Designing secure business processes for blockchains with SecBPMN2BC

Julius Köpke [a,*], Giovanni Meroni [b,c], Mattia Salnitri [b]

[a] *University of Klagenfurt, Universitaetsstrasse 65-67, Klagenfurt, Austria*
[b] *Politecnico di Milano, Via Ponzio 34/5, Milano, Italy*
[c] *Technical University of Denmark, Richard Petersens Plads, Build. 321, Kgs. Lyngby, Denmark*

ARTICLE INFO

ABSTRACT

Collaborative business processes can be seen as smart contracts, as they are oftentimes adopted to express agreements among different organizations. Indeed, they provide mechanisms to formalize the obligations of each involved party. For instance, collaborative business processes can specify when a certain task should be executed, under which conditions a service should be offered to the other participants, and how physical objects and information should be manipulated. In this setting, to prevent misuse of smart contracts and services and information provided, it is paramount to guarantee by *design* that security requirements are fulfilled. With the rise in popularity of blockchains, several approaches exploiting the trusted smart contract execution environment offered by this technology to enforce collaborative business processes have been proposed. Yet, the complexity of business processes, security requirements, and blockchain applications calls for an engineering approach that guides the design of secure business processes. Such an approach should both take advantage of the possibilities offered by blockchain technology to enforce some security requirements (e.g., non-repudiation), and take into account the limitations blockchain poses for other security requirements (e.g., confidentiality). However, we are not aware of any existing work that aims at addressing such issues following a similar approach.

In this article, we propose SecBPMN2BC: a model-driven approach to designing business processes with security requirements that are meant to be deployed on blockchains. SecBPMN2BC consists of: (i) an extension of BPMN 2.0 that allows designing secure smart contracts; (ii) a set of algorithms and their implementation that check incompatible security requirements and help the design of smart contracts; (iii) a workflow that guides the application of the method. The method has been validated with a survey conducted on security and BPMN experts.

1. Introduction

Smart contracts [1], as introduced by N. Szabo in the last millennium, aim at automating traditional contracts with hardware and software. A simple example is a vending machine that automatically executes contracts between buyers and sellers. The introduction of blockchains and the re-invention of smart contracts [2] as code executed on blockchains has led to new and promising platforms for the execution of N. Szabo's original smart contracts.[1] For example, to increase transparency after some scandals involving bribery and inefficiency, a municipality may decide to enforce the obligations between the citizens and external contractors for road misconstructions in form of a smart contract. In particular, whenever a citizen reports a claim to the municipality, the municipality is supposed to check the claim and its urgency and then decide to repair the misconstruction immediately, refund the citizen with a gift card, or plan the fix for the near future. Executing this smart contract on a blockchain could increase the transparency between the involved parties, as it would make it impossible for any of them to ignore claims or to cover their actions.

Smart contracts can be implemented manually in form of smart contract code executed on a blockchain. However, manual implementations are typically time-consuming, and error-prone [3]. Errors in smart contract code can lead to disastrous outcomes, as assets, e.g., funds, can easily be locked forever. Therefore, model-driven development of smart contracts is considered highly beneficial [4].

---

* Corresponding author. All authors contributed equally and were ordered alphabetically.
E-mail addresses: julius.koepke@aau.at (J. Köpke), giom@dtu.dk (G. Meroni), mattia.salnitri@polimi.it (M. Salnitri).

[1] In order to distinguish between original smart contracts and the ones on blockchains, we refer to the latter ones as smart contract code.

Many smart contracts can be represented in form of (inter-organizational) business processes [5]. For example, the smart contract for road misconstructions could be represented as the business process shown in Fig. 3. The execution of such smart contracts can, therefore, be supported by executing business processes on blockchains. Blockchain technology can naturally provide desired features such as observability and immutability. The potential of blockchains for business processes is also witnessed by a large body of research on the execution of business processes on blockchains [6–9]. Many existing approaches compile process models defined in Business Process Model and Notation (BPMN) into smart contract code. Consequently, business process instances are executed by the participants by calling the generated smart contract transactions.

While these approaches are building blocks for the model-driven development and execution of business processes executed on blockchains, they are limited to the expressiveness of BPMN. Indeed, BPMN lacks security concepts preventing modelers to define secure processes and, therefore, secure smart contracts. This is a serious shortcoming in the context of implementations for blockchain since security is a major concern and a driver for transaction costs. Smart contract codes are the most targeted blockchain components with more than 44% of the total security attacks in 2015 [10], with a loss of crypto assets worth 7.8 billion dollars between 2011 and 2020 [11].

We, therefore, argue that inter-organizational business processes that are realized via smart contract code should be developed based on the security-by-design approach. This paper will, therefore, answer the following research questions:

[RQ1] how to design secure business processes representing smart contracts?

[RQ2] how to model secure business processes for blockchain?

[RQ3] how to determine which business process elements should be executed on blockchain?

[RQ4] how to guide users to design, verify and implement secure business processes using blockchain?

The difference between RQ1 and RQ2 lays in how the modeling language is used: RQ1 focuses on security requirements independent of the implementation, while RQ2 focuses on the design of a blockchain-based solution.

Consequently, in this work, we introduce SecBPMN2BC, a method that guides process modelers and security experts in the design of smart contracts as secure business processes targeting blockchain-based implementations. SecBPMN2BC will also assess if and to what degree security requirements expressed in process models can be enforced on blockchains.

This article provides the following contributions:

1- to address RQ4, a workflow that guides the model-driven design of secure smart contracts specified as business processes (Section 2).

2- to address RQ1 and RQ2, SecBPMN2BC-ML: an extension of the BPMN 2.0 modeling language covering security and blockchain-specific requirements (Section 4).

3- to address RQ3, SecBPMN2BC-Tools: a set of rules and algorithms that define criteria to determine which parts of a business process should be executed on a blockchain, and that are used to check for security requirements incompatible with each other (Sections 5, 6).

4- An evaluation of the SecBPMN2BC method with an empirical experiment (Section 7).

To make this article self-contained, the foundations on which SecBPMN2BC is based are described in Section 3. Section 8 surveys the state of the art. Finally, Section 9 draws the conclusions and outlines future research directions.
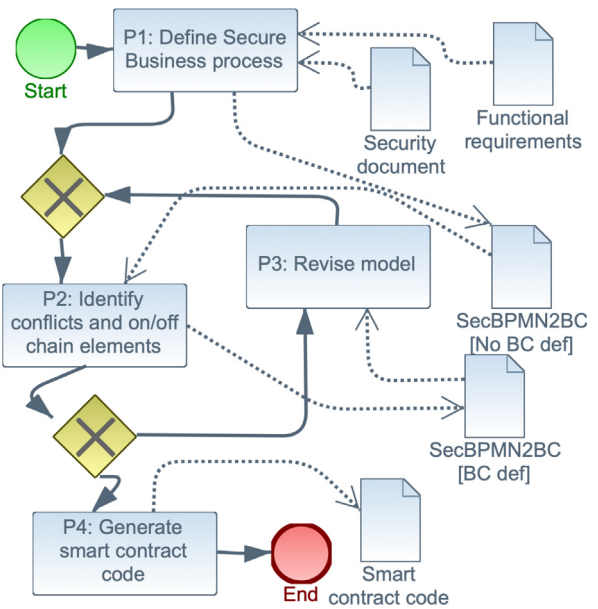

Fig. 1. SecBPMN2BC Method.

2. SecBPMN2BC method

We followed the Design Science Research Method (DSRM) [12] for the organization of the activities of the research work that led to the results described in this article. DSRM is a well-known method for design science that was defined aggregating many methods for design science and research work. It identifies activities that guide researchers from the definition of the problem to the communication of the results. DSRM led to the definition of the SecBPMN2 to BlockChain (SecBPMN2BC) method, which is described in the rest of the article. This method supports process modelers and security experts in the engineering of smart contracts represented as business processes. The method, therefore, supports the design, assessment, and deployment of secure business processes on a blockchain. It takes a model-driven and security-by-design approach, where smart contracts are designed using a graphical modeling language, named SecBPMN2BC-ML, that allows to represent them as business processes and to express their security requirements.

SecBPMN2BC requires as input functional specifications of the smart contract to be designed, as well as a security document where security requirements are defined. The output of the method is secure smart contract code. In this article, we use the term "security" in a broad sense, including privity and enforceability concepts.

SecBPMN2BC is composed of the following artifacts: the SecBPMN2BC-ML graphical modeling language, rules and algorithms that are supported by a software tool named SecBPMN-2BC-Tools, and a workflow that guides the users.

The workflow of the method follows an engineering approach for the design of secure business processes targeting blockchain-based implementations. The workflow phases are shown in Fig. 1. P1 models a smart contract as a secure business process using SecBPMN2BC-ML; P2 identifies conflicts and BPMN elements to be executed or stored on a blockchain using SecBPMN2BC-Tools; P3 revises the business process to resolve conflicts; P4 generates smart contract code based on the specified process. The process is iterative since, after P2, designers may decide to revise the process (P3) and repeat P2, or to generate the smart contract code and complete the workflow. Fig. 1 represents the SecBPMN2BC-ML models with data objects that can

have two states: "No BC def" and "BC def". The former represents a SecBPMN2BC-ML model for smart contracts without blockchain (BC) specific properties, while the latter represents a SecBPMN2BC-ML model with blockchain-specific properties. At the first iteration, P2 uses the output of P1, i.e., a SecBPMN2BC-ML model without blockchain-specific properties, while at the next iteration, it uses SecBPMN2BC-ML model with blockchain-specific properties. Besides the workflow, the other artifacts proposed in this article – that is, SecBPMN2BC-ML and SecBPMN2BC-Tools – focus on assisting the execution of the first three phases of the method while leaving P4 as a manual activity. As future work, we will extend SecBPMN2BC-Tools to assist the designers also in the generation of smart contract code starting from a SecBPMN2BC-ML model.

The core part of this article thoroughly details each component of SecBPMN2BC method.

3. Baseline

This section introduces the foundations of the SecBPMN2BC method: the concept of blockchain, the SecBPMN2 modeling language, smart contracts, privity, and enforceability requirements.

3.1. Blockchain

A blockchain is a protocol for the decentralized and fully replicated storage of a tamper-proof sequence of *transactions*, maintained and verified by the nodes participating in the network. Transactions are created by users or software agents and are cryptographically signed. Most blockchain systems follow the order–execute architecture, where signed transactions are broadcasted to the nodes of the blockchain, who validate them by replaying their underlying code and collating them into so-called blocks. Distribution and replication require an agreement on the content of the chain. This is realized by different consensus protocols.

Depending on the used consensus protocol, blocks are created by so-called miners or ordering nodes. The newly created blocks are then broadcasted to all nodes, who validate the blocks and append them to their local ledger. Each block contains the digest of its predecessor, thus creating a chain-like structure. In this way, the blockchain systems guarantee immutability and persistence: it is impossible to delete or alter a transaction without changing the digest of the block, which would break the chain. Nodes participating in the network guarantee that transactions and blocks are valid (i.e., the chain is not broken), and this prevents the data structure from being tampered with. Also, the replication of the ledger makes it possible to have the stored data always available locally to every node.

Blockchains can be divided into public and private ones. Public blockchains allow anybody to access all transactions, issue new ones, and take part in the validation process. Private blockchains limit access to the blockchain to a specific set of participants. This can be realized by deploying public blockchain software on a private network, or it can be established via permissioned blockchains. Permissioned blockchains provide access control mechanisms on various levels, such as access to the ledger, permission to issue transactions, and permission to participate in the consensus protocol. In addition, they often support the creation of internal partitions, named channels, to further restrict the information being manipulated by each participant. Depending on the access rights, participants will be able to access, validate, or issue transactions within the channels they have access to. Transactions belonging to the other channels will be invisible and inaccessible to them.

3.2. Supporting confidentiality on blockchain

Public blockchain systems such as Bitcoin [13], and Ethereum [2] follow a completely public approach: Every transaction is stored on the shared ledger, and basically every node verifies / replays every transaction leading to a high degree of enforceability. However, the balances of each account are publicly available.

A limited degree of privacy is achieved by using pseudonyms in form of blockchain addresses (hashed public keys) rather than real names. Ethereum [2] provides support for custom transactions in form of Turing Complete smart contract code. Transactions are basically replayed on each node of the network. If transaction output is dependent on data, such data must be available for all nodes in the unencrypted form either as on-chain data or as transaction input. This approach results in a high degree of proactive online enforceability but has a substantial toll on confidentiality. If a lower degree of enforcement is acceptable, off-chain enforcement can be used, where participants with data access (encrypted-on-chain or off-chain) check transactions in form of a distributed oracle [14], and only the execution of this protocol is on-chain [15].

Permissioned blockchains support a higher degree of privacy and confidentiality since access to the blockchain can be restricted. A well-known permissioned blockchain platform is Hyperledger Fabric [16]. It does not require a replay of transactions on all nodes. Instead, the sets of participants who have to verify transactions are defined via endorsement policies. To enhance confidentiality, Hyperledger Fabric supports two additional building blocks: channels and private data collections. As previously mentioned, channels are logically separate blockchains that are shared between a subset of participants. Since transactions are bound to a specific channel, cross-channel transactions are not supported. This results in a limited degree of enforceability as the correctness of real-world transactions with input data from multiple channels cannot be verified via smart contract code. Private data collections are an off-chain storage that is governed by the blockchain. Such data can be used within transactions, and the outcome of a transaction can be checked by participants who have access to the input data. However, not being on-chain data has a penalty on availability.

Zero-knowledge Proofs (ZKPs) [17] and in the blockchain context most relevant, their non-interactive version [18] allow a prover to prove some assertions without revealing any additional information. Without ZKPs, enforceability and confidentiality are conflicting requirements, and a balancing of the forces is required [15]. When ZKP are used for implementations, such conflicts can be resolved. However, the application of ZKP should be carefully planned and employed based on well-defined requirements. We argue that for business processes targeting blockchains, the requirements on enforceability and confidentiality should be explicitly defined in process models.

3.3. SecBPMN2 modeling language

SecBPMN2 [19] extends BPMN 2.0 with security requirements. We chose SecBPMN2 as the baseline for SecBPMN2BC since it is based on a widely known standard and includes a rich set of security requirements.

In the following, security requirements of SecBPMN2 are described, while the first part of Table 1 shows the corresponding annotations. Security annotations change their semantics based on the BPMN elements they are connected to. We, therefore, state the definition for each element they can be connected. To differentiate annotations' semantics based on the linked BPMN

Table 1
Graphical annotations of SecBPMN2BC-ML.

	Auditability		Separation of duties		Privity – public		Enforceability of the control flow
	Authenticity		Bind of duties		Privity – static		Enforceability of decisions – public
	Availability		Non Delegation		Privity – private		Enforceability of decisions – private
	Integrity		Privacy		Privity – strong dynamic		Enforceability of decisions – user defined
	Non Repudiation				Privity – weak dynamic		On-chain (To be used on P2 and P3)

element, we add a suffix (Act – activity, DO – data object, MF – message flow) to their names.

Auditability. It comes in three variants: AuditabilityAct, AuditabilityDO, and AuditabilityMF. They specify that it should be possible to keep track of all the actions performed when, respectively, a task is executed, a data object is accessed, and a message flow is used.

Authenticity. It comes in two variants: AuthenticityAct imposes that the identity of the users that are executing a task is verified. AuthenticityDO indicates that it should be possible to prove that a data object is genuine, i.e., that the data object was not modified by unauthorized parties.

Availability. It comes in three variants: AvailabilityAct indicates that a task should be ready for execution. AvailabilityDO indicates that a data object should be available when required. AvaliabilityMF indicates that it should always be possible to send a message.

Integrity. It comes in three variants: IntegrityAct, IntegrityDO and IntegrityMF. They specify, respectively, that a task, a data object, or a message should be protected from intentional corruption.

Non-Repudiation. It comes in two variants: NonRepudiationAct indicates that the execution of a task should be provable. NonRepudiationMF specifies that the sending of a message should be verifiable.

Separation of duties. It requires two or more different entities to be responsible for the completion of a task or set of related activities. It is linked to two pools, and it specifies that an entity cannot play at the same time the roles identified by the two pools.

Binding of duties. It requires the same entity to be responsible for the completion of a set of related activities. It is linked to two pools, and it specifies that they must be played by the same entity.

Non-delegation. It specifies that a set of actions must be executed only by the users assigned to that set. NonDelegationAct specifies that is not possible to assign part or the whole task to any other participant.

Privacy. It comes in two variants: privacyAct specifies that a task should be compliant with privacy legislation, and it should let users to control their own data. privacyDO is similar to the former one, but is targeted to a specific data object.

3.4. Smart contracts

N. Szabo coined the term *smart contract* in the 1990s in [1]. A smart contract is the counterpart of a traditional contract enforced by hardware and software. Szabo introduced the design goals of observability, online enforceability, and privity for smart contracts. *Observability* describes the possibility of each participant to observe each other's performance. It can be natively supported by blockchain technology. *Online enforceability*

can be proactive or reactive. Proactive online enforceability aims at making non-contractual behavior unfeasible. In contrast, reactive online enforceability is achieved by embedding the smart contract into the society (e.g., laws, courts, and executive organs). *Privity* aims at limiting the spread of knowledge and control to participants with a contractual need-to know. Szabo describes this as a generalization of the legal term privity, which requires that a contract should not define rights or obligations of third parties. There are numerous approaches for modeling smart contracts such as [20–24]. [20,23] aims at supporting reactive enforceability. [21,24] aims at connecting legal contracts with their electronic counterparts. Many smart contracts can be modeled in form of inter-organizational business processes [22]. Our approach is based on business process modeling and aims at providing extended modeling support for security requirements and on the smart-contract specific requirements on privity and proactive enforceability.

3.4.1. Privity

Traditionally, data objects reside in one pool and are assumed to be managed by the process engine of the organization owning the pool. This engine can potentially employ task based-access control [25], only granting access to actors when they are entitled to execute a task accessing the data object at runtime. This perspective changes when data objects are stored on-chain, as basically, every node of the blockchain has access to all on-chain data. To limit read access, additional techniques such as encryption or channels on private blockchains or off-chain storage need to be applied. However, this can limit the ability of the blockchain system to validate transactions based on data [15]. Therefore, we argue that modelers should be able to explicitly express read-access constraints for data objects.

One possibility for defining read-access control for data objects is the usage of the original SecBPMN2 confidentiality annotation (not introduced in this article for space limitations). It allows to provide a static list of participants for read and write access of each data object. However, with this approach, dynamic access restrictions cannot be defined. An alternative would be to define access restrictions using Role-based Access Control (RBAC) [26]. Such access control rules would need to be defined over the state of the process in order to respect the dynamic behavior. We consider the definition of potentially complex rules as a burden for modelers. Additionally, defining access control rules separately from data- and control flow may lead to ill-defined models and maintenance issues.

We, therefore, model read access requirements on top of privity spheres [15]. Privity spheres were introduced for the characterization of different patterns for the implementation of dataflows on blockchain. In [15] the privity spheres global, static, weak-dynamic and strong-dynamic were introduced. The work in [15] uses the term privity rather than confidentiality to emphasize on the application in smart contracts. See discussion in Section 3.4 for details.

The most restrictive sphere is strong-dynamic. During process execution, a participant is in the strong-dynamic sphere of a data

object last written by some writer if it is certain that she <u>will</u> execute a task reading the data value written by the writer. A slight relaxation is the weak-dynamic sphere: during process execution, a participant is in the weak-dynamic sphere of some data object last written by some writer if she <u>can</u> execute a task reading the data value written by the writer. A participant is in the static sphere of a data object if she owns any tasks accessing the data object. A participant is in the global sphere of a data object if she is a participant of the process. Privity Spheres are defined on an abstract process model. In the case of BPMN a participant is in the global sphere if she takes part in the collaboration diagram containing the data object.

Example The process in Fig. 3 shows a process between a citizen (C), a municipality (M) containing a resident registration office (M.R) and a mayor's office (M.M) and a timber-yard containing a mid term planner (T.M) and a road worker (T.R). The global sphere of the citizen's data object *CD* is {*C*, *M.R*, *M.M*, *T.M*, *T.R*}. The static sphere of *CD* is {*C*, *M.R*, *M.M* }. The weak-dynamic sphere at the task *ReportClaim* of *CD* is {*M.R*, *M.M*}, since both *M.R* and *M.M* can possibly read the data value written at *ReportClaim*. However, the strong-dynamic sphere at the task *ReportClaim* is {*M.R*} because, at that point, it is only certain that *M.R* will read the data.

For assessing privity properties of different implementation patterns, the work in [15] defined safety classes based on the privity spheres. E.g., an implementation is static-safe if only participants owning tasks reading or writing some data objects have read access to these data objects.

3.4.2. Enforceability

We focus here on proactive online enforceability aiming in making non-contractual behavior infeasible. Smart contract code can provide strong support for proactive online enforceability if the data required for some transactions are entirely stored on-chain or provided as input. Existing approaches for business processes on blockchain, such as [6,8] focus on guaranteeing that only permissible traces of executions are possible. E.g., an out-of-order execution is effectively prevented by smart contract code that keeps track of the state of the process instance.

However, checking that only the prescribed execution traces are admissible does not yet guarantee that control flow decisions are taken faithfully. The required degree of online enforceability of decisions depends on the use case. E.g., whether some buyer accepts an offer should solely depend on the buyer, while a check on whether the buyer has sufficient funds to actually pay for the ordered products should be backed by the blockchain. In other cases, it can be sufficient that other participants validate decisions rather than the entire blockchain network. Therefore, we propose to explicitly model requirements on enforceability for balancing potentially conflicting requirements and to select proper technologies for implementations.

4. SecBPMN2BC-ML modeling language

SecBPMN2BC-ML modeling language is based on collaboration diagrams of BPMN 2.0 and it integrates parts of the security annotations of SecBPMN2 with privity spheres and enforceability requirements. In particular, we adopted all annotations of SecBPMN2 but Accountability and Confidentiality. We merged Accountability with Auditability: blockchains structurally enforce Accountability since all traces of actions executed are available to all accounts. Confidentiality security annotations are substituted by annotations of privity spheres, which are far more expressive. We first present the graphical extensions in Section 4.1 and part of the meta-model Section 4.2.

4.1. Graphical annotations in SecBPMN2BC-ML

Table 1 shows the graphical annotations of Sec-BPMN2BC-ML. These annotations can be applied to specific BPMN flow elements We have marked all new annotations in bold in Table 1. We created the new graphical annotations following guidelines of Moody [27]. The On-Chain annotation is used in phases P2 and P3 of the proposed workflow.

4.1.1. Modeling privity requirements

The original privity spheres described in Section 3.4.1 were used for the characterization of implementation patterns using safeness classes. We build on top of privity spheres and use them for prescribing the minimal privity requirements of data objects or messages. Therefore, data objects and messages of a process model are annotated with privity spheres. An implementation of such a process model fulfills the privity requirements if, for every annotated data object or message *d* and every possible execution trace, only members of the required sphere of *d* can read *d*.

In addition, we have adopted and extended the set of spheres to better fit various requirements: public, private, weak-dynamic, strong-dynamic. The new public sphere allows the entire public to access all instances of data objects or messages. Accordingly, read access is not restricted. Potential implementations might store the data on a public blockchain or store it on a public web page. In order to avoid confusion, we have renamed the original global sphere to private. The original global sphere allows all participants of the process to read the data. We now use the new label private because this matches well with private blockchains, where the blockchain is likely only shared between the participants of the process. The static-, weak-dynamic- and strong-dynamic- spheres were not touched (see Section 3.4.1 for their definitions). In SecBPMN2BC-ML, privity requirements are represented by graphical annotations, which can be attached to data objects and messages. All graphical annotations on privity are shown in Table 1.

We do not explicitly model access control policies for writes, as an implementation of task-based access control [25] can easily be achieved via smart contract code. It is also worth noting that in SecBPMN2BC-ML we kept the privacy annotation, as its semantics does not overlap with privity annotations. In particular, we keep its original semantics, i.e., that the targeted element should be protected by privacy laws. As far as our knowledge goes, blockchain systems architecturally contradict some privacy requirements. Requirements such as the right to be forgotten, specified in GDPR [28], contradicts with the immutability of blockchain, one of the major structural properties of this technology. For these reasons we use the privacy annotation to specify data objects and activities that must comply with privacy requirements, which cannot be satisfied using blockchain and, therefore, cannot be stored or executed on blockchain.

4.1.2. Modeling enforceability requirements

SecBPMN2BC-ML allows to define requirements on enforceability for the correct execution of the control flow and for the correct execution of decisions. Enforceability requirements are expressed via a train symbol. The visual metaphor here is that a process with enforceability requirements cannot leave the prescribed control flow (its rails). Enforceability requirements on the control flow are expressed by a plain train symbol. It can be assigned to processes, pools, and subprocesses. This annotation requires that the correctness of the control flow is proactively enforced by the developed system. It must be unfeasible not to comply with the prescribed control flow.

The enforceability requirements on decisions are expressed via a train symbol with an additional circle defining the required level of enforcement. This annotation can be attached

Table 2
Connectable elements for enforceability and privity.

SecBPMN2BC element	Connectable elements
EnforceCF	Definitions, SubProcess, Pool
EnforceGW	Inclusive gateways, Exclusive gateways, Complex gateway
Privity	Data Object, Message

to conditional split gateways. If the gateway is connected to a business rule task, then the enforcement also holds for the business rule task. Otherwise, it only applies to the condition of the gateway itself. The required level of enforcement is defined by sets of verifiers. SecBPMN2BC-ML provides graphical annotations for public, private, and user-defined sets. The public set requires that decisions are verified by a set of nodes that is substantially larger than the set of participants of the process, i.e., this requirement can be fulfilled by smart contract code on a public blockchain executing the decision. The private set requires that all participants of the process verify the correctness of the decision, i.e., this requirement can naturally be fulfilled by smart contract code on a private blockchain executing the decision. Finally, the user-defined annotation allows the modeler to provide a set of participants who have to verify the decision. All graphical annotations on enforceability are shown in Table 1.

4.1.3. On-chain/off-chain annotations

In addition to modeling privity and enforceability requirements, SecBPMN2BC-ML also allows to explicitly define which instances of process elements should be stored or executed on- or off-chain. The on-chain storage or execution is graphically expressed by a chain symbol (see Table 1). Following our design process described in Section 2 this property is not a user input of the modeler in P1 or P2. Instead, it is derived in P2. However, since we allow refinements in P3, the user can change the property values when required.

4.2. SecBPMN2BC-ML metamodel

Table 2 shows the legal security associations for the new elements introduced in this article. EnforceCF can be connected to "Definitions" which represents in BPMN 2.0 the basic elements, i.e., the whole set of processes defined in the diagram; "SubProcess", since only a part of the flow of the process may need to be enforced; and "Pool", which identifies the part of the business process executed by a participant that has to be enforced. EnforceGW annotations can be connected to decision-based gateways since the annotations target the verifiability of such decisions. Privity annotations can be linked to data objects and messages since they specify the access limitations.

When designing a blockchain-based application, care should be taken in identifying which portions of the process governing the application should be carried out on-chain and which should instead rely on traditional, off-chain tools and techniques. This decision affects the following elements in business processes.

Structure. The structure of the process specifies the sequence of BPMN elements. If the structure is defined *on-chain*, it can be encoded with a smart contract that will keep track of the execution of the process. To this aim, the smart contract will emit events notifying when activities should be executed, and it will receive notifications when they are started or completed. If the structure is defined *off-chain*, the execution of the process will be managed off-chain (e.g., with a BPMS).

Tasks. Tasks represent a single unit of work that should be executed. They can be divided into fully automated, semi-automated, and manual tasks. *Generic*, *script*, *service*, *business rule*, *send* and *receive* tasks can be fully automated. *User* tasks can be semi-automated, whereas *manual* tasks can only be manual. If fully automated tasks are executed *on-chain*, it is possible to specify in the smart contract the instructions required for their execution. In this way, we can be certain that the execution of the task will be performed exactly as expected. If fully automated tasks are executed *off-chain*, they will be executed by external software (e.g., by a web service). It is worth noting that semi-automated and manual tasks require the intervention of external resources, and they cannot be executed on-chain.

Data. Data being manipulated by the process are represented as data objects or as messages. When stored *on-chain*, the process relies on the blockchain to validate, store, and retrieve them. In particular, the (read[2]) and write operations, as well as the logic to ensure the correctness of the data, are governed by smart contract code being executed on-chain. If the data are stored *off-chain*, both the storage and validation of such data must be managed by external applications.

4.2.1. Blockchain-specific properties

To specify these concepts, we introduced the following blockchain-specific properties in SecBPMN2BC-ML.

OnChainModel:boolean for process definition, pools, and subprocess activities. It specifies whether the execution logic, i.e., the logic that enforces control flow dependencies among activities and keeps track of when activities are executed, will be handled on-chain via smart contracts (if its value is `true`), or that the execution logic is handled off-chain (if its value is `false`). If `OnChainModel` is set to `true`, the On-chain graphical notation will be shown on the corresponding diagram element.

OnChainExecution:boolean for tasks. It specifies whether that a task will be executed on-chain by a smart contract (if its value is `true`), or it will be automated off-chain (if its value is `false`). For user and manual tasks (i.e., tasks that cannot be automated) this property can only be set to `false`. If `OnChainExecution` is set to `true`, the On-chain annotation will be shown on the corresponding task.

OnChainData:{unencrypted,encrypted,digest,none} for each data object or message. It specifies whether a data object will be entirely stored and validated on-chain (if its value is `unencrypted`), if the data will be stored on-chain in an encrypted form (if its value is `encrypted`), if the digest of the data (i.e., the result of a hash function) will be stored on-chain and the actual data off-chain (if its value is `digest`), or if the data will be entirely stored off-chain (if its value is `none`). If `OnChainData` is not set to `none`, the On-chain graphical notation will be shown on the corresponding data object or message.

BlockchainType:{public,private} for process definitions. It specifies whether the on-chain portion of the process will be executed on a public or private blockchain.

---

[2] In most blockchain systems, smart contract code can fully control write access and assist read access. The blockchain can guarantee data availability, but smart contract code cannot restrict read access of peers who are in possession of a copy of the ledger.
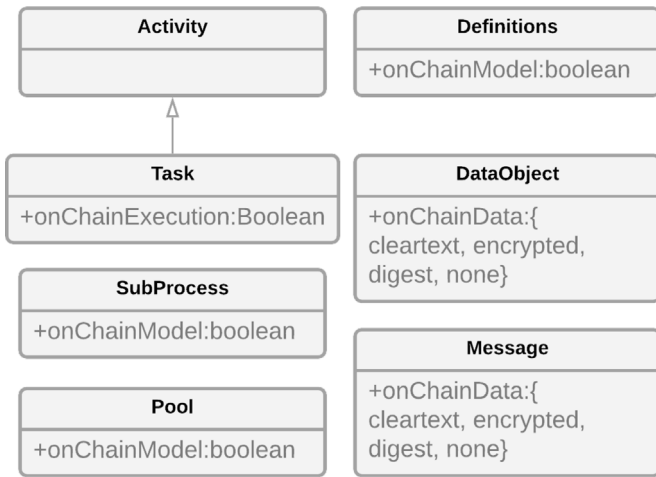
Fig. 2. Extension of BPMN 2.0.

Fig. 3. Example of SecBPMN2BC-ML diagram for a road misconstruction claim.

Those properties hold for the process elements where they are specified and to all the child elements unless the same properties are set differently for those elements. It is worth noting that, since those properties are related to the way the process is implemented, they are closer to software development than the security annotations introduced in the previous section and are used starting from phase P2 of the proposed workflow.

Fig. 2 defines the extension of BPMN 2.0 that is needed for the identification of part of the process to be executed on-chain and implemented as smart contract code. More information on these properties will be provided in following sections.

4.3. SecBPMN2BC-ML in action

Fig. 3 shows an example of a diagram created with Sec-BPMN2BC-ML, that represents the road misconstruction smart contract being mentioned in Section 1. Such a diagram is derived from a real case study we used for the validation of the method. In this example, some security annotations of SecBPMN2BC-ML are used. On the top of the diagram a dashed rectangle specifies an area that contains annotations to be applied to the whole diagram. In this case an Enforceability annotation specifies that it should be impossible for a process execution to violate the control flow dependencies in the model.

A Separation of duty annotation specifies that citizen and municipality should not be played by the same person. Citizen's personal data data object is linked to Integrity annotations to prevent data object from being tampered with by malicious users. The same data object is connected to Strong-dynamic privity sphere annotation, to specify that the data access must be restricted only to the users that will execute tasks that use it. Claim restriction and General renovation plan data objects are linked to Public privity sphere annotation since they will be available to anyone. Urgent? and Local? exclusive gateways of municipality process are linked with a Public enforceability annotations to specify that the correctness of the decision must be checked publicly.

5. Enforcement of security concepts using blockchain technology

In this section we analyze to what degree the security requirements expressed in SecBPMN2BC-ML models can be fulfilled by using blockchain based implementations. In particular, we base our analysis on the metamodel in Section 4.2, and analyze to what degree the requirements can be fulfilled, when different values
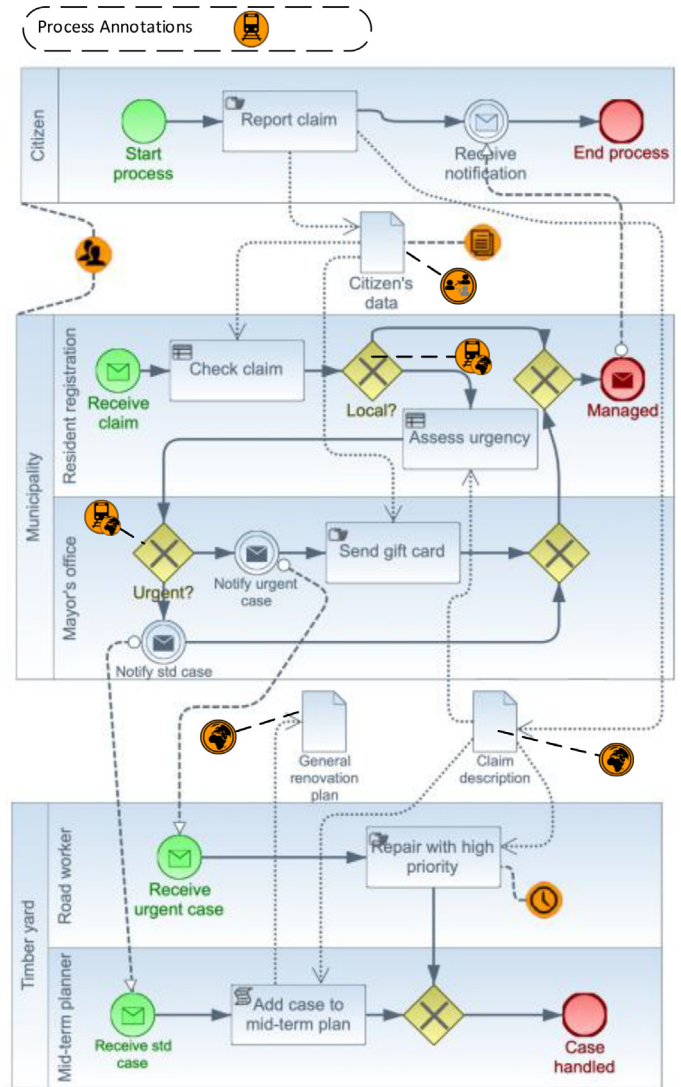
for the blockchain-specific properties are used. For each security requirement, multiple sets of value assignments are identified for the blockchain-specific properties. Then, each set is labeled based on the level of enforcement that can be achieved using the blockchain: *native*, when the requirement can be fully enforced by the blockchain using that set of property values, *possible*, when it can only partially be enforced, and *no enf.*, when the blockchain provides no help in enforcing the requirement. The rules to derive those labeled sets, together with the algorithms discussed in Section 6, constitute SecBPMN2BC-Tools.

5.1. SecBPMN2 security annotations

Table 3 summarizes the blockchain enforcement capabilities for the security annotations inherited from SecBPMN2. Due to space constraints, only the rules that are pertinent to the running example are discussed here. The reader should refer to the technical report provided with this paper [29] for the complete set of rules and additional details. In the following discussion, we assume that there exists either a static [9] or dynamic mapping [30] between process participants and blockchain identities.

Table 3

Blockchain enforcement rules for SecBPMN2 security annotations. Please refer to 4.2.1 for the meaning of the properties and their values. Note that the value *any* indicates that the associated property can assume any of the values it supports.

	OnChain model	OnChain execution	Output label		OnChain model	OnChain data	Output label		OnChain model	OnChain data	Output label
AuthenticityAct	any	true	native	**AuthenticityDO**	any	unencrypted	native				
	any	false	no enf.		any	encrypted	native				
					any	digest	no enf.				
					any	none	no enf.				
AuditabilityAct	any	true	native	**AuditabilityDO**	any	unencrypted	native	**AuditabilityMF**	any	unencrypted	native
	any	false	possible		any	encrypted	native		any	encrypted	native
					any	digest	no enf.		any	digest	possible
					any	none	no enf.		any	none	no enf.
AvailabilityAct	any	true	native	**AvailabilityDO**	any	unencrypted	native	**AvailabilityMF**	any	unencrypted	native
	any	false	no enf.		any	encrypted	native		any	encrypted	native
					any	digest	no enf.		any	digest	possible
					any	none	no enf.		any	none	possible
IntegrityAct	any	true	native	**IntegrityDO**	any	unencrypted	native	**IntegrityMF**	any	unencrypted	native
	any	false	possible		any	encrypted	native		any	encrypted	native
					any	digest	native		any	digest	native
					any	none	no enf.		any	none	no enf.
NonRepAct	any	true	native					**NonRepMF**	any	unencrypted	native
	any	false	possible						any	encrypted	native
									any	digest	native
									any	none	possible
NonDelAct	any	true	native								
	any	false	possible								
BoDPool (Act)	true	true	native	**BoDPool (DO)**	true	unencrypted	native				
	true	false	possible		true	encrypted	native				
	false	any	no enf.		true	digest	native				
					true	none	possible				
					false	any	no enf.				
SoDPool (Act)	true	true	native	**SoDPool (DO)**	true	unencrypted	native				
	true	false	possible		true	encrypted	native				
	false	any	no enf.		true	digest	native				
					true	none	possible				
					false	any	no enf.				

5.1.1. Availability of tasks

If `OnChainExecution` is set to `true` for the task linked to the annotation, then this property is *natively* enforced by the blockchain. Indeed, the architecture of blockchain ensures high level of availability via full replication [31]. If `OnChainExecution` is set to `false`, the enforcement of this property is mandated to the participant in charge of executing that task. Thus, this property is *not enforced* by the blockchain at all.

5.1.2. Integrity of data objects

If `OnChainData` is set to `unencrypted`, `encrypted` or `digest` for the linked data object, then this property is *natively* enforced by the blockchain. When data are on-chain, the distributed nature of the blockchain prevents them from becoming corrupted [32]. If `OnChainData` is set to `none`, the enforcement of this property is mandated to the entity that manages the external storage where the data reside. Thus, this property is *not enforced* by the blockchain at all.

5.1.3. Separation of duties

If `OnChainModel` is set to `true`, then this property is *natively* enforced by the blockchain for all the activities, placed inside one of the pools targeted by the annotation, whose `OnChainExecution` property is set to `true`. This property is also *natively* enforced for all data objects, linked to the activities placed inside the pools targeted by the annotation, whose `OnChainData` property is set to `unencrypted`, `encrypted` or `digest`. The activities are identified in Table 3 with SODPool(Act) while the data objects with SODPool(DO). Being the process logic stored on chain, when an agent triggers the execution of an on-chain task or tries to manipulate a data object, access control mechanisms can determine if the address of that agent differs from the one who previously interacted with the process portions belonging to the other pools, and only in that case authorize it [31]. Conversely, for activities placed inside one of the pools targeted by the annotation, if `OnChainExecution` property of

those tasks is set to `false`, the blockchain can only prevent notifications from unauthorized nodes to be accepted. Organizations have to implement their own access control mechanisms [33]. The same holds for data objects linked to tasks placed inside targeted pools, if `OnChainData` property of those data objects is set to `none`. Thus, with the blockchain alone, we can *possibly* achieve separation of duties.

If `OnChainModel` is set to `false`, then the enforcement of this property is *not enforced* by the blockchain at all. Indeed, being the process logic kept off-chain, invocations of activities and operations on data objects are considered independent from each other.

5.2. Privity spheres

Table 4 provides an overview of all constraints induced by the privity requirements on annotated data objects, based on the properties `BlockchainType` and `OnChainData`. Only the public privity sphere is *natively* supported on public blockchains using unencrypted on-chain data, and all other privity levels are *violated*. If encryption is used, all privity spheres can *possibly* be realized on public blockchains. However, this is not natively supported, as smart contract code cannot access the data, and additional key exchange between the participants is required [34]. In case of the strong-dynamic sphere, the key exchange needs to be dynamic (α) [15]. Private blockchains can natively support the public and private privity-sphere. However, the static-, weak-dynamic- and strong-dynamic-privity spheres require additional means such as channels to fulfill this requirement. Since current private blockchain systems such as HyperLedger Fabric (HLF) do not support cross-channel transactions [16], an implementation is not straight forward. Therefore, we assign *possible* for the enforcement level for the static-, weak-dynamic-, and strong-dynamic-spheres. The strong dynamic sphere (β) is especially channeling for private blockchains with channels because the same data might need to be written to multiple channels. For

Table 4

Constraints on enforcement of privity and enforceability requirements. Please refer to 4.2.1 for the meaning of the properties and their values.

BlockchainType OnChainData	Public Unencrypted	Public Encrypted	Private (Un)encrypted	Any Digest	Any None
Privity constraint on data objects					
Public	native	(possible)	(native)	(possible)	no enf.
Private	violated	possible	native	possible	no enf.
Static	violated	possible	possible	possible	no enf.
Weak-Dynamic	violated	possible	possible	possible	no enf.
Strong-Dynamic	violated	possible$^\alpha$	possible$^\beta$	possible	no enf.
Privity constraint on messages					
Public	native	possible	(native)	(possible)	no enf.
Private	violated	possible	native	possible	no enf.
Static	violated	native	possible	possible	no enf.
Weak-Dynamic	violated	native	possible	possible	no enf.
Strong-Dynamic	violated	possible	possible	possible	no enf.
Enforceability constraints on decisions					
Public	native	violated	possible	possible	no enf.
Private	(native)	possible	native/possible$^\gamma$	possible	no enf.
User-Defined	(native)	possible	native/possible$^\gamma$	possible	no enf.

example, in Fig. 3 the citizen's data cannot be written to a channel containing the Mayor, since it would allow an indiscriminate access to the data object, violating the strong-dynamic privity sphere. Instead, two different channels need to be used, and the second write must be delayed until the decision on the gateway "urgent?" of the municipality is taken.

Finally, if only a digest is stored on-chain, all privity levels on data can *possibly* be supported. However, the actual data transfer is out of scope of the blockchain. The case that neither data nor digest is stored on-chain is out of scope of the blockchain, and *no* blockchain-based *enforcement* is possible.

The restrictions of privity requirements on linked messages are shown in the second part of Table 4. They are similar to the ones for data objects except for encrypted messages. Blockchains come with built-in support for encrypted data exchange between two participants (e.g., Diffie Hellman Key Exchange). The downside is that smart contracts cannot access the message content. This allows native support for binary messages in the case of the static and weak-dynamic privity spheres. However, in the case of the strong-dynamic privity sphere, writing a particular message to the chain must be delayed until the receiver will certainly consume the message, leading to only possible enforcement. Global and public spheres allow a message to be read by multiple participants. These cases, therefore, match the case of data objects.

A note on on-chain tasks: if tasks are executed on-chain also all their accessed data objects or messages must either be already stored on-chain in unencrypted form, or appear as unencrypted parameters of smart contract transactions. On public blockchains this is only compatible with the public privity sphere of data objects or messages. If `BlockchainType` is set to `private` and `OnChainExecution` is set to `true`, then the public and private spheres can be natively supported. The more restrictive spheres are problematic as they require cross channel transactions, if data inputs from different channels are needed for some on-chain tasks.

5.3. Enforceability

5.3.1. Enforceability of the control flow

If `OnChainModel` is set to `true` for the process portion where the annotation holds, and `OnChainExecution` is set to `true` for all the tasks on that portion, then this property is enforced by the blockchain itself, as the blockchain forces the process to strictly adhere to the control flow dependencies specified in that

portion. Thus, with the blockchain alone, we can *natively* achieve enforceability.

If `OnChainExecution` is set to `false` for at least one of the tasks on that portion, then the blockchain cannot enforce if those tasks are executed when specified in the process. Thus, with the blockchain alone, we can only *possibly* achieve enforceability.

If `OnChainModel` is set to `false` for the process portion where the annotation holds, the blockchain provides no help in enforcing this property. Thus, with the blockchain alone, we can achieve *no enforceability*.

5.3.2. Enforceability of decisions

Process models can contain requirements on the enforceability of decisions by linking an enforceability annotation to the corresponding gateway. The third part of Table 4 provides an overview of all constraints, which are induced by the enforceability requirements of the data objects accessed by the task whose output is used by the linked gateway. *Native* support for public enforceability can be provided by public blockchains where the input data of the decision is stored in plain text on-chain. If only the digest of the data is stored on-chain, public enforceability is *possible* using advanced cryptographic approaches such as ZKP [17] which need to be validated on-chain. Private blockchains can *possibly* allow public enforceability if they include a substantial amount of nodes, which are not under control of the participants of the process.

Private and user-defined enforceability can be achieved *natively* on private blockchains. However, when privity constraints on data objects are enforced by using channels, additional constraints apply. Channels on private blockchains such as HLF are isolated blockchains with a subset of participants. This also implies that transactions spanning over multiple channels are an open issue. Therefore, privity requirements beyond the private privity sphere can result in the need to read data objects from different channels (see Discussion in Section 5.2). Enforcing decisions over data objects with privity constraints can therefore require cross-channel transactions. Such transactions need to be simulated with approaches such as voting protocols or ZKP. As a consequence, we assign *possible* (γ) to private and user-defined enforceability if any referenced data object has a privity requirement beyond the private sphere.

On public blockchains, private and user-defined enforceability can possibly be achieved if only the digest of the input data is stored on-chain or data is encrypted. In these cases, either ZKP or voting protocols [15] can be employed. In contrast to the public

sphere, it is sufficient to verify the ZKP off-chain. If neither data nor the digest is stored on-chain, *no enforcement* can be achieved by the blockchain alone.

6. Model verification and enhancement

Section 5 explained how different sets of value assignments for blockchain-specific properties affect the enforcement of security annotations for the associated process element. Thus, one may be tempted to choose property value assignments belonging to the set that provides maximum enforcement of that annotation. However, properties that hold for the same process element may influence multiple annotations, possibly in a conflicting way. Thus, one should pay attention in identifying, for each process element, a property value assignment that is compatible with all the annotations in the model, i.e., that is present in all the sets derived from the annotations that hold for that element.

6.1. Algorithm

To support phases P1, P2 and P3 of the proposed workflow, we defined an algorithm that (i) identifies, for each process element, a set of property values satisfying and possibly enforcing the security and privacy requirements specified by the annotations, (ii) verifies if current property value assignments are compatible with each other, and (iii) detects conflicting annotations in a SecBPMN2BC-ML model.

Similarly to what is typically done in many Business Process Management (BPM) tools, this algorithm relies on a tree structure to represent the process model. The root element represents the process definitions, and the `blockchainType` and `onChainModel` properties can be set for that element. The child elements of the process definitions are pools, data objects, and messages. Data objects and messages are leaf elements, and the `onChainData` property can be set for those elements. Pools are intermediate elements, and the `onChainModel` property can be set for those elements. The child elements of a pool are tasks, subprocess activities, gateways, and events. Tasks are leaf elements, and the `onChainExecution` property can be set for those elements. Gateways and events are leaf elements as well, and no blockchain-specific property can be set for those elements. Subprocess activities are intermediate elements, and the `onChainModel` property can be set for those elements. As for pools, the child elements of a subprocess activity are tasks, other subprocess activities, gateways, and events.

The scope of blockchain-specific property values holds for the element where that property is set, and for all its children and grandchildren. However, if the same property can be set for one of the child elements, then the old value is no longer propagated, and the new value holds for that child element and its children. For example, if `onChainModel` is set to `true` for the process definitions and to false for the Citizen and Municipality pools, then the structure of the process portion outside the pools (i.e., the message flows representing the process choreography) will be put on-chain, whereas the structure of the process within pools (i.e., the internal processes) will stay off-chain. We say that a property value assignment *directly* holds for a process element if the property belongs to that element. We say that a property value assignment *indirectly* holds for an element if that property belongs to an ancestor of that element, rather than to the element itself. The algorithm is composed of three main steps. Step 1 builds for each element in the process tree a set of property value combinations that hold locally for that element. Step 2 scans the process tree bottom-up to remove, for each element, combinations incompatible with its children. Step 3 scans the process tree top-down to identify, for each element, the combination that

Table 5
Property value assignments that hold locally for *Assess urgency*.

OnChainExecution	BlockchainType	OnChainModel	OutputLabel
false	public	true	no enf.
false	private	true	no enf.
false	public	false	no enf.
false	private	false	no enf.
true	public	true	native
true	private	true	native
true	public	false	no enf.
true	private	false	no enf.

maximizes the enforcement of the security constraints holding for that element. Also, it detects if conflicting security requirements are present in the model. Due to space limitations, in the next sections we will only briefly discuss these steps. To address this shortcoming, we have also created a technical report [29] to provide to the interested reader all the inner details on how the algorithm works.

6.1.1. Deriving local combinations

To identify, for each process element, a set of property value combinations that hold locally for that element, the following steps are performed:

1. For each security annotation, the corresponding rule discussed in Section 5 is evaluated, and a set of combinations $S_R(e_i)$ is identified for each process element e_i affected by the rule. Each combinations of $S_R(e_i) = [C_j]$, where $C_j = \langle [P_k], s \rangle$, is a combination of property values $P_k = \langle property, value \rangle$ which directly or indirectly hold for e_i and rule R. Value s in C_J determines to which extent the rule R is enforced by the blockchain, where $s = \{native, possible, no\_enf.\}$, and $native > possible > no\_enf$. If the same element e_i is subject to multiple rules R, all sets of e_i are combined to obtain the most stringent requirements. Given $S_{R1}(e_i)$ and $S_{R2}(e_i)$, with $C_j = \langle [P_k], s \rangle$ in $S_{R1}(e_i)$ and $C_w = \langle [P_k], t \rangle$, if C_w exists in $S_{R2}(e_i)$ (note that the combinations P_k must be the same in C_j and C_w) where $s >= t$, then C_w is added to the resulting set. Otherwise, i.e., if $s < t$, then C_j is added to the resulting set.
   If e_i is not subject to any rule, then $S(e_i)$ is composed by every possible combination of all properties that may directly or indirectly hold for that element.
2. If no value was previously assigned to any of the properties of $S(e_i)$ before the algorithm was run, this step is skipped. Otherwise, for each element e_i, the set of property values $A(e_i) = [P_l]$ already assigned to the properties of e_i is computed. Then, all the combinations incompatible with $A(e_i)$ are removed from $S(e_i)$. More precisely, for each P_l in $A(e_i)$, for each $[C_j]$ in $S(e_i)$, if $P_l \notin [P_k]$ of C_j, then C_j is removed from $S(e_i)$.

Table 5 shows the outcome of this step for task *Assess urgency* in the running example. In particular, the set $S(Assess\_urgency)$ is obtained by merging the sets $S_{SoDPool}(Assess\_urgency)$, $S_{PubPrivity}(Assess\_urgency)$, and $S_{EnforceCF}(Assess\_urgency)$.

6.1.2. Bottom-up scan

In the previous step, we identified the combinations that hold for each process element, without taking into account the tree structure of the process. However, it may happen that a combination derived for a parent element may not be *compatible* with any of the combinations derived from one of its child elements. In particular, a combination C_j is *compatible* with a set $S(e'_i)$ if at least a combination C'_j exists in $S(e'_i)$ such that, for each property value assignment $P'_k = \langle property', value' \rangle$, a property value assignment

$P_k = \langle property, value \rangle$ does not exists in C_j such that $property = property' \wedge value \neq value'$. For example, if the `onChainData` property for Citizen's data is set to `unencrypted` by the user, then the combination

`blockchainType:public` for the process definitions would be incompatible with all the combinations for Citizen's data.

Therefore, to take into account all the parent–child relations in the process tree, it is necessary to remove from each element the combinations that are incompatible with their child elements. To do so, Algorithm 1 is used.

Algorithm 1: propagateUp

---

input : SecBPMN2BC e_i: element
output: Set S_temp: admissible property value combinations for parent element
S_temp=newSet();
for *C_j in S[e_i]* **do**
  C_temp = newCombination();
  for *P_k in C_j.properties* **do**
    if *P_k.name not in e_i.properties* **then**
      C_temp.properties.add(P_k);
  S_temp.add(C_temp);
if *not e_i.isLeaf* **then**
  for *e_i' in e_i.children* **do**
    S_parent = newSet();
    S_local = newSet();
    for *C_child in propagateUp(e_i')* **do**
      C_parent = newCombination();
      C_local = newCombination();
      for *P_child in C_child.properties* **do**
        if *P_child not in e_i.properties* **then**
          C_parent.properties.add(P_child);
        C_local.properties.add(P_child);
      S_parent.add(C_parent);
      S_local.add(C_local);
    S[e_i] = constrain(S[e_i],S_local);
    S_temp = constrain(S_temp,S_parent);
return *S_temp*;

Starting from the root element e_i of the process tree, an empty set S_{temp} is created. Then, for each combination C_j in $S(e_i)$, a new combination S_{temp}, which contains only property values that indirectly hold for that element, is derived.

If e_i is not leaf element, two empty sets S_{parent} and S_{local} are created. Then, for each element e_i' that is a child of e_i, Algorithm 1 is recursively invoked and the following steps are repeated.

For each combination C_{child} in the set returned by that invocation, two combinations C_{parent} and C_{local} which contain only the property values in C_{child} that, respectively, indirectly and directly hold for e_i are derived and are added, respectively, to S_{parent} and S_{local}. Then, the combinations in $S(e_i)$ that are incompatible with S_{local} are removed. Similarly, the combinations in S_{temp} incompatible with S_{parent} are removed.

Eventually, S_{temp} is returned by the algorithm.

It is worth noting than, when executing this step, none of the combinations for an element e_i may be compatible with its children. In this case, an empty set is assigned to $S(e_i)$, indicating that a conflict exists. Such a conflict will be handled subsequently in the next and final step.

6.1.3. Top-down scan

In the previous step of the algorithm, for each process element, the combinations compatible with all security annotations – that is, the ones directly holding for that element and the ones holding for the other elements that have a parent–child relation – have been identified. However, it is worth noting that multiple combinations may be compatible with a process element, providing different levels of enforcement of the security annotations. Also,

when one of such combinations is selected, the combinations compatible with its child elements are restricted to the ones compatible with the combination selected for the parent. Finally, if no combinations exist for an element, then a conflict in the security annotations holding for that element exists. To select the combinations that provide the local maximum enforcement, Algorithm 2 is used.

Algorithm 2: propagateDown

---

input : SecBPMN2BC e_i: element
output: Boolean noConflicts: true if the model passes all constraints
noConflicts = true;
if *e_i.parent* **then**
  S[e_i]=constrain(S[e_i],S[e_i.parent]);
if *S[e_i].size = 0* **then**
  err.raise(e_i,'Conflict detected');
  noConflicts = false;
else
  C_best = getBestCombination(e_i);
  S[e_i] = newSet(C_best);
  assignValues(e_i);
  for *e_i' in e_i.children* **do**
    temp = propagateDown(e_i');
    if *noConflicts = true* **then**
      noConflicts = temp;

return *noConflicts*;

Starting from the root element e_i of the process tree, the boolean variable *noConflicts* is initially set to `true`, indicating that no conflict was detected so far. Then, if e_i has a parent element e_{parent}, the combinations in $S(e_i)$ that are incompatible with $S(e_{parent})$ are removed.

After that, if $S(e_i)$ is an empty set, a conflict is detected for e_i and *noConflicts* is set to `false`. Otherwise, the combination C_{best} that provides the maximum enforcement for e_i – that is, the one whose value s is the highest among the other combinations – is selected from $S(e_i)$, and the property values of e_i are set according to the ones in C_{best} that directly hold for e_i. Then, for each element e_i' that is a child of e_i, Algorithm 2 is recursively invoked, and its result is used to update *noConflicts* as long as that variable is `true`.

Eventually, *noConflicts* is returned by the algorithm.

6.2. Tool support

We created a software tool, named SecBPMN2BC-Tools,[3] as part of the SecBPMN2BC method, that implements the proposed algorithm. SecBPMN2BC-Tools was written in Java with Eclipse EMF SDK and STS-Tool libraries.[4] With this software it is possible to model and analyze SecBPMN2BC-ML models. In particular, SecBPMN2BC-Tools identifies and highlights conflicts when present. Furthermore, SecBPMN2BC-Tools identifies BPMN 2.0 elements that need to be executed and stored on blockchain in order to enforce the security and privacy requirements specified by the annotations. It, therefore, adds the On-chain annotation automatically.

Table 6 shows the property value assignments identified by SecBPMN2BC-Tools for the road misconstruction claim process, introduced in Section 4.2. In particular, SecBPMN2BC-Tools received in input the SecBPMN2BC-ML model shown in Fig. 3, after setting the `blockchainType` property to `public` and leaving the other properties unassigned.

Note that, being the target a public blockchain, only the digest of the citizen's data is stored. Indeed, storing this information on-chain in unencrypted form would allow anybody to access it, thus

---

3 https://github.com/MattiaSalnitri/SecBPMN2BC
4 See https://www.sts-tool.eu

Table 6
Property value assignments identified by SecBPMN2BC-Tools for the running example.

Process element	Type	Property	Decision	Reason
Process model	Control flow	onChainModel:true	On-chain	Control flow enforceability requirement
Citizen's data	Data object	onChainData:digest	On-chain	Strong-dynamic sphere and integrity requirement
Claim description	Data object	onChainData:unencrypted	On-chain	Public sphere requirement
General renovation plan	Data object	onChainData:unencrypted	On-chain	Public sphere requirement
Report claim	Task	onChainExecution:false	Off-chain	User tasks cannot be automated
Check claim	Task	onChainExecution:false	Off-chain	Strong-dynamic sphere on Citizen's data prevents on-chain execution
Assess urgency	Task	onChainExecution:true	On-chain	Public enforceability of decision on *urgent?* gateway
Send gift card	Task	onChainExecution:false	Off-chain	User tasks cannot be automated
Add case to mid-term plan	Task	onChainExecution:true	On-chain	Script tasks reading public data can be automated on-chain
Repair with high priority	Task	onChainExecution:false	Off-chain	Manual tasks cannot be automated

violating the strong dynamic sphere requirement. Consequently, task Check claim cannot be executed on-chain, as it needs to access the citizen's data. Conversely, the claim description is stored on-chain in unencrypted form, as it is subject to the public sphere requirement, which can be natively enforced by the blockchain. Also, task Assess urgency is executed on-chain, as its outcome determines the branch taken by the Urgent? gateway, which is subject to the public enforceability requirement. In this way, this requirement can be natively enforced.

7. Evaluation

We evaluated the SecBPMN2BC method using three real case studies to evaluate the expressiveness of the modeling language and the utility of the software tool. We also designed and performed an empirical experiment for a wider and unbiased analysis of the method.

7.1. Case study evaluation

We selected the case studies based on the relevance of their domain and the sensitivity of the information used: (i) management of birth certificates in a large Greek municipality; (ii) visit of a pediatric patient in an Italian hospital; (iii) teleconsultation of a pediatric patient between an Italian and a Spanish hospitals.

The case studies were not originally designed for blockchain execution. Therefore, we slightly modified their processes. For each case study, a description of the original and the modified processes can be found here [35]. The Business processes we used had respectively 21, 37, 48 activities and 21, 16, 28 security (and privity) annotations. Thus, they can be considered medium-size processes.

Using SecBPMN2BC-ML, we were able to fully define the processes thanks to the expressiveness of SecBPMN2 which is the baseline of the modeling language, and thanks to the extensive support for security annotations provided by our extension. To this aim, SecBPMN2BC-Tools proved to be a very useful support, since it detected conflicts, and it identified tasks and data objects that required to be put on-chain to satisfy security annotations. For example, the assignment discussed in the last paragraph of Section 6 about the example shown in Fig. 3 was detected thanks to software and algorithms we developed.

When selecting the blockchain type as private, SecBPMN2BC-Tools identified that all data objects can be stored in unencrypted form on-chain. In this case also Check claim should be executed on-chain to enforce the public strong-dynamic privity requirement of the Local? gateway. However, this will only possibly support the public enforceability requirement on a private blockchain. Since there are no privity requirements on Claim description, this can be stored on-chain regardless of the blockchain type and the corresponding decision tasks should also be executed on-chain to natively enforce the public enforceability requirement on Urgent?.

If a designer is not satisfied by the derived solution, she can manually change the on/off-chain annotations and re-run the application. E.g., if other requirements (e.g., storage cost) only allow to store a digest on-chain, this can be enforced by overriding the derived values and re-executing the software. The workflow proposed in the method was followed naturally. Applying SecBPMN2BC to the case studies was a success, yet we identified a limitation inherited from BPMN 2.0: the inability to distinctively identify data objects among different diagrams. This prevents to refer (easily) to external data objects, not allowing to derive, for example, which users are allowed to access a data object in case of static or private spheres.

7.2. Empirical experiment

The design of the empirical experiment being used for evaluating the SecBPMN2BC method was based on the Wohlin et al. approach [36]. Such an approach is well known and widely used, and it guides the designers from the definition of the scope of the experiment to a critical analysis on the validity threats.

7.2.1. Experiment scoping and research questions

We focused on the targeted users of the method: a team of security (and privacy) experts and business process experts that will collaborate for the design of the functional part of business processes and security requirements to be enforced.

We identified the following Empirical Research Questions (ERQs), based on the contributions proposed in this paper to address the research questions identified in the introduction:

[ERQ1] is SecBPMN2BC-ML expressive enough to specify security (and privity) requirements of business processes?

[ERQ2] is SecBPMN2BC-ML expressive enough to specify secure business processes to be executed in blockchain?

[ERQ3] are the rules implemented in SecBPMN2BC-Tools for the identification of on/off-chain elements valid?

[ERQ4] Does the SecBPMN2BC method correctly guide users?

ERQ1 aims at investigating the fit of the modeling language to be used to specify business processes with a focus on security (and privity) requirements, i.e., these requirements need to be specified along with functional requirements of processes.

ERQ2 focuses on the suitability of the proposed modeling language as a specification for the enforcement of the business processes, with security requirements, in blockchain.

ERQ3 tests the rules and the algorithms defined in this article, in order to understand if the identification of elements to be placed on/off chain is reasonable. Rules may change on contexts and application domains. We, therefore, test the suitability of rules in certain domains, not their absolute validity.

ERQ4 is focused on the method described in this article. In particular, it aims at evaluating the applicability of the process provided with the method and if it effectively guides users in the part of the design life cycle of blockchain that is covered by SecBPMN2BC method.

7.2.2. Experiment design

The design of the experiment was guided by the research questions. We opted for anonymous online questionnaire to reach the highest possible number of subjects. The questionnaire[5] is divided in 6 parts, described below.

Part 1 asks demographic data of subjects, and their expertise in security, privacy, business process modeling and blockchain.

Part 2 shows a ten minutes video with the training on SecBPMN-2BC, after that it shows a simple SecBPMN2BC-ML diagram of a road misconstruction business process, asking questions on the model (to verify if the subject can correctly read the diagram), and questions related to the ERQs defined before.

Part 3 shows a larger SecBPMN2BC-ML diagram and asks similar questions of Part 2. This part is optional.

Part 4 provides information on SecBPMN2BC method and asks the subjects questions related to ERQ 4.

Part 5 shows the diagram already presented in Part 2, asking the subjects to identify on/off chain elements. After that, the questionnaires shows the results obtained with SecBPMN2BC-Tools applying the rules defined in this article, and it asks subjects if they agree and, if not, which are the reasons.

Part 6 is an informal assessment of the questionnaire.

The business processes provided to the subject of the experiment in parts 2 and 3 were created using a common situation that can be easily understood by most of the subjects: the focus of the experiment is not on the expressiveness of the modeling language for functional requirements, rather on the security ones. Similarly, in order to evaluate the modeling language with a larger case study, we opted for an extension of the diagram provided in Part 2. In this way, we minimized the effort of subjects in answering the questionnaire.

7.2.3. Results

42 subjects answered the questionnaire. 40% of them are security experts (i.e., they classify their knowledge on security 4 or 5 on a scale from 1 (lowest) to 5 (highest)). 54% of the subject are BPMN experts. This is coherent with our dissemination strategy since we targeted BPMN and security experts, that are the target users of the method proposed in this article.

73.8% of the subjects have a medium to high knowledge on security/privacy by design (3 to 5 points on a 1 to 5 scale) and 73.8% have a medium to high knowledge on blockchain. 19% of the participants are experts in both BPMN and security, in particular, 35% of the BPMN experts are security experts while 47% of the security experts are BPMN experts. 48% of the subjects are both BPMN and blockchain experts, while 14% of the subjects are both BPMN and security by design experts. 59% of the subjects are both security and security by design experts. 81% of the subjects are researchers (professors, researcher, Ph.D. students), while 7% are M.Sc. students and 12% employees. The optional part of the form was completed by 82% of the subjects.

Table 7 shows the aggregated results of the experiment. Table 8 shows the most relevant textual feedback left by the subjects. The complete raw data can be found here [35]. The rest of the section discusses the results collected, organized by research questions.

ERQ1 Answers to questions 1, 2, 5 in Table 7 show that subjects find SecBPMN2BC-ML easy to understand. Even if the scores are not so high, respectively 3.60, 3.45 and 3.52 (on a scale from 1 (lowest) to 5 (highest)), they are convincing if we consider

that subjects answered the questionnaire after only a 10-minutes training done with a video. A longer training, with a more distributed cognitive load, will have a considerably positive impact on these results. This is also highlighted on the more positive answers of Question 2 (Q2) after the second (larger) diagram: after only a small exercise, subjects were more confident on the interpretation of security constraints. It is worth noticing that scores of Q5 are lower for a large diagram, this is consistent with the well-known issue of scalability of the BPMN modeling language. This is also partially reflected in the answers of Q7, where the first diagram has a higher score of readability respect to the second one.

Q6 highlights that most of the subjects believe that the necessary security concepts are covered by SecBPMN2BC-ML, i.e., it is expressive enough for their needs (76%, 82% after the second diagram). Moreover, by examining the textual feedback given by the subjects that answer negatively to Q6, we noticed that they expressed the need to represent out of scope concepts, such as safety and threats.

We can, therefore, conclude that the interviewed subjects believe that SecBPMN2BC-ML is easy to understand and it covers most of the security concepts that are needed. For future work we may consider extending the language with other concepts such as safety and threats.

ERQ2 This research question focuses on the feasibility of enforcing security concepts expressed in SecBPMN2 using blockchain technology. This can be evaluated by considering results of Q3 and Q4. In particular, subjects confirm that blockchain technology can be used to enforce security concepts in general (Q3) and for specific cases as the ones proposed in the questionnaire (Q4). The results show that subjects agree that SecBPMN2BC-ML can be used to specify secure business processes, and those processes can be enforced with blockchain.

ERQ3 Q10, Q11, and Q12 are related to ERQ3. Their results indicates that the large majority of the subjects did not identify on/off chain elements as indicated by the heuristic we specified (Q11, 69%). This result, considered with Q10 and Q12 whose results highlight a strong need for a software tool that supports the rules, indicates that subjects experienced difficulties on the application of the rules. Yet, they believe that, if supported by SecBPMN2BC-Tools, such rules will be useful and valid. This is also confirmed by the textual feedback we received, as in general subjects did not know how to apply the rules, even if they agree with them. Given the results of the questionnaire, we can conclude that the rules are perceived as valid and useful but, to be applied, they need to be supported by SecBPMN2BC-Tools, since they are too complex to be applied by users.

ERQ4 We interpreted a "correct" method as a method that is easy to follow and adhere to best practices and approaches followed by the targeted users. Results of Q8 and Q9 shows that the method we defined in this article is considered easy to follow (Q8) and uses an approach similar to the one subjects would have used on their own (Q9). The scores of Q8 and Q9 are higher for BPMN experts, probably because they are used to work with similar approaches. Very relevant is result of Q13 that highlights the usefulness of the method for designing secure business process targeting blockchain. Also in this case we can consider ERQ4 to be true, since subjects believe the method is perceived as natural and it is easy to follow.

Generally speaking, the results are all positive, but not so high to consider all ERQs completely answered. There is still a margin of uncertainty. There may be many reasons for this uncertainty, spanning from the length of the questionnaire, to the missing software support. However, the positive results allow to

---

5 A pdf version of the questionnaire can be found at [35]

Table 7

Results of the questionnaire on diagrams. Questions 1–5, 8–10, and 12–13 are on a scale from 1 (lowest) to 5 (highest), and the values in the *dev/%* columns indicate the standard deviation. Questions 6, 7, and 11 are closed-ended, and the values in the *dev/%* columns indicate the percentage for each answer.

Questions			Small diagram						Large diagram					
			All		Security exp.		BPMN exp.		All		Security exp.		BPMN exp.	
Id	Text		avg	dev/%	avg	dev/%	avg	dev/%	avg	dev/%	avg	dev/%	avg	dev/%
1	Is it easy to identify the annotations?		3,60	0,87	3,24	0,81	3,78	0,93	3,47	0,92	3,38	0,74	3,68	0,92
2	Is it easy to understand the constraints specified by the proposed annotations?		3,45	0,76	3,35	0,76	3,52	0,71	3,56	0,81	3,38	0,84	3,63	0,67
3	Do you think blockchain technology can be used to enforce security concepts expressed by the annotations?		3,55	0,85	3,18	0,92	3,57	0,92	3,59	0,73	3,38	0,84	3,68	0,73
4	Do you think it is realistic to use blockchain technology to enforce security in this business process?		3,52	0,76	3,29	1,02	3,52	0,71	3,50	0,81	3,08	0,92	3,58	0,67
5	Is this process easy to read?		3,52	1,12	3,47	1,04	3,70	1,08	3,21	0,99	3,31	0,72	3,26	1,07
6	Do you think there are other security concepts that cannot be represented with the annotations proposed in this survey?	yes		0,24		0,28		0,33		0,18		0,21		0,20
		no		0,76		0,67		0,63		0,82		0,71		0,75

Questions			All		Security exp.		BPMN exp.	
Id	Text		avg	dev/%	avg	dev/%	avg	dev/%
7	Compare the first and the second figures, which one is easier to read?	equally easy		0,47		0,43		0,45
		first figure		0,38		0,36		0,40
		second figure		0,15		0,14		0,10
8	Do you think it is easy to follow the process described?		3,88	0,98	3,59	0,97	4,09	0,88
9	Do you think the process follows naturally your approach to a similar problem?		3,76	0,92	3,76	0,88	3,96	0,86
10	Do you think it is useful having a software tool to identify conflicting requirements?		4,62	0,72	4,41	0,84	4,74	0,53
11	Did you identify the same BPMN elements to be stored/executed on chain?	yes		0,31		0,28		0,38
		no		0,69		0,67		0,58
12	Do you think it is useful having a software tool that derives solutions as the one shown in the table and the image above?		4,38	0,84	4,18	0,98	4,52	0,77
13	How helpful would be the method proposed in this survey for system designers to support the design of secure business processes for blockchain?		3,81	1,05	3,65	0,90	3,87	1,12

Table 8

Feedback from subjects.

ID	Question/Feedback
QF1	*Do you think there are other security concepts that cannot be represented with the annotations proposed in this survey?*
F1	Enforceability might require just a subset of a group, e.g., any 3 out of a specified 5; or party A and B plus 2 out of another 7.
QF2	*Do you have any remarks on the activities identified?*
F2	It was hard for me what it means to implement an activity "on-chain", as there can be many ways of doing so (mapping BPMN to smart contracts or directly using a smart contract) [...]
QF3	*Do you have any comments regarding the method proposed in this survey? If yes, please specify it here.*
F3	I also would like to see, how the approach supports evolution of processes and applications that may lead to conflicts between two versions.
F4	I think that the method requires more than basic security and blockchain notions. I am not sure that a BPMN expert without much background on blockchain would be able to fully specify the constraints and understand the results proposed by the software.
F5	It would help to have a common element in the symbols for privacy/privity to quickly identify and differentiate them from the other concerns – just like the train in enforceability. The access spheres seem really to be about privacy.

consider the solution proposed in this paper a valid answers to the research questions of this paper, i.e., RQ1 to RQ4.

Table 8 reports the most interesting comments left by the subjects. Feedback 1 (F1) highlights the possibility of a finer granularity for the specification of enforceability. This will increase the applicability of SecBPMN2BC, we will consider this in a future extension of the method.

F2 highlights the different perspectives on how to consider a process element to be on-chain. An example is the upload of data digest on chain: even if we consider this solution as on-chain, it might be considered off-chain by some experts.

F3 asks for a version control of evolution of a process, another interesting feature we will consider as future work.

F4 considers the requirements the method poses for security and BPMN experts. The method should ease the approach, for these experts, when dealing with blockchain by hiding the complexity of the rules. With the rules applied by SecBPMN2BC-Tools (it was not the case on the questionnaire), much of the complexity will be hidden to the final users.

F5 highlights an issues about the number of symbols introduced by SecBPMN2BC-ML. We agree with the feedback and we propose a solution where in the diagram visualized by the software tool, the type of enforceability and security annotations are visualized as property. This will reduce the visual complexity of the diagrams while using a virtual diagram. For the printed version we will keep using the original version of the annotations.

7.2.4. Threats to validity

A fundamental question about the results of an empirical experiment is how valid the results are. We analyzed the relevant threats to validity and provided a description of the actions we took to mitigate them.

Table 9
Feature comparison of related works.

	DC	NtK	DL	BoD/SoD	Int	Aud	Av	Privacy	SC/BC
SecBPMN2BC-ML	+	+/−	+	+	+	+	+	+/−	+
SecBPMN2 [19]	+/−	−	+	+	+	+	+	+/−	−
SecureBPMN [37]	+	+/−	+	+	−	−	−	−	−
UMLSec [38]	+	−	−	−	+	−	−	−	−
PE-BPMN [39]	+/−	−	−	−	+	−	−	+	−
Labda et al. [40]	+/−	+/−	−	+	−	−	−	+	−
Rodríguez et al. [41]	+/−	−	−	−	+	+	−	+/−	−
Wolter et al. [42]	+/−	−	−	+	+	+	+	−	−
Saleem et al. [43]	+/−	−	−	−	+	+	+	−	−
Mülle et al. [44]	+/−	−	+	+	+	+	−	+/−	−

Wohlin et al. [36] divide the threats in four classes: (i) conclusion validity, (ii) internal validity, (iii) construct validity and (iv) external validity. The following part reports the analysis of the most relevant threats, with the mitigation solutions we adopted. For the complete list please refer to [29].

Low statistical power: the power of a statistical test, in our case a limited number of subjects. We reached a relevant number of subjects, considering the targeted population.

Maturation: subjects react differently as time passes, in our case it can impact negatively (e.g., tired subjects) or positively (learning). For what concerns negative impacts, we design the experiment to be as short as possible: it can be executed in less than 20 min. For what concerns positive impacts, subjects will gain experience by answering questions in Part 2 and they will use this experience for Part 3. This is a desired effect, since we also test the learning curve of the method.

Hypothesis guessing: "When people take part in an experiment, they might try to figure out what the purpose and intended result of the experiment is." [36]. We design the experiment and prepare the questionnaire to be as neutral as possible. Subjects of the experiment are randomly chosen and, in general, have little or no relation with the designers of the experiment.

Interaction of selection and treatment: the subjects are not representative of the population. We distributed the questionnaire to as many experts as possible. The first part of the questionnaire elicits information on the experts in order to, possibly, filter the subjects that do not represent the targeted population.

8. Related work

8.1. Business processes on blockchain

During the last years, several researchers explored the synergy between the blockchain and BPM [6–8,45,46]. Contributions range from monitoring and enforcing the execution of choreographies [7] to on-chain process engines such as [6] or the collaborative management of models on blockchains [47].

The framing theme is a model-driven approach where process models are compiled in smart contract code. Consequently, processes are executed by calling blockchain transactions. In early approaches like [6], the major focus was laid on securing the correctness of the control flow perspective, while later extensions also covered the resource perspective by supporting static [48] and dynamic role binding [30]. Explicit modeling of decisions and securing the correctness of decisions were addressed in [49], while [50] extended it to confidentiality.

Smart Contract code can only access data that are either provided as input for blockchain transactions or are already stored on-chain. In the same sense, transaction output data are always stored on-chain. Consequently, specific care needs to be taken when off-chain data is required on-chain, and vice-versa. Entities providing external data to the blockchain or vice-versa are referred to as oracles. A collection of oracle patterns for blockchain based applications is provided in [51]. On-chain data including all transaction inputs and outputs are visible to all nodes of the network. In many applications, constraints on confidentiality and on cost require the usage of off-chain data. A discussion on the architectures for blockchain-based applications including the placement of data can be found in [52]. While the execution of business processes on blockchain comes with several advantages, confidentiality is not intrinsically solved on the widely adopted blockchain frameworks. The work in [53,54] gives an overview of the aspects of confidentiality in business processes on blockchain and discusses existing techniques to (partially) address this aspect. An early discussion on the need to balance enforceability and privity/confidentiality requirements was presented in [15].

8.2. Modeling security requirements of business processes

Various works propose the extension of business process models with security constraints. The approach in [55] proposes to include fine-grained and dynamic access control definitions in inter-organizational processes. This is a novelty compared to the predominant RBAC model, where rules of real world systems are mostly static not referring to the current instance state. In [38] a security extension for UML, including security requirements, concepts, primitives, and threat scenarios, is proposed. The works [19,37,39–43] provide security extensions for BPMN. An overview of the related works regarding the features Data Confidentiality (DC), Need to Know (NtK), Delegation (DL), Binding of Duties and Separation Of Duties (BoD/SoD), Integrity (Int), Auditability (Aud), Availability (Av), Privacy (Priv), and Blockchain and Smart Contract Specific requirements (SC/BC) is shown in Table 9. In the table, we use (+) to express support for a certain feature, (+/−) for partial support, and (−) for no support.

We based our work on SecBPMN2 [19] because it already covers a wide range of security requirements and it was comprehensively evaluated. However, it misses modeling features for smart contract-specific (privity, enforceability) and blockchain-specific requirements. We have therefore created SecBPMN2BC-ML to cover these missing aspects. None of the other approaches addresses these requirements. Original SecBPMN2 does not support dynamic confidentiality constraints, which are now covered by SecBPMN2BC-ML in form of privity requirements. The approaches [37,42] provide dynamic access control for data objects. However, this is basically achieved by low-level, non-graphical annotations. While such low-level access rules can easily be enforced with access control systems as proposed in [37,42], this is not appropriate for blockchains. E.g., temporal access permissions/revocation of access is impossible for on-chain data. Therefore, confidentiality constraints can have stronger consequences on the modeled system. In our approach, we provide high-level graphical privity annotations that can be used for modeling and communicating blockchain-relevant aspects of confidentiality, including static and dynamic cases. This is especially relevant as privity and enforceability requirements can lead to goal conflicts on blockchains [15].

In this paper, we did not focus on processes affected by privacy regulations. The work in [40] additionally addresses the privacy requirements user consent and Necessity to Know (NtK). NtK is similar to privity. It aims to limit access to information to only those users who strictly need it for executing some tasks [40]. Our definition of strong-dynamic privity can be seen as a dynamic interpretation of this property, where we allow access once it is certain that a task with a NtK of a data object will be executed by the participant in question. This dynamic aspect is absent in [40] where NtK rules are not bound to specific BPMN tasks. Also, SecureBPMN [37] supports NtK as a specialization of authorization. Here the same comments as for access control in SecureBPMN apply. Finally, also the work [39] addresses privacy issues in Business processes. However, it focuses on the technological level by annotating business processes with Privacy Enhancing Technologies.

9. Conclusion and future work

This article proposes SecBPMN2BC, a method for the design of secure business processes for blockchains. It is composed of three main elements: a workflow, the SecBPMN2BC-ML modeling language, and the SecBPMN2BC-Tools software. The workflow follows a model-driven approach where experts are guided in designing secure business processes in SecBPMN2BC-ML. The SecBPMN2BC-Tools support experts in identifying conflicting requirements and in determining sets of on-chain and off-chain process elements to comply with the modeled requirements via rules. We evaluated the SecBPMN2BC method with three case studies, showing that SecBPMN2BC can be effectively used for the definition of secure business processes for blockchains, and with an empirical experiment that validates SecBPMN2BC-ML, the usability of the method, and the validity of SecBPMN2BC-Tools. The positive results of the empirical experiment validate the solution we proposed in this paper, answering the research questions defined in Section 1. SecBPMN2BC lays the ground for the model-driven engineering of security-aware information systems on blockchains opening alleys for future works. The proposed algorithms compute locally optimal solutions leaving algorithms for globally optimal solutions as future work. While the generated blockchain-specific SecBPMN2BC-ML models already provide important insights for manual implementations, we consider automatic recommendations of patterns for developers as well as the automatic generation of on-chain and off-chain code as an important next step. Finally, we did not focus on exception handling and delegated it to the existing methods of BPMN. However, modeling exceptions and their handling is even more important in our setting as no superuser can resolve issues in an ad-hoc manner. This leaves room for interesting future works.

CRediT authorship contribution statement

Julius Köpke: Conceptualization, Methodology, Writing – original draft. **Giovanni Meroni:** Conceptualization, Methodology, Software, Writing – original draft. **Mattia Salnitri:** Conceptualization, Methodology, Validation, Investigation, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data are shared via Mendeley Data and code is shared via GitHub. Links are provided in the paper.

References

[1] N. Szabo, Formalizing and securing relationships on public networks, First Monday 9 (2) (1997).

[2] V. Buterin, Ethereum: A next-generation smart contract and decentralized application platform, 2014, URL https://ethereum.org/en/whitepaper/. (Accessed 28 October 2021).

[3] K. Delmolino, M. Arnett, A. Kosba, A. Miller, E. Shi, Step by step towards creating a safe smart contract: Lessons and insights from a Cryptocurrency Lab, in: Financ. Cryptogr. Data Secur., Springer, 2016, pp. 79–94.

[4] R. Hull, V.S. Batra, Y. Chen, A. Deutsch, F.F.T.H. III, V. Vianu, Towards a shared ledger business collaboration language based on data-aware processes, in: Proc. of ICSOC 2016, 2016, pp. 18–36.

[5] J. Ladleif, M. Weske, A unifying model of legal smart contracts, in: Proc. of ER, Vol. 11788, Springer, 2019, pp. 323–337.

[6] O. Pintado, L. García-Bañuelos, M. Dumas, I. Weber, A. Ponomarev, Caterpillar: A business process execution engine on the ethereum blockchain, Softw. - Pract. Exp. (2019).

[7] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, J. Mendling, Untrusted business process monitoring and execution using blockchain, in: Proc. of BPM, 2016, pp. 329–347.

[8] A.B. Tran, Q. Lu, I. Weber, Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management, in: Proc. of BPM, 2018, pp. 56–60.

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

[10] Y. Huang, Y. Bian, R. Li, J.L. Zhao, P. Shi, Smart contract security: A software lifecycle perspective, IEEE Access 7 (2019) 150184–150202.

[11] Crystal Blockchain, Map of Security Breaches and Fraud Involving Crypto 2011–2021. Last visited April 2021, Tech. Rep..

[12] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manage. Inf. Syst. 24 (3) (2007) 45–77.

[13] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2009, http://www.bitcoin.org/bitcoin.pdf.

[14] D. Basile, V. Goretti, C. Di Ciccio, S. Kirrane, Enhancing blockchain-based processes with decentralized oracles, in: Proc. of BPM Blockchain and RPA Forum, 2021, pp. 102–118.

[15] J. Köpke, M. Franceschetti, J. Eder, Balancing privity and enforceability of BPM-based smart contracts on blockchains, in: Proc. of BPM Blockchain Forum, in: LNCS, vol. 361, Springer, 2019, pp. 87–102.

[16] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: A distributed operating system for permissioned blockchains, in: Proc. of EuroSys, 2018, pp. 1–15.

[17] O. Goldreich, Y. Oren, Definitions and properties of zero-knowledge proof systems, J. Cryptol. 7 (1) (1994) 1–32.

[18] M. Blum, P. Feldman, S. Micali, Non-interactive zero-knowledge and its applications, in: Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, Association for Computing Machinery, New York, NY, USA, 2019, pp. 329–349.

[19] M. Salnitri, E. Paja, P. Giorgini, Maintaining secure business processes in light of socio-technical systems' evolution, in: Proc. of IEEE MoDRE, 2016, pp. 155–164.

[20] D.L. Hofman, Legally speaking: Smart contracts, archival bonds, and linked data in the blockchain, in: Proc. of ICCCN, 2017, pp. 1–4.

[21] A. Savelyev, Contract law 2.0: 'Smart' contracts as the beginning of the end of classic contract law, Inform. Commun. Technol. Law 26 (2017) 116–134.

[22] J. Ladleif, M. Weske, A legal interpretation of choreography models, in: Proc. of BPM Workshops, in: LNCS, vol. 362, pp. 651–663.

[23] V. Dwivedi, A. Norta, A. Wulf, B. Leiding, S. Saxena, C. Udokwu, A formal specification smart-contract language for legally binding decentralized autonomous organizations, IEEE Access 9 (2021) 76069–76082.

[24] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor, X. Xu, On legal contracts, imperative and declarative smart contracts, and blockchain systems, Artif. Intell. Law 26 (4) (2018) 377–409.

[25] W. Tolone, G.-J. Ahn, T. Pai, S.-P. Hong, Access control in collaborative systems, ACM Comput. Surv. 37 (1) (2005) 29–41.

[26] D. Basin, J. Doser, T. Lodderstedt, Model driven security: From UML models to access control infrastructures, TOSEM 15 (1) (2006) 39–91.

[27] D. Moody, The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering, in: TSE 2009, Vol. 35, pp. 756–779.

[28] E. Parliament, C. of the European Union, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), Off. J. Eur. Union (2016).

[29] J. Köpke, G. Meroni, M. Salnitri, SecBPMN2BC: Specifications and Rules, Tech. Rep., Politecnico di Milano, University of Klagenfurt, Technical University of Denmark, 2022, URL http://purl.org/secbpmn2bc.

[30] O. López-Pintado, M. Dumas, L. García-Bañuelos, I. Weber, Dynamic role binding in blockchain-based collaborative business processes, in: Proc. of CAiSE, 2019, pp. 399–414.

[31] X. Xu, I. Weber, M. Staples, Architecture for Blockchain Applications, Springer, 2019.

[32] C.D. Ciccio, G. Meroni, P. Plebani, On the adoption of blockchain for business process monitoring, Softw. Syst. Model. 21 (3) (2022) 915–937.

[33] S. Rouhani, R. Deters, Blockchain based access control systems: State of the art and challenges, in: WI 2019, ACM, 2019, pp. 423–428.

[34] X. Xu, C. Pautasso, L. Zhu, Q. Lu, I. Weber, A pattern collection for blockchain-based applications, in: Proceedings of the 23rd European Conference on Pattern Languages of Programs, EuroPLoP '18, 2018, pp. 3:1–3:20.

[35] J. Köpke, G. Meroni, M. Salnitri, SecBPMN2BC Case Studies Evaluation V5, Mendeley Data, 2022.

[36] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer, 2012.

[37] A.D. Brucker, I. Hang, G. Lückemeyer, R. Ruparel, SecureBPMN: Modeling and enforcing access control requirements in business processes, in: Proc. of SACMAT, 2012, pp. 123–126.

[38] J. Jurjens, UMLsec: Extending UML for secure systems development, in: Proc. of UML, 2002, pp. 412–425.

[39] P. Pullonen, R. Matulevičius, D. Bogdanov, PE-BPMN: Privacy-enhanced business process model and notation, in: Business Process Management, Springer, Cham, 2017, pp. 40–56.

[40] W. Labda, N. Mehandjiev, P. Sampaio, Modeling of privacy-aware business processes in BPMN to protect personal data, in: Proc. of SAC, 2014, pp. 1399–1405.

[41] A. Rodríguez, E. Fernández-Medina, M. Piattini, A BPMN extension for the modeling of security requirements in business processes, Inst. Electron. Inform. Commun. Eng. Trans. Inform. Syst. 90 (4) (2007) 745–752.

[42] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, C. Meinel, Model-driven business process security requirement specification, J. Syst. Archit. 55 (4) (2009) 211–223.

[43] M. Saleem, J. Jaafar, M. Hassan, A domain-specific language for modelling security objectives in a business process models of SOA applications, Adv. Inf. Sci. Serv. Sci. 4 (1) (2012) 353–362.

[44] J. Mülle, S. von Stackelberg, K. Böhm, A Security Language for BPMN Process Models, Tech. Rep. 9 (2011), Karlsruher Institut für Technologie, 2011.

[45] C. Sturm, J. Scalanczi, S. Schönig, S. Jablonski, A blockchain-based and resource-aware process execution engine, FGCS 100 (2019) 19–34.

[46] M.F. Madsen, M. Gaub, T. Høgnason, M.E. Kirkbro, T. Slaats, S. Debois, Collaboration among adversaries: distributed workflow execution on a blockchain, in: Symposium on Foundations and Applications of Blockchain, 2018.

[47] H. Fill, F. Härer, Storing and attesting conceptual models on blockchains (invited paper), in: Proc. of Modellierung, Vol. 2542, 2020, pp. 51–52.

[48] L. Mercenne, K. Brousmiche, E.B. Hamida, Blockchain studio: A role-based business workflows management system, in: Proc. of IEMCON, 2018, pp. 1215–1220.

[49] S. Haarmann, K. Batoulis, A. Nikaj, M. Weske, DMN decision execution on the ethereum blockchain, in: Proc. of CAiSE, 2018, pp. 327–341.

[50] S. Haarmann, K. Batoulis, A. Nikaj, M. Weske, Executing collaborative decisions confidentially on blockchains, in: Proc. of BPM Blockchain and CEE Forum, Springer, 2019, pp. 119–135.

[51] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, U. Zdun, Foundational oracle patterns: Connecting blockchain to the off-chain world, in: Proc. of BPM Blockchain and RPA Forum, Springer, 2020, pp. 35–51.

[52] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, P. Rimba, A taxonomy of blockchain-based systems for architecture design, in: Proc. of ICSA, 2017, pp. 243–252.

[53] B. Carminati, E. Ferrari, C. Rondanini, Blockchain as a platform for secure inter-organizational business processes, in: Proc. of CIC, 2018, pp. 122–129.

[54] S. Ghesmati, W. Fdhila, E.R. Weippl, Studying bitcoin privacy attacks and their impact on bitcoin-based identity methods, in: Proc. of BPM Blockchain and RPA Forum, Vol. 428, Springer, 2021, pp. 85–101.

[55] M.H. Kang, J.S. Park, J.N. Froscher, Access control mechanisms for inter-organizational workflow, in: Proc. of SACMAT, 2001, pp. 66–74.

Julius Köpke is an Associate Professor of Informatics at the Department of Informatics-Systems of the University of Klagenfurt. His primary research interests are smart contracts, inter-organizational business processes, and application interoperability. His major works include the automatic optimization of data-flows in inter-organizational processes, interoperability of artifact-centric and activity-centric processes, and an annotation method for message schemas for supporting data interoperability based on a reference ontology. Recently, Julius Köpke has focused on the mode-driven development of inter-organizational processes targeting blockchain-based implementations.

Giovanni Meroni is an Assistant Professor at Technical University of Denmark. Previously, he was a Postdoctoral Research Assistant at Politecnico di Milano. He received a Ph.D. in Information Technology in 2018 at Politecnico di Milano. His research interests include Information Systems, Business Process Management (BPM), Blockchain, and the Internet of Things. His main research work focuses on IoT-based Multi-party Process Monitoring, Ontology-based Sensors and Sensor Data Retrieval, and Synergies between blockchain and BPM. His Ph.D. dissertation was awarded the CAiSE 2019 PhD Award, and he received the best paper award at the BPM 2021 Blockchain Forum.

Mattia Salnitri is a research fellow at Politecnico di Milano and a visiting researcher at Bournemouth University. Before that, he was a post-doc researcher at University of Trento, where he received the Ph.D. in computer science and telecommunication in 2016. His research focuses on business processes, security, privacy and socio-technical systems. His main research works include extension of BPMN with security and privacy concepts and methods for the design of secure, large socio-technical systems.