Blockchain as a platform for secure inter-organizational business processes

Barbara Carminati
DiSTA
University of Insubria, Italy
Email: barbara.carminati@uninsubria.it

Elena Ferrari
DiSTA
University of Insubria, Italy
Email: elena.ferrari@uninsubria.it

Christian Rondanini
DiSTA
University of Insubria, Italy
Email: c.rondanini@uninsubria.it

***Abstract*—Today, most of the services one may think of are based on a collaborative paradigm (e.g., social media services, IoT-based services, etc.). One of the most relevant representative of such class of services are inter-organizational processes, where an organized group of joined activities is carried out by two or more organizations to achieve a common business goal. Inter-organizational processes are therefore vital to achieve business partnerships among different organizations. However, they may also pose serious security and privacy threats to the data each organization exposes. This is mainly due to the weak trust relationships that may hold among the collaborating parties, which result in a potential lack of trust on how data/operations are managed. In this paper, we discuss, how blockchain, one of today hottest technology, can be used in support of secure inter-organizational processes. We further point out which additional security issues the use of blockchain can bring, illustrate the ongoing research projects in the area and discuss future research directions.**

***Keywords*—Business processes; Service composition; Blockchain; Smart contracts; Oracles; Privacy; Confidentiality;**

I. Introduction

Today we are living in an era where technology makes more and more easy collaboration in many different applications domains (e.g., social networks, IoT, Big Data analytics) and multi-organizational settings. Collaboration clearly brings many benefits in terms of resource usage optimization, better services, knowledge sharing, and so on. However, it poses also many research challenges. One of the most relevant is the lack of mutual trust among the collaborating entities. The issue of trust is particular relevant when collaboration is due to inter-organizational business processes. In such a case, collaborating entities may not be related by strong trust relationships, still they have to collaborate to perform a mutual beneficial process. As a result such collaboration may enact the misuse or leakage of confidential data and information.

In this paper, we focus on one of the main technology that promises to achieve secure inter-organizational processes among untrusted entities, that is, blockchain [1]. A blockchain is a distributed data structure, replicated and shared among members of a network, acting as a *distributed ledger*, used to keep track of every exchange of resources or assets between participants of a network. These changes are recorded into *transactions*, batched into time-stamped linked blocks, forming the so-called *chain of blocks*. Transactions are inserted into blocks only if they are considered *valid* by the network participants. Transaction validation is reached through a distributed consensus protocol that, in general, is considered secure if the majority of network participants are honest. An important aspect, when leveraging on blockchain, is that the computation involved in transaction validation can be encoded into predefined programs. For instance, the well-known Bitcoin framework provides a set of programs tailored for cryptocurrency management. In contrast, more recent blockchain frameworks, like Ethereum [2] [3], support the idea of running arbitrary user-defined programs, called *smart contracts*. The idea is to translate contractual clauses into smart contracts stored and executed on the blockchain. As such, blockchain can be seen as a distributed ledger storing results (i.e., transactions) of (smart) contracts whose correct evaluation have been validated by network participants. This view brings several benefits. The first is that it does not require a central trusted party to validate the correct execution of contracts. Moreover, the obtained transparency well overcomes the lack of trust among parties involved in the contracts. As a result, blockchain/distributed ledger have gained an increasing interest from companies aiming at encoding with smart contracts their processes and collaborations.

By leveraging on blockchain to support secure inter-organizational business processes, collaborating organizations expose their services to be directly invoked by smart contracts, having the smart contract playing the logic of monitoring the overall service composition. Distributed consensus ensures the correctness of smart contract execution, aka their transactions, ensuring thus the correctness of collaborations among different organizations. This is a promising approach, which is receiving growing attention by the research community (e.g., [4] [5] [6] [7], [8]). However, it brings also further security issues.

In this paper, we discuss how blockchain can be used on support of secure inter-organizational business processes and which are the benefits and the security shortcomings it poses. We then survey existing research proposals and ongoing projects in the area, and outline future research directions.

The remainder of this paper is organized as follows. Section II discusses how blockchain can be used on support of inter-organizational processes, whereas Section III illustrates the related security issues. Section IV gives an overview of ongoing research in the area. Finally, Section V concludes the paper by discussing future research directions.

II. Blockchain for inter-organizational processes

A blockchain is a distributed data structure, replicated and shared among members of a network, which acts as a distributed ledger to keep track of every exchange of resources or assets among participants of that network. These changes are recorded into transactions, batched into time-stamped linked blocks, forming the so-called *chain of blocks*. This latter is built assuming that: 1) each block is identified by its hash value (i.e., the value returned by a cryptographic hash function applied on the block content); 2) each block contains the hash value of the block that precedes it in the chain. Moreover, transactions are inserted into blocks only if they are considered *valid* by the network participants through a distributed consensus protocol that, in general, is considered secure if the majority of network participants are honest.[1] Several distributed consensus protocols have been so far proposed, however, thanks to Bitcoin, the most famous one is *Proof of work* (PoW), where network nodes have to solve a computationally intensive task in order to be selected for inserting a new block in the chain. *Proof of Stake* (PoS) is another protocol, famous thanks to the Ethereum platform,[2] where selection of the new block creator is based on the amount of stake held by network nodes. Other distributed consensus protocols have been investigated as well, ranging from the well-known Bizantine Full Tolerance algorithm (BFT) [9], [10], to new protocols tailored for the application purpose of the blockchain.

In general, transactions have to be validated according to pre-defined rules. Here, it comes the concept of *Smart Contract* (SC), aka a pre-defined program encoding the computation of transactions validation. SC-enabled blockchains can be considered general purpose application platforms, among which the most used, famous, and supported at the time of writing is Ethereum. Another relevant initiative in this direction is the Hyperledger Project, a cross-industry collaboration aiming at identification of standard open source SC-enabled blockchains and related tools.[3]

Moreover, based on the application domain, we can have different blockchain implementation, as explained in what follows:

- **Public blockchains**. In these blockchains, anyone can join the network without a specific identity. Public blockchains can be further classified into two groups, according to the constraints they impose to be part of the consensus algorithm:
  - Permissionless blockchains. In these blockchains (e.g., Bitcoin) any node in the network can participate in the consensus algorithm, being able to validate transactions.
  - Permissioned blockchains. In this type of platforms (e.g., Ripple[4], stellar[5]), only nodes that respect the rule "stake a minimum amount of tokens (coins)" are allowed to validate transactions and so to be part of the consensus algorithm.
- **Private blockchains**. In such a blockchain, only a selection of nodes are authorized to join the network. Similar to public blockchains, they can be further classified into permissionless, when any of the nodes can participate in the consensus algorithm (e.g., Ethereum), and permissioned blockchains, where only a selection of the nodes are further authorized to validate transactions (e.g., Hyperledger Fabric). In this case, the distributed consensus protocol exploits the *Proof of Authority*, to explicitly authorize nodes to create new blocks.

A. Business Processes over Blockchain

A business process is a collection of activities that, executed in a specific sequence, reach a business goal. This process can be modeled according to a workflow coordinating the various activities, formally defined via the Business Process Model Notation (BPMN).[6] In order to deploy the business process, the workflow has to be executed, aka activities have to be properly invoked. In general, these invocations can be coordinated according to two main approaches, namely, *choreography* and *orchestration* [11]. The first follows a decentralized model, where processes executing the activities coordinate in an autonomous way their invocations. In contrast, orchestration implies the presence of a central broker invoking and monitoring activities execution. Exploiting an orchestration paradigm is an interesting way for deploying a business process in blockchain, as a smart contract could play the role of central broker. Indeed, given a workflow, we can write a smart contract such that it properly invokes and monitors the process execution (aka activities invocation). To have a more concrete example, we can assume business processes are executed via web services, according to the well known service-oriented architecture (SOA).[7] In this case, the workflow can be formally defined via the Business Process Execution Language (BPEL) [12], having thus the smart contract implementing the BPEL encoding the workflow.

It is relevant to note that a smart contract cannot directly retrieve data outside the blockchain, as, by design, direct interactions with external entities are not possible. To overcome this limitation, blockchain relies on the presence of a "trustable" mediator, called *Oracle*, that retrieves data from an external source and directly delivers them to smart contracts. Figure 1 depicts the overall architecture of an inter-organizational business process deployed on the blockchain.

The idea of exploiting blockchain for inter-organizational business process execution has been investigated in several works. To the best of our knowledge, the first approach that has

[1]On assumption security of blockchain depend on the mechanism adopted to reach distributed consensus.

[2]https://ethereum.org/

[3]https://www.hyperledger.org/

[4]https://ripple.com/

[5]https://www.stellar.org/

[6]https://www.omg.org/spec/BPMN/

[7]https://www.oasis-open.org/

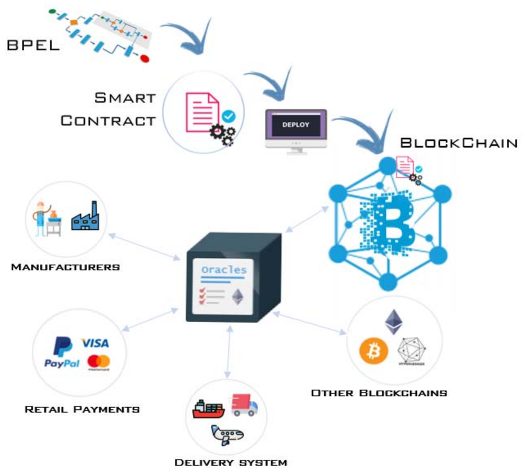

Fig. 1. Architecture of an inter-organizational business process deployed on the blockchain

proposed the idea of having the smart contract as a mediator to control the collaborative process is [5]. The translation process of a BPM to a smart contract proposed in [5] has been further optimized via petri networks in [4], whose prototype, named Cartepillar, is described in [6]. Another approach for blockchain integration in business process execution has been proposed in [13], where a blockchain-based framework for run-time verification of choreography-oriented process execution is presented. It is further interesting to note that, as discussed in [14], blockchain technology can be of support to several phases of the business process lifecycle (e.g., process identification, discovery, analysis, execution, monitor, etc.).

III. Security issues in inter-organizational processes over Blockchain

As introduced in Section I, the deployment of inter-organizational processes over blockchain can bring several benefits (e.g., trust, transparency, accountability). However,it also brings further security issues that should be properly addressed. The main ones are discussed in what follows.

- *Data integrity*. As distributed ledgers, blockchain has the intrinsic benefit of ensuring integrity of data stored into transactions. However, nothing is done for data that are not included in transactions. This means that smart contracts have to be designed in such a way to store any relevant data produced during the process execution into transactions. This solves the integrity issues but brings a further one, in that data are stored in plain text into transactions, meaning that their confidentiality is not preserved.
- *Data confidentiality*. By design, blockchain does not provide primitives for ensuring data confidentiality. This implies that every data in the blockchain, e.g., stored on the chain or simply used in a smart contract, is public and for this reason very easy to be retrieved. To cope with this issue, a first requirement is to protect sensitive data contained in transactions stored in the blocks. We refer to this requirement as **transaction confidentiality**. Furthermore, in a scenario where inter-organizational processes are coordinated via smart contracts on blockchain, data confidentiality implies two additional requirements. The first is *ensuring that each piece of sensitive data consumed during the process execution (e.g., users input, service input/output parameters) is selectively accessed only by the organization that indeed needs it for the completion of its task*. This holds both for data consumed by services involved in the process execution (e.g., the shipping address passed to a delivery service), as well as data produced during the process execution (e.g., the shipping cost returned by a delivery service). We refer to this requirement as **selective access to smart contract data**. Moreover, we recall that it is not possible for a smart contract to have direct interactions with external entities. To overcome this limitation, blockchain platforms typically rely on the presence of an *Oracle*, that is, a mediator that delivers data from external sources to smart contracts. This implies that in a smart contract playing the role of broker for an inter-organizational process, any interaction with external organizations' services needs to be mediated through the Oracle. The presence of an Oracle further implies an additional data confidentiality requirement. Indeed, we have to ensure that an *Oracle can access only information needed to correctly invoke the service, but not the input/output parameters passed to the service*. Again, as an example, this implies that when a smart contract invokes the delivery service via an Oracle, the user's shipping address has not to be disclosed by the Oracle. We refer to this requirement as **Oracle data confidentiality**.
- *Confidentiality of the process*. Data is not the only asset that should be protected when different organizations are involved into a collaborative process. Indeed, the workflow itself might reveal sensitive information about how the involved organizations managed the offered services. However, smart contracts are public, as they have to be available to peers/validators willing to check their correct execution. Therefore, by design, the inter-organizational process deployed trough a blockchain via a smart contract is exposed. Obviously this might represent a problem. Indeed, a first relevant requirement is to *ensure a confidential execution of smart contracts*, that is, the execution of smart contract steps by preserving the confidentiality of the processed sensitive data. As an example, peers/validators should be able to compute smart contract math operations on sensitive values (e.g., compute the total price as sum of shipping cost plus sales tax), by at the same time not being able to access the sensitive value (e.g., shipping cost). In the literature, the problem of protection of sensitive data during computation is known as *privacy-preserving computation*. Therefore, we refer to this requirement as **privacy-preserving smart contract**

computation A second requirement is even harder, as *the workflow encoded in the smart contract (e.g., smart contract instructions) might reveal sensitive information*. Indeed, by reading the workflow one might infer the purpose of the process, the involved organizations as well their services interface (e.g., input/output parameters), and so on. We refer to this requirement as **smart contract instructions confidentiality**.

- *Trust in the correct execution of the process*. Entities collaborating in the inter-organizational process may be related by weak trust relationships. One of the main shortcoming of this is that, since organizations are collaborating with potentially untrusted parties, they may not be sure of the correct execution of the collaborative process. However, thanks to distributed consensus, the blockchain can be considered trusted w.r.t. the correctness of smart contract execution. As such, having the smart contract implementing the workflow underling the collaborative process ensures the correct execution of inter-organizational processes. However, the presence of an Oracle as a mediator between the smart contract broker and external entities requires to carefully consider the possible data security breaches this might bring. The oracle could read and modify services invocation, by maliciously modifying the smart contract workflow. As an example, the Oracle could invoke the delivery service of a corrupted organization rather the one required by the smart contract. As such, an additional requirement for ensuring a correct execution is to assure the correct service invocation via Oracle. We refer to this requirement as **Oracle correct flow**.
- *Data provenance*. Data provenance plays a fundamental role in inter-organizational processes, where trust among the involved entities cannot always be assumed. Data provenance is the documentation of the origin of a data and, therefore, capturing and storing provenance data enables higher trustworthiness among the involved entities and greatly enhances the value of the offered services, since operating on data whose provenance is not sure may lead to incorrect results. Therefore, it is not only important to devise proper mechanisms for collecting and use provenance data, but it is also relevant to protect provenance data in that they might be more sensitive than the data itself.

IV. Ongoing solutions

Recently, several works have proposed blockchain-based frameworks offering some security services. An interesting field of application is, as an example, the one of access control in IoT where smart contracts can be of support to access control policies enforcement (e.g., [31]–[34]). Another application of blockchain for a security-related service is the one related to provenance verification. Indeed, thanks to its intrinsic accountability features, blockchain can be exploited to automatically verify the provenance of items. This has been widely investigated in different application domains (e.g., provenance for scientific data [35], supply-chain [36], [37], cloud environment [38]). These solutions could be adapted also to address the problem of *data provenance in inter-organizational processes* highlighted in previous section.

It is also interesting to note that given the increasing relevance of blockchain, several proposals have somehow addressed some of the issues introduced in Section III. In the following, for each issue we provide an overview of ongoing proposals. A summary of the security issues addressed by the considered proposals is given in Table I.

- **Transaction confidentiality.** We recall that addressing this issue implies the protection of the confidentiality of data contained into transactions. So far, this has been investigated focusing mainly on confidentiality of cryptocurrency transactions (e.g., Bitcoin). Here, sensitive information are related to payment information (i.e., amount, identities/pseudonymous of payer/payee). The first proposal dealing with this issue has considered only the amount of payment transaction [22], whereas more complex privacy-preserving payment schemes have been presented later on (e.g., [23]–[26]). However, the most relevant limitation of these proposals is that they do not support smart contracts.
  Recently a solution dealing with transaction confidentiality and supporting smart contract execution has been proposed, i.e., *Lightstream*.[8] This blockchain platform, defined for Initial Coin Offering (ICO) purpose, supports decentralized applications requiring confidentiality on transaction data. It exploits *proof of Authority* as consensus mechanism and an authorization protocol, called *permissioned blocks*, to enforce selective access to sensitive information.[9] In particular, thanks to this protocol, transactions containing sensitive information are stored into a *distributed secure vault*, aka permissioned blocks, on which accesses are granted only to authorized nodes.
- **Privacy-preserving smart contract computation.** To the best of our knowledge, so far this is the most investigated issue, with the results of solutions exploiting different privacy-preserving methods (e.g., zero-knowledge proofs, secure multi-party computation, homomorphic encryption, etc). In the following, we provide an overview of these solutions, organized according to the exploited privacy-preserving method.
  Zero-Knowledge protocol. Zero-knowledge protocols allow a party P to provide to another party V a proof that he/she knows some facts (e.g., a secret value) without revealing to V any information about that facts. These protocols can be exploited to hide those during smart contract execution. As an example, in order to privately process data in a blockchain, [15] presents a privacy-preserving decentralized smart contract system, called $Hawk$, that relies on a zero-knowledge protocol (zk-

[8] https://lightstreams.network/

[9] https://github.com/autocontracts/permissioned-blocks/blob/master/whitepaper.md

	Transaction confidentiality	**Privacy-Preserving SC computation**	**SC instructions confidentiality**	**Selective access to SC data**	**Oracle data confidentiality**	**Oracle correct flow**
Encrypter [8]		✓		✓	✓	✓
Hawk [15]		✓				
Town Crier [16]		✓			✓	✓
Ekiden [17]	✓	✓	✓	✓		
Enigma [18]		✓		✓		
Lightstream [19]	✓	✓				
Raziel [20]		✓				
PDO [21]	✓	✓	✓	✓		
[22]–[26]	✓					
[27]–[29]		✓				
[30]	✓	✓	✓			
Oracle solutions (e.g., Oraclize, Reality Keys)						✓

TABLE I
SUMMARY OF ONGOING PROPOSALS W.R.T. SECURITY ISSUES

SNARKs). The system has a trusted off-chain node that executes contracts and posts on-chain only zero-knowledge proofs.
Other solutions exploiting zero-knowledge protocols for achieving confidentiality of smart contract computation have been proposed, in [27], [28], [29].
Secure multi-party computation. This term points to cryptographic schemes enabling a set of participants to jointly compute the result of a function over their inputs without revealing them. An example of solution exploiting secure multi-party computation in blockchain is given in [18], where a framework, called *Enigma*, for the execution of "private smart contract" on blockchain has been proposed. The key features of enigma are: (1) a secure multi party computation to ensure confidentiality of data involved in the smart contract execution; (2) an off-chain component carrying out the privacy-preserving protocol.
In [20] is presents another approach, called *Raziel*, that combines multi-party computation and zero-knowledge proofs to guarantee privacy, correctness and verifiability of smart contracts. Through the exploitation of zero-knowledge proofs, Raziel is able to create certificate, called Proof-Carrying Code, that can be exploited by smart contract owners to prove smart contract validity to third parties, without revealing any information about the source code.
Homomorphic Encryption scheme. In general, homomorphic encryption schemes are defined such that, given two encrypted values `E(a)` and `E(b)`, the computation of an operator $\oplus$ on them, i.e., `E(a)`$\oplus$`E(b)`, generates an encrypted value `E(c)`, whose decryption corresponds to the result of $\oplus$ applied on the plaintexts `a` and `b`, i.e., `E(a)`$\oplus$ `E(b)=E(c)`, where `a`$\oplus$`b=c`. By allowing computation on encrypted values, homomorphic encryption schemes could be exploited for ensuring confidentiality of data processed by smart contracts. This has been done, as an example, in [8]. Here, the goal of the proposed framework is the deployment of a composition of web services (aka inter-organizational processes) over the blockchain, having the smart contract acting as process coordinator. At this aim, the framework consists of an off-chain process, called *Encrypter*, that translates the BPEL document into a smart contract, where instructions on sensitive data (e.g., user's credential, input/output service parameters) are replaced with privacy-preserving computation. This is achieved exploiting homomorphic encryption [39], which ensures the confidentiality of data exploited by smart contracts. As it will discussed later on, [8] also ensures selective access on data.
Trusted Execution Environment - TEE. In general, this term refers to an isolated execution environment where data can be securely processed, ensuring both their confidentiality and integrity. TEE can be achieved thanks to embedded hardware technologies (e.g., Intel Secure Guard Extensions - SGX).[10] Recently some proposals exploit TEE as a support for privacy-preserving computation of smart contracts. A relevant proposal in this direction is *Ekiden* [17]. Ekiden provides a programming model for highly performing and confidentiality-preserving smart contract execution that also enhances transaction confidentiality. The key component of *Ekiden* is the integration of blockchain with the Intel software guard extension (SGX) as TEE. As such, Ekiden adopts an architecture where computation is separated from consensus. It uses off-chain computing nodes to perform smart contract computation on TEEs, then it attests their

[10]https://software.intel.com/en-us/sgx

correct execution on chain. The underlying blockchain is maintained by consensus nodes, for which it is not required to have TEE. Another interesting approach is the one in [30], where an Hyperledger Fabric extension has been proposed for the execution of smart contract (called chaincode) on private data using TEE (Intel SGX) in order to guarantee privacy preserving on smart contract. Moreover, the extension exploits the so called ”channel”, which defines a private network for a subset of the members to make transactions that ensure privacy and confidentiality.

- **Smart contract instructions confidentiality.** To the best of our knowledge, currently this issue is only addressed by exploiting TEE. Confidentiality of the smart contract code and data is provided by entirely executing the smart contract inside a trusted environment (Enclave), i.e., the protected areas of execution in the memory. For instance, [21] proposes an interesting approach exploiting “Private Data Objects” (PDO), that is an integration of distributed ledger with Intel SGX. The approach performs smart contract execution inside an SGX enclave, protecting integrity and confidentiality of SC code and data.
- **Selective access to smart contract data**. It is interesting to note that all proposals offering off-chain TEE nodes for ensuring privacy-preserving smart contract computation (i.e., [17], [18], [30]) can exploit TEE features to also ensure a selective access on data involved in the process. Indeed, by design, TEE provides mechanisms for fine-grained access control on data involved in process executed in the trusted enclave. However, to the best of our knowledge, the only work addressing this issue without TEE support is the one presented in [8]. Selective access is enforced during the service composition (i.e., smart contract execution) by ensuring that services' input/output parameters, as well as users' credentials, are consumed only by authorized services. To achieve this, data contained in smart contracts are selectively encrypted exploiting different keys, such that each piece of data can be decrypted only by authorized services (aka services provided with the corresponding key). As described previously, by exploiting homomorphic encryption, [8] allows computation on these encrypted data.
- **Oracle data confidentiality**. This issue implies to ensure confidentiality of data sent to or retrieved by Oracles. To address this problem, an interesting approach has been proposed in [40]. Here, authors present *Town Crier*, a framework relying on TEE able to ensure an authenticated and confidential data feed from/to an Oracle. Moreover, [8] addresses this issue without relying on a trusted hardware, as it encrypts each data sent to the Oracle (i.e., service parameters) such that only the invoked service can decrypt them.
- **Oracle correct flow**. This requirement asks for a proof that the Oracle mediator has indeed retrieved the data from the service inquired by the smart contract. This is a relevant requirement that existing Oracle commercial products already address. As an example, both Oraclize[11] and Reality Keys[12] provide cryptographic proofs of correct data inquiry to guarantee their honest behavior. However, these solutions do not consider confidentiality of parameters passed for service invocation.

V. Conclusion and future research directions

Blockchain is nowadays recognized as a technology with the potential to lead to a drastic revolution in economic and business models. Its features (e.g., transparency, accountability, consensus-based verification, smart contracts execution) make this platform suitable for a wide range of applications, among which the one of inter-organizational business process execution. In this paper, we have first discussed how a business process could be deployed over blockchain, i.e., having the collaborating organizations exposing their services and a smart contract monitoring the overall service composition. We have also highlighted the security issues that should be properly addressed when dealing with inter-organizational business process execution on the blockchain, by presenting the existing proposals able to partially address some of them. Even if these solutions represent a good starting point, witnessing of the interest that blockchain is gaining from security research groups, these are not enough. These solutions have been designed mainly to cope with the problem that on blockchain data is public (e.g., transactions, smart contract). In contrast, they do not deal with the issues that an inter-organizational process, by its collaborative nature, intrinsically implies. Similarly to other business-to-business collaborations, indeed these processes demand for *mechanism enabling a selective, fine-grained, temporal access to shared resources in blockchain, by complying with ordering enacted by the underlying workflow*. This cannot be reached easily exploiting the existing proposals; rather a comprehensive framework able to support a secure workflow-based collaboration on blockchain has to be designed.

Moreover, as highlighted by existing proposals, the smart contract runtime environment[13] poses a restriction on the exploitation of computationally intensive cryptographic schemes. To overcome this limitation, the proposed solutions exploit off-chain components with/without hardware-based TEE. However, when dealing with an inter-organizational process the target scenario should be the one of SOA. Here, services are discovered, composed and deployed on the fly based on the needs. In such a dynamic scenario, the design of a comprehensive solution should limit the assumptions on organization environments (e.g., the presence of TEE).

Furthermore, the design of such a comprehensive framework should also keep into account the type of blockchain over which deploying the process (i.e. public, private, permissoned, permissionless). Indeed, while it is evident that all the discussed security issues have to be addressed when dealing with

[11]http://www.oraclize.it/

[12]https://www.realitykeys.com/

[13]The smart contract runtime environment is represented by the isolated virtual machine on which blockchain nodes execute a smart contract

public blockchains, the contrary is not so obvious. Indeed, with a permissioned blockchain more control on which nodes have to be involved in the consensus (aka smart contract validation) can be introduced. As a consequence, the presence of trusted authorized nodes could help in relaxing some data confidentiality issues. But it does not completely solve the problem when dealing with a process involving different collaborating entities. Indeed, in this setting, each organization might have different criteria to identify the set of authorized nodes. Therefore there is the need of investigating a strategy for *blockchain governance*. This implies the definition of high-level security policies able to state, in a permissioned blockchain, which nodes are authorized to access, which portions of data (e.g., transactions and smart contract code) by complying, at the same, with the workflow underlying the collaboration.

REFERENCES

[1] M. Swan, *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *Ethereum white paper*, 2014. [Online]. Available: https://ethereum.org/

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger. ethereum project yellow paper 151 (2014)," 2014. [Online]. Available: http://gavwood.com/paper.pdf

[4] L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber, "Optimized execution of business processes on blockchain," in *International Conference on Business Process Management.* Springer, 2017, pp. 130–146.

[5] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted business process monitoring and execution using blockchain," in *International Conference on Business Process Management.* Springer, 2016, pp. 329–347.

[6] O. López-Pintado, L. García-Bañuelos, M. Dumas, and I. Weber, "Caterpillar: A blockchain-based business process management system," in *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain*, 2017.

[7] J. Mendling, I. Weber, W. Van Der Aalst, J. v. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. Di Ciccio, M. Dumas, S. Dustdar *et al.*, "Blockchains for business process management-challenges and opportunities," *arXiv preprint arXiv:1704.03610*, 2017.

[8] B. Carminati, C. Rondanini, and E. Ferrari, "Confidential business process execution on blockchain," in *2018 IEEE International Conference on Web Services (ICWS)*, July 2018, pp. 58–65.

[9] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[10] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International Workshop on Open Problems in Network Security.* Springer, 2015, pp. 112–125.

[11] C. Peltz, "Web services orchestration and choreography," *Computer*, vol. 36, no. 10, pp. 46–52, 2003.

[12] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte *et al.*, "Business process execution language for web services," 2003. [Online]. Available: http://xml.coverpages.org/BPELv11-20030505-20030331-Diffs.pdf

[13] C. Prybila, S. Schulte, C. Hochreiner, and I. Weber, "Runtime verification for business processes utilizing the bitcoin blockchain," *Future Generation Computer Systems*, 2017.

[14] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar *et al.*, "Blockchains for business process management-challenges and opportunities," *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 1, p. 4, 2018.

[15] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Security and Privacy (SP), 2016 IEEE Symposium on.* IEEE, 2016, pp. 839–858.

[16] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security.* ACM, 2016, pp. 270–282.

[17] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution," *arXiv preprint arXiv:1804.05141*, 2018. [Online]. Available: https://arxiv.org/pdf/1804.05141.pdf

[18] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *arXiv preprint arXiv:1506.03471*, 2015.

[19] 2018. [Online]. Available: https://lightstreams.network/

[20] D. C. Sánchez, "Raziel: private and verifiable smart contracts on blockchains," *arXiv preprint arXiv:1807.09484*, 2018.

[21] M. Bowman, A. Miele, M. Steiner, and B. Vavala, "Private data objects: an overview," *arXiv preprint arXiv:1807.05686*, 2018.

[22] G. Maxwell, "Confidential transactions," *URL: https://people. xiph. org/~greg/confidential_values.txt*, 2015.

[23] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy (SP).* IEEE, 2014.

[24] N. Van Saberhagen, "Cryptonote v 2.0," 2013. [Online]. Available: https://bravenewcoin.com/assets/Whitepapers/CryptoNote-V-2.0-whitepaper-annotated.pdf

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

[26] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Security and Privacy (SP), 2013 IEEE Symposium on.* IEEE, 2013, pp. 397–411.

[27] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *International Conference on Financial Cryptography and Data Security.* Springer, 2017, pp. 357–375.

[28] B. Stein, K. Kuznecov, S. Lee, and J. Müller, "A public blockchain solution permitting secure storage and deletion of private datadraft," 2018. [Online]. Available: https://www.lition.io/

[29] T. Espel, L. Katz, and G. Robin, "Proposal for protocol on a quorum blockchain with zero knowledge," 2017. [Online]. Available: https://eprint.iacr.org/2017/1093.pdf

[30] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric," *arXiv preprint arXiv:1805.08541*, 2018. [Online]. Available: https://arxiv.org/pdf/1805.08541.pdf

[31] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "Fairaccess: a new blockchain-based access control framework for the internet of things," *Security and Communication Networks*, pp. 5943–5964, 2016.

[32] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the internet of things," *arXiv preprint arXiv:1802.04410*, 2018. [Online]. Available: https://arxiv.org/pdf/1802.04410.pdf

[33] O. Novo, "Blockchain meets iot: an architecture for scalable access management in iot," *IEEE Internet of Things Journal*, 2018.

[34] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, "Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *Journal of Network and Computer Applications*, vol. 116, pp. 42–52, 2018.

[35] A. Ramachandran and M. Kantarcioglu, "Smartprovenance: A distributed, blockchain based dataprovenance system," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy.* ACM, 2018, pp. 35–42.

[36] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

[37] H. M. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intelligent Systems in Accounting, Finance and Management*, vol. 25, no. 1, pp. 18–27, 2018.

[38] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 2017, pp. 468–477.

[39] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[40] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proc. of the ACM Conference on Computer and Communications Security, CCS*. New York, NY, USA: ACM, 2016, pp. 270–282.