Udokwu, C., Dwivedi, V., Aroh, S., Nikolajev, I.: A blockchain implementation for configurable multi-factor challenge-set self-sovereign identity authentication. In: 2022 IEEE International Conference on Blockchain (Blockchain). pp. 455–461. IEEE (2022)

[155] Norta, A., Ma, L., Duan, Y., Rull, A., Kõlvart, M., Taveter, K.: econtractual choreography-language properties towards cross-organizational business collaboration. Journal of Internet Services and Applications **6**(1), 1–23 (2015)

[156] Norta, A., Othman, A.B., Taveter, K.: Conflict-resolution lifecycles for governed decentralized autonomous organization collaboration. In: Proceedings of the 2015 2nd International Conference on Electronic Governance and Open Society: Challenges in Eurasia. pp. 244–257 (2015)

[157] Norta, A.: Exploring Dynamic Inter-Organizational Business Process Collaboration. Ph.D. thesis, TU-Eindhoven, NL. (20O7)

[158] Oliveira, L., Zavolokina, L., Bauer, I., Schwabe, G.: To token or not to token: Tools for understanding blockchain tokens. In: 39th International Conference on Information Systems, San Francisco (2018)

[159] Oranburg, S., Palagashvili, L.: The gig economy, smart contracts, and disruption of traditional work arrangements. Smart Contracts, and Disruption of Traditional Work Arrangements (2018)

[160] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)

[161] Palacin, L.: Accelerate blockchain technology adoption with bonita bpm and chain core. URL https://vimeo. com/202058656. Accessed pp. 04–08 (2018)

[162] Pan, S., Trentesaux, D., McFarlane, D., Montreuil, B., Ballot, E., Huang, G.Q.: Digital interoperability in logistics and supply chain management: state-of-the-art and research avenues towards physical internet. Computers in Industry **128**, 103435 (2021)

[163] Papadis, N., Tassiulas, L.: Blockchain-based payment channel networks: Challenges and recent advances. IEEE Access **8**, 227596–227609 (2020)

[164] Papazoglou, M.P., Van Den Heuvel, W.J.: Business process development life cycle methodology. Communications of the ACM **50**(10), 79–85 (2007)

[165] Park, J., Chitchyan, R., Angelopoulou, A., Murkin, J.: A block-free distributed ledger for p2p energy trading: Case with IOTA? In: Advanced Information Systems Engineering. pp. 111–125. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2-8, `https://link.springer.com/chapter/10.1007/978-3-030-21290-2-8`

[166] Pelaitis, D., Spathoulas, G.: Developing a universal, decentralized and immutable erasmus credit transfer system on blockchain. In: 2018 Innovations in Intelligent Systems and Applications (INISTA). pp. 1–6. IEEE (2018)

[167] Peltz, C.: Web services orchestration and choreography. Computer **36** (2003)

[168] Peng, L., Feng, W., Yan, Z., Li, Y., Zhou, X., Shimizu, S.: Privacy preservation in permissionless blockchain: A survey. Digital Communications and Networks **7**(3), 295–307 (2021)

[169] Pinna, A., Ibba, S.: A blockchain-based decentralized system for proper handling of temporary employment contracts. In: SI conference. Springer (2018)

[170] Pintado, O.L.: Challenges of blockchain-based collaborative business processes: An overview of the caterpillar system. Blockchain and Robotic Process Automation pp. 31–42 (2021)

[171] Piriou, P.Y., Boudeville, O., Deleuze, G., Tucci-Piergiovanni, S., Gürcan, Ö.: Justifying the dependability and security of business-critical blockchain-based applications. In: 2021 Third International Conference on Blockchain Computing and Applications (BCCA). pp. 97–104. IEEE (2021)

[172] Pittl, B., Starflinger, S., Mach, W., Schikuta, E.: Bazaar-contract: A smart contract for binding multi-round bilateral negotiations on cloud markets. In: FiCloud (2019)

[173] Poelstra, A., Back, A., Friedenbach, M., Maxwell, G., Wuille, P.: Confidential assets. In: International Conference on Financial Cryptography and Data Security. pp. 43–63. Springer (2018)

[174] Pons, J.: Blockchains and smart contracts in the culture and entertainment business. Réalités industrielles (2017)

[175] Pries-Heje, J., Baskerville, R., Venable, J.R.: Strategies for design science research evaluation. ECIS Proceedings. 87. (2008)

[176] Prybila, C., Schulte, S., Hochreiner, C., Weber, I.: Runtime verification for business processes utilizing the bitcoin blockchain. FGCS **107**, 816–831 (2020)

[177] Rella, L.: Steps towards an ecology of money infrastructures: materiality and cultures of ripple. Journal of Cultural Economy **13**(2), 236–249 (2020)

[178] Reuters: Sweden tests blockchain technology for land registry (2020), `https://www.reuters.com/article/us-sweden-blockchain/sweden-tests-blockchain-technology-for-land-registry-idUSKCN0Z22KV`, Accessed on 2020-01-28

[179] Rivest, R.L., Adleman, L., Dertouzos, M.L., et al.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978)

[180] Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)

[181] Röck, D.: The foundation of distributed ledger technology for supply chain management. In: HICSS (2020)

[182] Ryu, S.H., Casati, F., Skogsrud, H., Benatallah, B., Saint-Paul, R.: Supporting the dynamic evolution of web service protocols in service-oriented architectures. ACM Transactions on the Web (TWEB) **2**(2), 1–46 (2008)

[183] Sambamurthy, V., Bharadwaj, A., Grover, V.: Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. MIS Quarterly **27**, 237–263 (2003). https://doi.org/10.2307/30036530

[184] Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE symposium on security and privacy. pp. 459–474. IEEE (2014)

[185] Schulte, S., Sigwart, M., Frauenthaler, P., Borkowski, M.: Towards blockchain interoperability. In: International conference on business process management. pp. 3–10. Springer (2019)

[186] Scoping, S., Taskforce, T.: Information supplement: Pci dss tokenization guidelines. Standard: PCI Data Security Standard (PCI DSS) **24** (2011)

[187] Sharma, R., Wingreen, S., Kshetri, N., Hewa, T.: Design principles for use cases of blockchain in food supply chains. In: AMCIS (2019)

[188] Singh, A., Click, K., Parizi, R.M., Zhang, Q., Dehghantanha, A., Choo, K.K.R.: Sidechain technologies in blockchain networks: An examination and state-of-the-art review. Journal of Network and Computer Applications **149**, 102471 (2020)

[189] Slaats, T.: Flexible process notations for cross-organizational case management systems (2015)

[190] Slaats, T., Hildebrandt, T.T., Carbone, M., Völzer, H.: Flexible process notations for cross-organizational case management systems. ITU Copenhagen (2015)

[191] Smiley, L.: Blockchain-based e-auction to fight corruption in ukraine, `https://cointelegraph.com/news/blockchain-based-e-auction-to-fight-corruption-in-ukraine`

[192] Sonnino, A., Król, M., Tasiopoulos, A.G., Psaras, I.: Asterisk: Auction-based shared economy resolution system for blockchain. arXiv preprint arXiv:1901.07824 (2019)

[193] Amaral de Sousa, V., Burnay, C., Snoeck, M.: B-merode: a model-driven engineering and artifact-centric approach to generate blockchain-based information systems. In: International Conference on Advanced Information Systems Engineering. pp. 117–133. Springer (2020)

[194] Strehle, E., Maurer, M.: The dibichain protocol: Privacy-preserving discovery and exchange of supply chain information. In: International Conference on Model and Data Engineering. pp. 231–247. Springer (2021)

[195] Sturm, C., Scalanczi, J., Schönig, S., Jablonski, S.: A blockchain-based and resource-aware process execution engine. FGCS (2019)

[196] Sun, S.F., Au, M.H., Liu, J.K., Yuen, T.H.: Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: European Symposium on Research in Computer Security. pp. 456–474. Springer (2017)

[197] Suri, K., Cadavid, J., Alferez, M., Dhouib, S., Tucci-Piergiovanni, S.: Modeling business motivation and underlying processes for rami 4.0-aligned cyber-physical production systems. In: 2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA). pp. 1–6. IEEE (2017)

[198] Szabo, N.: Formalizing and securing relationships on public networks. First monday (1997)

[199] Telegraph, C.: Georgia records 100,000 land titles on bitcoin blockchain: BitFury (2020), https://cointelegraph.com/news/georgia-records-100000-land-titles-on-bitcoin-blockchain-bitfury

[200] Togan, M., Pleşca, C.: Comparison-based computations over fully homomorphic encrypted data. In: 2014 10th international conference on communications (COMM). pp. 1–6. IEEE (2014)

[201] Tran, A.B., Lu, Q., Weber, I.: Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In: BPM (Dissertation/Demos/Industry). pp. 56–60 (2018)

[202] Tripathy, S., Mohanty, S.K.: Mappcn: Multi-hop anonymous and privacy-preserving payment channel network. In: International Conference on Financial Cryptography and Data Security. pp. 481–495. Springer (2020)

[203] Troncia, M., Galici, M., Mureddu et al, M.: Distributed ledger technologies for peer-to-peer local markets in distribution networks. Energies (2019)

[204] Uahi, R., Pereira, J.L.: Task allocation in business processes supported by bpms: Optimization perspectives. In: 2016 11th Iberian Conference on Information Systems and Technologies (CISTI). pp. 1–6. IEEE (2016)

[205] Underwood, S.: Blockchain beyond bitcoin. ACM **59**(11), 15–17 (2016)

[206] Van Der Aalst, W.M., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: From public views to private views–correctness-by-design for services. In: International Workshop on Web Services and Formal Methods. pp. 139–153. Springer (2007)

[207] Victor, F., Lüders, B.K.: Measuring ethereum-based erc20 token networks. In: International Conference on Financial Cryptography and Data Security. pp. 113–129. Springer (2019)

[208] De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Distributed query evaluation over encrypted data. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 96–114. Springer (2021)

[209] Vom Brocke, J., Mathiassen, L., Rosemann, M.: Business process management (2014)

[210] Wang, D., Zhao, J., Wang, Y.: A survey on privacy protection of blockchain: The technology and application. IEEE Access **8**, 108766–108781 (2020)

[211] Wang, S., Qu, X.: Blockchain applications in shipping, transportation, logistics, and supply chain. In: Smart Transportation Systems 2019, pp. 225–231. Springer (2019)

[212] Wang, S., Huang, X., Yu, R., Zhang, Y., Hossain, E.: Permissioned blockchain for efficient and secure resource sharing in vehicular edge computing. arXiv preprint arXiv:1906.06319 (2019)

[213] Weber, C.: Exploring dlt and blockchain for alternative finance: A collection of case studies. Tech. rep., Technical report, European Crowdfunding Network, November (2019)

[214] Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: International conference on business process management. pp. 329–347. Springer (2016)

[215] Webster, J., Watson, R.: Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly **26** (06 2002). https://doi.org/10.2307/4132319

[216] Weinhardt, C., Gimpel, H.: Market engineering: An interdisciplinary research challenge. In: Dagstuhl seminar proceedings (2007)

[217] Wiemuth, M., Junger, D., Leitritz, M., Neumann, J., Neumuth, T., Burgert, O.: Application fields for the new object management group (omg) standards case management model and notation (cmmn) and decision management notation (dmn) in the perioperative field. International journal of computer assisted radiology and surgery **12**(8), 1439–1449 (2017)

[218] Wood, G.: A secure decentralised generalised transaction ledger, ethereum proj. Yellow Pap (2014)

[219] Xiang, G., Cui, Z.: The algebra homomorphic encryption scheme based on fermat's little theorem. In: 2012 international conference on communication systems and network technologies. pp. 978–981. IEEE (2012)

[220] Xiong, F., Xiao, R., Ren, W., Zheng, R., Jiang, J.: A key protection scheme based on secret sharing for blockchain-based construction supply chain system. IEEE Access **7**, 126773–126786 (2019)

[221] Xu, X., Weber, I., Staples, M.: Architecture for Blockchain Applications. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-03035-3, http://link.springer.com/10.1007/978-3-030-03035-3

[222] Xu, X., Weber, I., Staples, M.: Blockchain patterns. In: Architecture for Blockchain Applications, pp. 113–148. Springer (2019)

[223] Xu, X., Weber, I., Staples, M.: Case Study: AgriDigital: Blockchain Technology in the Trade and Finance of Agriculture Supply Chains, pp. 239–255. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-03035-3-12, http://link.springer.com/10.1007/978-3-030-03035-3-12

[224] Xue, X., Dou, J., Shang, Y.: Blockchain-driven supply chain decentralized operations–information sharing perspective. Business Process Management Journal (2020)

[225] Yeh, J.H.: A probabilistic homomorphic encryption algorithm over integers-protecting data privacy in clouds. In: 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom). pp. 653–656. IEEE (2015)

[226] Yu, B., Kermanshahi, S.K., Sakzad, A., Nepal, S.: Chameleon hash time-lock contract for privacy preserving payment channel networks. In: International Conference on Provable Security. pp. 303–318. Springer (2019)

[227] Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town crier: An authenticated data feed for smart contracts. In: Proceedings of the 2016 aCM sIGSAC conference on computer and communications security. pp. 270–282 (2016)

[228] Zhang, Y., Lee, C., Niyato, D., Wang, P.: Auction approaches for resource allocation in wireless systems: A survey. IEEE Communications Surveys Tutorials **15**(3), 1020–1041 (2013). https://doi.org/10.1109/SURV.2012.110112.00125

[229] Zhao, R.: An empirical analysis of supply chain BPM model based on blockchain and IoT integrated system. In: Web Information Systems and Applications. pp. 539–547. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30952-7-54, `https://link.springer.com/chapter/10.1007/978-3-030-30952-7-54`

[230] Zouina, M., Outtai, B.: Towards a distributed token based payment system using blockchain technology. In: 2019 International Conference on Advanced Communication Technologies and Networking (CommNet). pp. 1–10. IEEE (2019)

Appendices

Examples of DCR graph inputs

Delivery process - Customer view

```
pk[role=Driver] = 0x89033bC8f73Ef5b46CCb013f6F948b00954a06BB
pk[role=Florist] = 0x1ED034135e576A6c1bf3ee8E05aaDEEF24D4A819
pk[role=Customer] = 0x5AfBDd0e5DE3315a96504C06ac49bF34B5ECACB5

## declare events

# Choreography exchanges
e1[Shipping src=Driver tgt=Florist tgt=Customer]
e2[CheckDelivery src=Customer tgt= Florist tgt=Driver]
e3[Accept src=Customer tgt=Florist tgt=Driver]
e4[Reject src=Customer tgt=Florist tgt=Driver]
e5[Pay src=Customer tgt=Florist]
e6[UnloadTruck src=Driver tgt=Customer tgt=Florist]

## link events
e1 --<> e2
e1 *--> e2

e2 *--> e3
e2 --<> e3
e2 *--> e4
e2 --<> e4


e3 -->+ e5
e3 *--> e5
e3 -->+ e6
e3 *--> e6
```

```
## exclude events
e3 -->% e3
e4 -->% e4

e4 -->% e5
e4 -->% e6

e5 -->% e5
e6 -->% e6
```

Delivery process - Driver view

```
## declare events

# Choreography exchanges
e1[Shipping src=Driver tgt=Florist tgt=Customer]
e2[CheckDelivery src=Customer tgt= Florist tgt=Driver]
e3[Accept src=Customer tgt=Florist tgt=Driver]
e4[Reject src=Customer tgt=Florist tgt=Driver]
e6[UnloadTruck src=Driver tgt=Customer tgt=Florist]
e7[PayDriver src=Florist tgt=Driver]

# Internal processes
"ReturnTruck" [role=Driver]

## link events
e1 --<> e2
e1 *--> e2

e2 *--> e3
e2 --<> e3
e2 *--> e4
e2 --<> e4


e3 -->+ e6
e3 *--> e6

ReturnTruck -->* e7
ReturnTruck *--> e7
```

```
e4 *--> e7
e5 *--> e7
e6 *--> e7

e4 *--> ReturnTruck
e5 *--> ReturnTruck
e6 *--> ReturnTruck

## exclude events
e3 -->% e3
e4 -->% e4
e4 -->% e5
e4 -->% e6

e6 -->% e6
ReturnTruck -->% ReturnTruck
```

Delivery process - Florist view

```
pk[role=Driver] = 0x89033bC8f73Ef5b46CCb013f6F948b00954a06BB
pk[role=Florist] = 0x1ED034135e576A6c1bf3ee8E05aaDEEF24D4A819
pk[role=Customer] = 0x5AfBDd0e5DE3315a96504C06ac49bF34B5ECACB5

## declare events

# Choreography exchanges
e1[Shipping src=Driver tgt=Florist tgt=Customer]
e2[CheckDelivery src=Customer tgt= Florist tgt=Driver]
e3[Accept src=Customer tgt=Florist tgt=Driver]
e4[Reject src=Customer tgt=Florist tgt=Driver]
e5[Pay src=Customer tgt=Florist]
e6[UnloadTruck src=Driver tgt=Customer tgt=Florist]
e7[PayDriver src=Florist tgt=Driver]

# Internal processes
"PrepareCommand" [role=Florist]
"CallShipper" [role=Florist]
"SettleCommand" [role=Florist]

## link events
e1 --<> e2
e1 *--> e2
```

```
e2 *--> e3
e2 --<> e3

e2 *--> e4
e2 --<> e4

e3 -->+ e5
e3 *--> e5
e3 -->+ e6
e3 *--> e6

PrepareCommand -->* CallShipper
PrepareCommand *--> CallShipper

CallShipper -->* e1
CallShipper *--> e1

e7 -->* SettleCommand
e7 *--> SettleCommand

e4 *--> e7
e5 *--> e7
e6 *--> e7

e4 *--> SettleCommand
e5 *--> SettleCommand
e6 *--> SettleCommand

## exclude events
e3 -->% e3
e4 -->% e4

e4 -->% e5
e4 -->% e6

e5 -->% e5
e6 -->% e6
SettleCommand -->% SettleCommand
```

Proof of Concepts

- Hybrid on/off-chain DCR choreography deployment and execution using blockchain https://archive.softwareheritage.org/swh:1:dir:211c6bdb1ce9f256c363caee54a56f53ada05d9b;origin=https://github.com/tiphainehenry/hybridChoreo;visit=swh:1:snp:6376000436aeaf872f39fb5f1a9d5aaa417c5cea;anchor=swh:1:rev:a5dd01e0ed0cb034c1e35daa0e6b3c6e08ff6d97
- Hybrid on/off-chain DCR choreography deployment and execution using blockchain with running instance change support https://archive.softwareheritage.org/swh:1:dir:8bde592496c231084627448dbd2399446f1cf2ce;origin=https://github.com/tiphainehenry/adaptiveChangeDCR;visit=swh:1:snp:89cd57a878c58593ac077da79d99a0f00cb8ac41;anchor=swh:1:rev:fac4a9a64d86fac19dcb11515b35de2de402ff1e
- Dynamic and trustworthy QoS-based allocation

  https://archive.softwareheritage.org/swh:1:dir:cc1dc66c637f8427836d808745f1bf1630816527;origin=https://github.com/tiphainehenry/smart-logistics;visit=swh:1:snp:d97e3a03446f13ef7112faba5abec02254568449;anchor=swh:1:rev:e06b091de377c1bfe29644fb2b539af99381dd1e
- Trustworthy sealed-bids allocation using FHE https://archive.softwareheritage.org/swh:1:dir:e08b491a6240075052af4c13709f95a83ef52ae6;origin=https://github.com/tiphainehenry/fhe_oracle;visit=swh:1:snp:05d0f85de663d57daabd74fe5fbc38a83cb2c953;anchor=swh:1:rev:8666576002ad5f375f9550798630a468681f29c6
- Privacy-preserving payments with random-value tokens https://archive.softwareheritage.org/swh:1:dir:60577355b219ab188003bdaa624a31cf564f2b98;origin=https://github.com/tiphainehenry/random-value-token-payment;visit=swh:1:snp:dd7f3b87913aef5990d76d4c7fc3e11998cb0a8d;anchor=swh:1:rev:e2ad349e7d65c6a3dd735d48ce5a9e73ff9541c0

Résumé Etendu

La technologie Blockchain a été introduite comme un outil de désintermédiation fiable pour la gestion des processus métiers inter-organisationnels au cours de la dernière décennie. La blockchain assure la traçabilité de l'exécution des activités tout en appliquant le flux de contrôle préalablement convenu au moment de la conception avec les autres partenaires. De plus, les contrats intelligents (smart contracts) peuvent contribuer à l'automatisation fiable des tâches redondantes. Cependant, assurer la confiance dans le protocole de déploiement et d'exécution, la flexibilité des processus et la confidentialité des données reste un défi dans l'environnement de la blockchain.

Tout d'abord, le protocole de déploiement et d'exécution des instances de gestion des processus métier nécessite une séparation fiable des vues métier de chacun des participants. En effet, les données internes telles que l'historique d'exécution de tâches privées ne doivent pas être rendues visibles à d'autres partenaires lors du cycle de vie du processus. Par conséquent, un compromis entre la garantie de la confidentialité des processus privés des partenaires et l'exposition des processus publics par l'intermédiaire du réseau blockchain survient lors du déploiement et de l'exécution du processus.

De plus, la nature dynamique des processus inter-organisationnels nécessite (i) une adaptation du processus à l'exécution et (ii) une affectation dynamique des acteurs. La plupart des travaux connexes prennent uniquement en compte les changements dans les orchestrations de processus. De plus, les approches liant les acteurs aux rôles dans une collaboration de processus demandent aux acteurs de vérifier eux-mêmes l'effet transitif des nouveaux changements sur les nouvelles parties. Enfin, à notre connaissance, aucune règle d'allocation basée sur les notes de qualité de service et gérée par contrats intelligents n'a été proposée dans la littérature.

Le troisième défi concerne la confidentialité des données de processus sensibles. En effet, les processus inter-organisationnels exploitent des informations sensibles comme le prix des offres. Néanmoins, les propriétés d'ouverture et de transparence de la blockchain mettent en cause cet impératif de confidentialité totale ou partielle dans les processus métier inter-organisationnels. Il existe une lacune dans l'état de l'art concernant (1) l'allocation des services basée sur le tri multi-objectifs de manière à préserver

la confidentialité tout en garantissant la vérifiabilité, ainsi que (2) une gestion des paiements par contrat intelligent qui préserve la confidentialité des offres et l'auditabilité des échanges sur le registre blockchain.

Dans ce manuscrit, nous proposons les trois contributions suivantes visant à solutionner les problématiques susmentionnées.

Tout d'abord, nous concevons et mettons en œuvre une stratégie de déploiement et d'exécution on/off-chain pour les processus chorégraphiques. Cette solution assure une séparation fiable des vues métier entre les participants à chaque étape du déploiement et de l'exécution des instances. Ce faisant, nous tirons parti d'un langage de modèle de processus métiers déclaratif, DCR, qui permet d'abstraire le flux de contrôle.

Deuxièmement, nous proposons d'apporter une flexibilité au flux de contrôle du système de gestion de processus métier. Pour ce faire, nous proposons un mécanisme de changement d'instances de chorégraphies DCR en cours d'exécution sur la blockchain. Notre système permet à un partenaire dans une instance en cours d'exécution de modifier son processus DCR privé. Un changement affectant d'autres partenaires est propagé aux processus concernés à l'aide d'un contrat intelligent. Nous proposons également un système tirant parti des contrats intelligents pour une sélection dynamique de fournisseurs de services. Le système analyse les performances des fournisseurs de services stockées sous forme de logs blockchain et décide dynamiquement de l'attribution d'une tâche en fonction des notes de qualité de service de chacun des candidats.

Enfin, nous proposons deux mécanismes préservant la confidentialité des enchères et des paiements dans un contexte blockchain. Le premier mécanisme se base sur du chiffrement entièrement homomorphe. Les calculs gérés par la blockchain opèrent ainsi sur des nombres chiffrés. En pratique, un contrat intelligent rassemble et orchestre la comparaison des offres, tandis qu'un oracle effectue les comparaisons sur des données chiffrées. Le deuxième mécanisme utilise les banques en tant qu'intermédiaires de confiance. Cette solution utilise une banque et un jeton de paiement lié à une valeur aléatoire. Les partenaires peuvent utiliser des jetons pour procéder à plusieurs paiements tout en préservant la confidentialité des valeurs. Ce système permet de plus de préserver la traçabilité des offres, les transactions étant stockées dans la blockchain. Ainsi, les pairs externes peuvent auditer de manière fiable l'historique des paiements.

Nous démontrons la faisabilité de chaque contribution au travers d'un prototype et son efficacité via des expérimentations ancrées dans le domaine logistique.

ECOLE
DOCTORALE

Titre : Vers une gestion de processus métiers pair à pair fiable, flexible, et respectueuse de la vie privée

Mots clés : Processus métiers, chaînes de blocs, chorégraphies

Résumé : La technologie blockchain peut être utilisée comme outil de désintermédiation fiable pour la gestion des processus métiers inter-organisationnels. Elle assure la traçabilité de l'exécution des activités et du flux de contrôle. Cependant, assurer la confiance dans le protocole de déploiement et d'exécution, permettre la flexibilité des processus et respecter la confidentialité des données restent trois verrous à solutionner dans cet environnement.
Pour résoudre les problématiques susmentionnées , nous présentons les contributions suivantes.
D'abord, nous présentons un mécanisme de déploiement et d'exécution on/off-chain pour les processus chorégraphiques. Cette solution assure une séparation fiable des vues entre les participants à chaque étape du déploiement et de l'exécution des instances. Ce faisant, nous tirons parti d'un langage de modèle de processus métiers déclaratif, DCR, qui permet d'abstraire le flux de contrôle.
Deuxièmement, nous proposons un mécanisme de changement d'instances de chorégraphies DCR en cours d'exécution sur la blockchain. Un changement affectant d'autres partenaires est propagé aux processus concernés via un contrat intelligent. Nous proposons également une sélection dynamique de fournisseurs de services gérée par contrat intelligent.
Enfin, nous proposons deux mécanismes préservant la confidentialité des enchères et des paiements dans un contexte blockchain. Le premier mécanisme se base sur du chiffrement entièrement homomorphe. Les calculs gérés par la blockchain opèrent sur des nombres chiffrés. Le deuxième mécanisme utilise une banque et un jeton de paiement (NFT) à valeur aléatoire. Les partenaires utilisent ces jetons pour procéder à plusieurs paiements tout en préservant la confidentialité des valeurs.
Nous démontrons la faisabilité de chaque contribution au travers d'un prototype et son efficacité via des expérimentations ancrées dans le domaine logistique.

Title : Towards trustworthy, flexible, and privacy-preserving peer-to-peer business process management systems

Keywords : Business Process Models, Blockchain, Choreographies

Abstract : Blockchain technology has been introduced as a trustworthy disintermediation tool for managing cross-organizational business processes in the past decade. It ensures activities' execution traceability while enforcing the control flow agreed upon at design time with other partners.
However, ensuring trust in the deployment and execution protocol, process flexibility, and data privacy remains challenging in this environment.
To address these challenges, we propose the three following contributions in this manuscript.
First, we design and implement an on/off-chain deployment and execution strategy for on/off-chain choreographies, which enforces a trustworthy separation of concern between participants at each step of the deployment and execution. Meanwhile, we leverage a constraint-based business model language, DCR, abstracting the control flow under a set of constraints.
Second, we propose to bring control-flow flexibility to the blockchain-based business process management system through change management: a change impacting other partners is propagated to affected processes using a smart contract. We also leverage smart contracts for a dynamic selection of service providers. The system analyses service providers' performance stored as blockchain logs and dynamically decides on the allocation of a task based on the QoS outputs.
Finally, we propose two mechanisms to reconcile privacy imperatives with the benefits of blockchain. The first mechanism leverages fully homomorphic encryption for blockchain-based calculations such as sealed-bid auctions. Smart contracts gather and orchestrate bid comparison, while a computation oracle carries comparisons over ciphered data. The second mechanism leverages banks as trustworthy intermediaries while secreting the payment value. Partners can use per-collaboration tokens backed by a random value to proceed to multiple payments confidentially.
We demonstrate the feasibility of each contribution through an implemented prototype and its effectiveness via experiments anchored in the logistics domain.

Institut Polytechnique de Paris
91120 Palaiseau, France