Balancing Confidentiality and Transparency for Blockchain-based Process-Aware Information Systems

Alessandro Marcelletti[1],
Edoardo Marangone[2], Michele Kryston[3], and Claudio Di Ciccio[3]

[1] University of Camerino, Camerino, Italy, alessand.marcelletti@unicam.it
[2] Sapienza University of Rome, Rome, Italy, edoardo.marangone@uniroma1.it
[3] University of Utrecht, Utrecht, The Netherlands, m.kryston@uu.nl,c.diciccio@uu.nl

Abstract. Blockchain enables novel, trustworthy Process-Aware Information Systems (PAISs) by enforcing the security, robustness, and traceability of operations. In particular, transparency ensures that all information exchanges are openly accessible, fostering trust within the system. Although this is a desirable property to enable notarization and auditing activities, it also represents a limitation for such cases where confidentiality is a requirement since interactions involve sensitive data. Current solutions rely on obfuscation techniques or private infrastructures, hindering the enforcement capabilities of smart contracts and the public verifiability of transactions. Against this background, we propose CONFETTY, an architecture for blockchain-based PAISs to preserve confidentiality and transparency. Smart contracts enact, enforce and store public interactions, while attribute-based encryption techniques are adopted to specify access grants to confidential information. We assess the security of our solution through a systematic threat model analysis and evaluate its practical feasibility by gauging the performance of our implemented prototype in different scenarios from the literature.

Keywords: Business process management · Distributed ledger technologies · Blockchain · Attribute-based encryption · Security · Privacy

1 Introduction

Blockchain enables new forms of trustable Process-Aware Information Systems (PAISs) and inter-organizational collaborations [27]. This is possible thanks to the characteristics of public permissionless blockchains, which provide strong guarantees removing the need for third-party authorities [22]. Starting from the foundation, the distributed ledger and consensus mechanisms permit distributed and immutable information storage. These features, combined with the cryptographic primitives, ensure security properties and accountability of recorded operations. Smart contracts are programs immutably stored inside the blockchain that can encode and enforce business logic. On top of this, transparency is a key aspect, as information stored in the ledger and performed operations are accessible to everyone within the system.

Business logic enforcement and transparency are highly desirable properties for collaborative process execution and monitoring [10]. However, public observability of the whole set of exchanged data hinders the adoption of blockchain-based solutions in contexts where confidentiality is a critical requirement. Striking a balance between these two aspects is

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

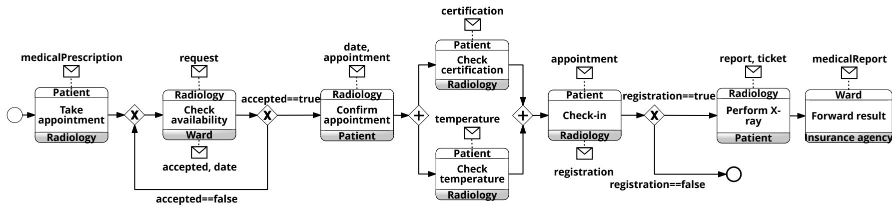


Fig. 1: BPMN choreography diagram of an X-ray diagnostic analysis [7]

crucial in highly regulated sectors wherein sensitive data are treated (consider, for instance, the domains of the pharmaceutical supply-chain and healthcare [11]). Existing solutions to solve this conundrum employ private permissioned blockchains [30,6]. Nevertheless, these solutions may hide data from auditors and assume high robustness and security guarantees of the consortium's system. Other approaches propose encryption techniques to restrict visibility of data stored on-chain only to authorized parties [15,16,20,21]. However, their integration with process management systems is limited or non-existent.

Against this background, we propose an approach and an architecture for **blockchain-based process enactment that preserves the confidentiality of exchanged information while keeping public enforcement and transparency of process execution.** We name our approach CONFidentiality EnforcemenT TransparencY (CONFETTY). In particular, we rely on smart contracts to encode and execute business process logic while logging the interactions between parties, and we resort to Multi-Authority Attribute-Based Encryption (MA-ABE) [4] to control the access of different parties to the activities' data payloads and information artifacts, thus safeguarding confidentiality. We demonstrate the security of our approach against a threat model for the proposed architecture. We implemented our artifact and evaluated the feasibility of our solution by showing its application in the context of a healthcare scenario taken from the literature. Also, we analyzed the execution performance with collaborative processes presented in the related scientific literature.

The remainder of the paper is organized as follows. Section 2 illustrates the example and highlights the motivation for this work. Section 3 provides the analysis of the state of the art and introduces the fundamental concepts on which CONFETTY relies. Section 4 describes the proposed architecture. Section 5 evaluates our solution and provides an assessment of its security and feasibility. Finally, Sect. 6 concludes the work and draws future research directions.

2 Example, Problem Illustration and Requirements

To illustrate the problem we aim to tackle, we introduce a running example in the healthcare domain inspired by [7]. Figure 1 depicts the management process for X-ray diagnostics in the form of a BPMN choreography diagram. The patient first presents the medical prescription, asking for an appointment. The radiology department checks the availability of the ward, and if a date is available, the appointment is confirmed. Otherwise, there is a new tentative. After obtaining the registration details, the temperature and vaccine certification of the patient are verified. A radiology clerk checks whether the appointment can be confirmed. If so, the X-ray exam is done, and the results are provided to the ward with a final report. In the end, an

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

Table 1: Requirements and corresponding actions in the approach

	Requirement	Approach
R1	Public enforcement and transparency of process execution should be guaranteed	The control-flow of the process is managed by smart contracts deployed on public blockchain
R2	The process should be independently auditable with low overhead while guaranteeing authenticity	On-chain information retains publicly available process execution tracking information and hashes with locators for associated encrypted data
R3	Information artifacts should be written in a permanent, tamper-proof and non-repudiable way	Messages are stored in a tamper-proof distributed off-chain file storage, and the resource locators are stored on-chain to keep track of information
R4	Access to information artifacts should be controlled while ensuring their overall integrity and availability	MA-ABE encrypts information artifacts stored in a distributed environment. Decryption is possible only if the requester holds the necessary attributes.
R5	User-defined policies should control access levels for authenticated users with a fine-granular scheme	MA-ABE policies are associated with messages within individual activities

insurance agency receives the patient's results. The insurance agency can access the patient's medical report to follow up with compensatory actions in the subscribed contract (e.g., the reimbursement of medical bills). Beyond the boundaries of the choreography diagram in Fig. 1, we assume the presence of an inspector of the local Ministry of Health. They are not among the process participants, yet they should have full access to the information exchanged even after the conclusion of the instance for auditing purposes.

Table 1 lists the requirements derived from the above use case. To begin with, ensuring public execution and tracking of healthcare procedures is critical to improving process efficiency and trust in the public towards the healthcare system. We concretize this aspect in Req. R1. Enabling auditing of those procedures is crucial since it permits investigations on misconduct or lack of compliance with regulations (Req. R2). This is allowed by the data-based evidence, which should be ensured by information storage in permanent, tamper-proof and non-repudiable information artifacts (Req. R3). The guarantees offered by a public blockchain-based PAIS are fit for purpose from this standpoint. However, sensitive patient data must be shielded from unauthorized inspections (Req. R4). For instance, although the insurance agency is involved in the process, it should not have full access to all exchanged data (e.g., the vaccine certification). Publicly disclosing the result of the diagnostic analysis or information about vaccine certification would severely violate the privacy sphere of the patient. To prevent this, access should be restricted only to designated and authorized entities through a fine-grained access control mechanism (Req. R5). Balancing a secure, publicly verifiable and decentralized infrastructure with the need for data confidentiality remains an open challenge, which motivates our research.

3 Background and State of the Art

In this section, we first describe the core concepts underlying our approach, namely blockchain technology and Attribute-Based Encryption (ABE), and then illustrate how these pillars serve as the foundation for existing approaches in the literature relevant to our investigation.

3.1 Background

Blockchain technology. A ledger is an append-only singly-linked list of transactions, representing asset transfers among accounts. A blockchain collates ledger segments into blocks,

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

with headers storing block numbers and hashes linking to previous blocks, ensuring transaction order and data integrity. Selected nodes append new blocks and receive cryptocurrency as rewards. The blockchain is replicated across all nodes, ensuring transparency. Consensus protocols and cryptographic primitives enable public verification and decentralization without third-party oversight. Several platforms implement the blockchain protocol (e.g., Ethereum [31]). Some blockchains allow for the deployment of *smart contracts* [13] which are programs deployed onto the blockchain and executed within a dedicated virtual machine (e.g., the Ethereum Virtual Machine, EVM). Smart contracts functions are invoked through transactions and their core features are code immutability and invocation storage on the blockchain, making such information available for auditing purposes [10]. Smart contracts execution incurs a cost on the invoker. The cost is measured in gas units and depends on the complexity of the called function and the size of the exchanged data. To save on the latter, distributed tamper-proof storage systems are often employed to store the actual data in a non-modifiable file, while a permalink to that file is saved in the smart contract state. The InterPlanetary File System (IPFS) [1] is a typical example of such a system. Based on a peer-to-peer network wherein each node stores chunks of the data, IPFS binds every file with a string used both as a unique identifier and resource locator. The string is based on the hash of the file content itself so that if the file is altered, the permalink does not match. As we will see next, these features fostered the development of blockchain-based PAISs using smart contracts as the backbone for business (process) rule enforcing (in our example, the BPMN choreography is translated into smart contract code, and its steps correspond to invocations to its functions) [7,9].

Attribute-based Encryption. ABE is a public key encryption where the ciphertext, an encrypted plaintext, and the corresponding decryption key are linked through custom boolean attributes [25]. In Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [2], a variant of ABE, every user bears a set of attributes encoded in their key. For example, the owner of the blockchain account `0xB0...1AA1` can be endowed with attributes `Patient` and `PID476948` specifying role and process instance 476948, respectively. Data owners write policies to be attached to the CP-ABE-encrypted document (the *ciphertext*). Those policies express conditions for granting in-clear access to the data. Policies are propositional logic formulae employing ABE attributes as literals. The policy is evaluated on attributes associated with a potential reader: Only if the formula is satisfied can the user's key decrypt the data. For instance, the key of the user `0xB0...1AA1` can decrypt data locked with a policy like `PID476948 and PATIENT`, but cannot unlock the contents of a document associated with a policy like `PID476949 and RADIOLOGY`. Attributes are typically assigned by one authority: Multi-Authority Attribute-Based Encryption (MA-ABE) [4] is a variant of ABE that removes this single point of failure using a multi-authority method. In MA-ABE, every authority creates a part of the decryption key. Once collected the user merges all parts to obtain the final key. In this paper, we employ the Ciphertext-Policy variant of MA-ABE [17]. It integrates MA-ABE with the aforementioned policy-based approach.

3.2 State of the Art

Over the last few years, several solutions that automate collaborative processes using blockchain technology have been designed [27,9,18,29]. Such approaches demonstrated the effectiveness of blockchain-based solutions in improving trust among participants in multi-party

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

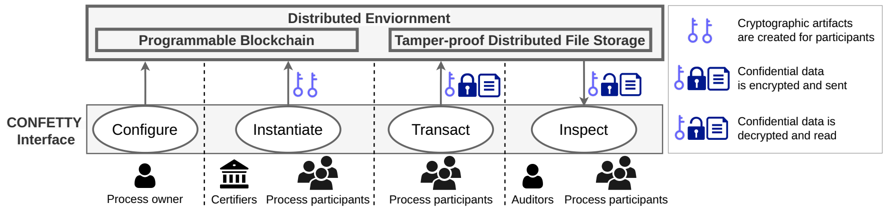


Fig. 2: A graphical sketch of the CONFETTY functionalities, users, artifacts, and interfaces

collaborations and allowing for monitoring and auditing [10,8]. These studies enhance the integration of blockchain technology with process management, unlocking security and traceability benefits. However, they primarily focus on the control-flow perspective and lack mechanisms for secure access control to the stored data on public platforms.

A relevant area of research for our work pertains to data privacy and integrity within blockchain systems. Several studies examine the application of encryption techniques to achieve this goal. Hawk [14] employs user-defined private smart contracts to automate the implementation of cryptographic protocols. Pham et al. [24] propose a decentralized storage solution that integrates InterPlanetary File System (IPFS), MA-ABE, and blockchain technology. Hong et al. [12] introduce a data-sharing solution based on decentralized ABE (MA-ABE), blockchain, and IPFS. Bramm et al. [3] present BDABE, an access control mechanism that applies a distributed ABE scheme within a consensus-driven infrastructure for real-world applications. Yan et al. [33] introduce a scheme for fine-grained access control by implementing proxy encryption and decryption while supporting policy hiding and attribute revocation. The encrypted data is stored on IPFS, and the metadata is stored on the blockchain. All these studies, along with the aforementioned [21,20,19], cater for secure access control on blockchains but lack the integration of this technology with process management systems.

To the best of our knowledge, this is the first work that focuses on combining blockchain-based process execution engines with encryption schemes that guarantee data confidentiality and access control over data shared on public blockchain platforms. In the next section, we provide a description of our solution.

4 The CONFETTY Architecture

In this section, we describe the proposed CONFETTY architecture. We first introduce its core components and functionalities (Sect. 4.1). Then, we describe how components interact and manage information to support confidential-preserving blockchain-based process execution while meeting Reqs. R1 to R5 (Sect. 4.2). Due to space restrictions, we do not enter the technical details of our approach but provide an overarching presentation of our solution's rationale and key concepts, motivating them with the requirements they aim to meet.

4.1 Functional Viewpoint

Figure 2 provides an overview of the CONFETTY approach, which caters for the following main functionalities. *(i)* The **configure** functionality helps the process owners specify the

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

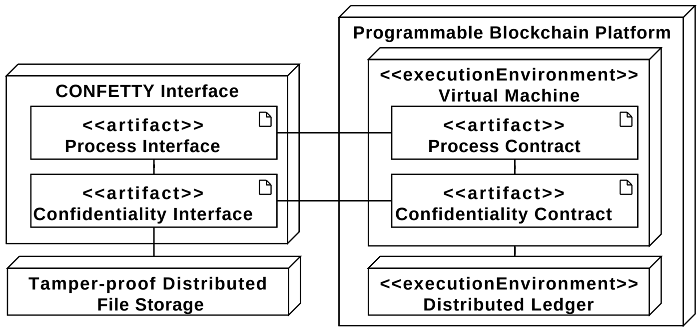


Fig. 3: The deployment diagram of CONFETTY

process to run on-chain. It requires the registration of process information, including activities to execute, data to exchange, control-flow routing, the expected participants' roles (like the information enclosed in the choreography diagram in Fig. 1) and how they can operate with confidential data (via MA-ABE policies) to control information access. *(ii)* The **instantiate** functionality lets certifiers authenticate the participants, define their role in the process (e.g., that user `0xB0...1AA1` is a patient), and kickstart a new process instance on-chain by setting its public state. *(iii)* The **transact** functionality allows process participants to interact, following the process, to exchange data at run time. This functionality manages both public and confidential data, updating and enforcing the process state according to the logic in the blockchain. *(iv)* The **inspect** functionality allows participants and auditors to access public and confidential data.

The functionalities above are realized by the CONFETTY main components depicted in Fig. 3. CONFETTY relies on two external components used as architectural buttresses: *(i)* A **Programmable Blockchain Platform** (like Ethereum), *(ii)* and a **Tamper-proof Distributed File Storage** (like IPFS). The former maintains the public state, allowing for the creation of logic and constraints while notarizing the executed operations. It comprises two core elements: The Distributed Ledger, storing the transactions, and the Virtual Machine, hosting and running smart contracts. The latter (which we will henceforth name as Distributed FS for brevity) stores business data and thus improves the performance and costs of a blockchain as in common practice. The data is saved on the external storage (upon encryption, if it must remain secret), while a resource locator with the hash of that data is kept on-chain for notarization and future retrieval. Notice that we take inspiration from known patterns adopting Distributed FSs for large data storage [32]. Based on this known pattern, we devise new techniques to preserve secrecy and control access to that process data at runtime. The primary components of CONFETTY are the following: *(i)* The **Process Interface** handles all the operations involving the participants and related to the process. This is done by acting as an interface for the blockchain, abstracting from the technical implementation and exposing functionalities to the users. In particular, the Process Interface instantiates the process on the blockchain on given process specification data. To support the *instantiate* functionality, it provides capabilities for role assignment to participants. *(ii)* The **Process Contract** is a smart contract that manages the process instances and their execution. It elaborates and manages all process specifications and their instances' public states, acting as a software realization both of factory and proxy patterns [32]. It supports the *instantiate* functionality by enforcing the control flow, the data

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

Table 2: Access policies. `$PID` represents a placeholder for the identifier of the process instance before its assignment

Message	Access policy
Medical prescription	`MINISTRY-INSPECTOR or ($PID and (PATIENT or RADIOLOGY))`
Registration	`MINISTRY-INSPECTOR or ($PID and (RADIOLOGY or PATIENT))`
Report	`MINISTRY-INSPECTOR or ($PID and (RADIOLOGY or WARD))`

flow, the assignment of tasks to participants, and the routing of decision points. In this way, the Process Contract updates the public state of the process and generates transparent records that are accessible and verifiable by interested auditors. *(iii)* The **Confidentiality Interface** is in charge of handling the participants' authorizations and the operations over confidential data. It operates as a façade towards the blockchain to store access grants, distributes decryption keys to the users solely based on their roles and involvement in the process, and handles the storage and retrieval of encrypted data. *(iv)* The **Confidentiality Contract** is a smart contract handling the notarization of authorizations for confidential data, as well as for their writing and access.

We conclude this subsection with a few considerations about the users and trust model. We assume that the users we mentioned while discussing the functionalities (i.e., process owners, certifiers, process participants, and auditors) possess a blockchain account with a key pair to be able to publicly identify themselves (as with the aforementioned user `0xB0...1AA1`) and sign blockchain transactions built through the CONFETTY modules. We also remark that we consider the CONFETTY platform as a trusted party, while users are assumed to be *honest but curious*. Finally, notice that the involvement of a pre-appointed third-party certifier to authenticate process participants is necessary as a user alone can confirm their identity (e.g., via the aforementioned public-private key scheme) but cannot attest alone to the truthfulness of assertions regarding themselves. The selection and appointment of certifiers transcends the goals of this paper, but in our motivating scenario, e.g., certifiers can be trusted third parties like the Public Ministry of Health (for hospital personnel), the Public Registry (for citizens), and the Insurance Registrar (for agencies), or subsidiaries thereof. In our architecture, certifiers operate like blockchain oracles [23].

4.2 Information Viewpoint

After providing an overview of the CONFETTY architecture, we describe how information is handled in realizing the aforementioned functionalities by the CONFETTY components.

The first functionality is *configure*. Here the process owner sends the process specification files (like the diagram in Fig. 1) to the Process Interface. The Process Interface, in turn, processes the input data, storing the behavioral and business constraints it reports. Also, it builds a set of parametric access policies, associating roles of the process with the data they can access via encryption directives (meeting Req. R5). Once created, the policies are returned to the process owner. The one associated with the medical prescription (sent by the patient to the radiology clerk to take an appointment in Fig. 1), e.g., reads "`$PID and (PATIENT or RADIOLOGY)`". Here, `PATIENT` and `RADIOLOGY` are the attributes representing the actors' roles in the process. Not all patients and radiology clerks should access any medical prescription. Hence `$PID`, an attribute placeholder for the process instance identifier (a piece of information that can be known only at runtime), restricting access to the sole

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

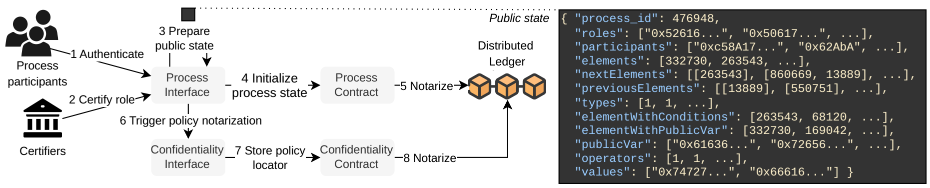


Fig. 4: Communication diagram of the *instantiate* functionality, with a snapshot of public state

actors involved in that process instance. We give the process owner the option to revise the policies by adding custom roles that are not expected active parties in the process but need access to data (as per Req. R2). This is the case, for instance, of external auditors who need to certify data and its compliance with norms, laws and regulations. Considering the medical prescription, the revised policy specifies that a user has to be attested to have the role of a `MINISTRY-INSPECTOR`, or again to be a `PATIENT` or `RADIOLOGY` within the running process instance: "`MINISTRY-INSPECTOR or ($PID and (PATIENT or RADIOLOGY))`". Table 2 shows an extract from the final access policies of our running example. Once policies are confirmed, they are sent to the Confidentiality Interface, which stores them in the Distributed FS. Their permalink is passed back to the Confidentiality Interface to save it.

Figure 4 depicts the *instantiate* functionality. Differently from the *configure*, needed once for each process, this one is designed for every new instantiation thereof. To initialize a new process instance (e.g., number 476948), participants send their data to the Process Interface (1) to authenticate themselves as participants in the process with their role (e.g., that user `0xc5...Fd43` operates as a `WARD` clerk in a new process instance `PID476948`). (2) The authentication is attested to by certifiers confirming the user's self-declared roles. When all process participants are identified and associated with a role, (3) the Process Interface merges the received information with the process data acquired in the configuration. Thereupon, (4) a selected participant kickstarts a new process instance through the Process Interface associated with the aforementioned information on the Process Contract, thereby initializing the instance's public state. This step is done by sending a transaction to the Process Contract with the public state of the process, including conditions driving the control flow, such as exclusive gateway choices (as per Req. R1). Notice that this crucial step is notarized on chain (5) (meeting Req. R2). The selection of the participant, or participants, on behalf of whom the transaction is signed, transcends the scope of this paper and depends on the decision process and guidelines of the consortium. After initializing the process state, (6) the Process Interface triggers the Confidentiality Interface, which (7) stores the policy locator previously saved on the Confidentiality Contract. Finally, (8) the Confidentiality Interface notarizes the operation.

Figure 5 displays the *transact* functionality, which represents the exchange of information during the execution of an activity in the process instance. It encompasses a public and a confidential case, depending on the process participants and their willingness to classify information as disclosable or not. In both cases, this functionality updates the public state in the blockchain according to previous enforcement as per Req. R1 (e.g., activity to execute, expected participant) and manages accessibility (meeting Req. R4).

Let us begin with the exchange of public data (and the update of the public state). The main steps are highlighted in blue in Fig. 5 and salient on-chain operations are reported in

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-032-02929-4_14

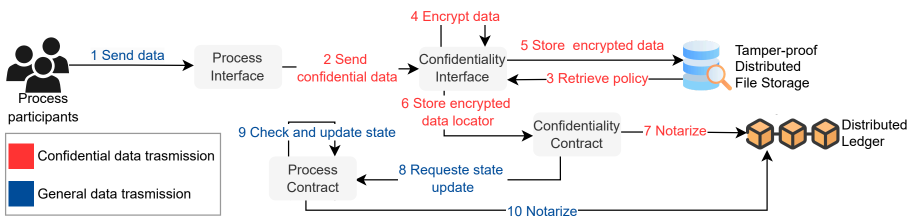


Fig. 5: Communication diagram of the *transact* functionality

Algorithm 1: On-chain logic for public state update and next element activation

```
1  Function updatePublicState(instanceID, messageID, variables):
2      authUser ← instances[instanceID].elements[messageID].role
3      element ← instances[instanceID].elements[messageID]
4      if sender is authUser & element.type is MESSAGE & element.state is ENABLED then
5          foreach variable in variables do instances[instanceID].publicVars[variable.name] ← variable.value
6          activateNext(instanceID, messageID)

7  Function activateNext(instanceID, messageID):
8      currentElement ← instances[instanceID].elements[messageID]
9      instances[instanceID].elements[messageID].state ← COMPLETED
10     foreach nextElement in currentElement.nextElements do
11         if nextElement is GATEWAY then executeGateway(instanceID, nextElement.ID)
12         else nextElement.state ← ENABLED
```

Alg. 1. Initially, a user sends the execution data to the Process Interface in the context of an activity (Req. R1). For example, the `WARD` clerk identified by `0xc5...Fd43` replies to the "Check availability" activity request by the `RADIOLOGY` user within process instance 476948. To do so, the clerk sends a "Check availability" response message wherein the `accepted` variable is assigned with `true` since the appointment is confirmed alongside an available `date`. The boolean answer (`accepted`) is public data that is used later to make control-flow decisions and advance the state of the process, defining the next activity to perform. In our scenario, the exclusive choice gateway is based on the value of the `accepted` variable. Then, the Process Interface invokes the Process Contract to request a state update, sending the execution data (Line 1). In turn, the Process Contract enforces the business and control-flow logic through the run of its routine (based on previously agreed upon and immutable code, see *configure* functionality), which checks (Line 4) if the executed activity is enabled and if the message sender (in this case, user `0xc5...Fd43`) is the expected one (a `WARD`), finally updating the state (Line 5) if the verification outcome is positive. This step advances also the state of the process (Line 6), updating the expected next element (here, "Confirm appointment") and participant (`RADIOLOGY`). In case the element is a gateway, its logic is automatically executed to advance the control flow of the process (Line 11). In the end, this piece of information is notarized into the Distributed Ledger (meeting Req. R2).

For the confidential case, we refer again to Fig. 5 (see the marks in red). Overall, (1) once the Process Interface has received the data, (2) it forwards it to the Confidentiality Interface. The latter (3) retrieves the policy from the Distributed FS, and (4) uses it to encrypt the data via MA-ABE (as per Req. R4). For example, considering Fig. 1 and Table 2, the medical prescription of the patient is encrypted with the following policy: `{MINISTRY-INSPECTOR`

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

or (PID476948 and (PATIENT or RADIOLOGY))}. In particular, the Confidentiality Interface retrieves the specific authorization for the data to be stored and embeds it in the encryption mechanism, ensuring that only authorized participants can access it. This step is needed only in confidential cases, as in public ones, information is directly stored on the blockchain and intentionally kept publicly readable. Once encrypted, (5) the Confidentiality Interface stores data in the Distributed FS, (6) its locator in the Confidentiality Contract, and (7) notarizes the operation in the Distributed Ledger (meeting Req. R2 and Req. R3). In this case, the resource locator that contains the encrypted medicalPrescription (e.g., LXBw...2DAx) is Qmo9...0weI. The execution of this routine by the Confidentiality Contract includes a call to the Process Contract (8) to request a state update enclosing the execution data. Following the example, this data is the activity being executed ("Check availability"), the user that performs it (e.g.,, PATIENT, along with their corresponding address 0x2e...6dd9), and the process instance (e.g.,476948). Then, (9) the Process Contract checks the current public state to update it. The operations involving the smart contracts (10) are notarized in the Distributed Ledger.

The *inspect* functionality pertains to the access to message data. The public data can be freely retrieved, while the confidential data is restricted to the sole authorized users included in the access policy provided during the encryption phase (see the *configure* functionality above) (Req. R5). To read public data, a process participant or an auditor may send a request to the Process Interface, which in turn retrieves the necessary information from the Process Contract and sends it back to the requester. This is the case, e.g., for the accepted field of the "Check availability" task. Reading confidential data requires more passages and a preliminary step. All users should request at least once (and at every attribute update) a decryption key, to be used for every message they are going to read. For the key request, the process participant (e.g., a WARD clerk) or an auditor (e.g., a HEALTHCARE-INSPECTOR) ask for a decryption key to the Confidentiality Interface. The latter forwards the request to the Confidentiality Contract, which is thus notarized on the Distributed Ledger. Then, the Confidentiality Interface generates the attribute-based key (or *a-b key* for short) based on the attributes of the user requesting it (e.g., WARD and PID476948, or HEALTHCARE-INSPECTOR) via MA-ABE, and sends it back. The aforementioned step enables the decryption of confidential data. Once the process participants have received the a-b key, they can use it to decrypt the confidential data. To do so, a user makes a request to the Confidentiality Interface. Then, the Confidentiality Interface retrieves the encrypted data from the Distributed FS and forwards it to the user. At this point, the latter uses the a-b key to decrypt the confidential data via MA-ABE. Only if the user's attributes satisfy the policy previously used to encrypt the data can the user's a-b key decrypt the confidential data (and the user inspects it in clear).

A note on decryption keys. We remark that an a-b key stays with the user and lasts for as long as their attributes are not updated (if a user is involved in a new process instance, say 476949, a new attribute gets associated with them: PID476949). The same key will be used to access all messages. For example, consider user 0xB0...1AA1 (who holds the attributes PID476948 and PATIENT). Per the policies in Table 2, their a-b key can decrypt the medical prescription and the registration, but not the report. The a-b key of the user 0xc5...Fd43 (PID476948 and WARD) can decrypt the report instead, but not the other two documents. Notice that a HEALTH-INSPECTOR, instead, can use their unique a-b key to access all the aforementioned documents regardless of the process instance they pertain to. The key, in other words, is associated with the user, and it is the sole instrument to access a document. Therefore, the Con-

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

fidentiality Interface does not need to generate a different decryption key for every document and give the same copy to multiple users, nor create numerous copies of the same document for different decryption keys (one per user), thus saving unnecessary data replications and key distributions, and ensuring integrity (Req. R4). Also, as the key stays with the users, and the documents are stored and notarized on tamper-proof external infrastructures (the Distributed FS and Blockchain Platform, respectively), the information remains available even should the CONFETTY system stop functioning (Req. R3). These aspects have an impact on security considerations, as we discuss next.

5 Evaluation

To assess the feasibility of the CONFETTY architecture, we perform a two-step evaluation. First, we analyze the CONFETTY threat model, evaluating it against the proposed architecture. Then, we describe the implemented prototype and analyze its performance. The source code of CONFETTY, alongside the experimental data we use and performance analysis results, are available at https://doi.org/10.5281/zenodo.15482587.

5.1 Threat Model

In this section, we identify potential security threats that could compromise data availability (Req. R2) and confidentiality (Req. R5) within our solution, and outline how the CONFETTY architecture addresses them. Our threat analysis is based on the STRIDE framework [26], a theoretical model that categorizes threats into six distinct groups: spoofing (i.e., impersonation of a legitimate entity), tampering (i.e., unauthorized modification of data to compromise its integrity), repudiation (i.e., the denial of having performed a particular action), information disclosure (i.e., the unauthorized exposure of sensitive data), denial of service (DoS, i.e., the disruption or degradation of system availability), and elevation of privileges (i.e., the unauthorized acquisition of higher-level access rights). We assume users and components to be honest but curious and trusted, respectively. The possible threats thus reside in the communication between them, subject to potential attacks (see Sect. 4.1). Figure 6 displays a Data Flow Diagram (DFD), in which the orange dotted lines indicate the trust boundaries crossing data exchange points between different perimeters, represented by black arrows connecting different elements. The opening side of each boundary indicates the trusted component in the information exchange.

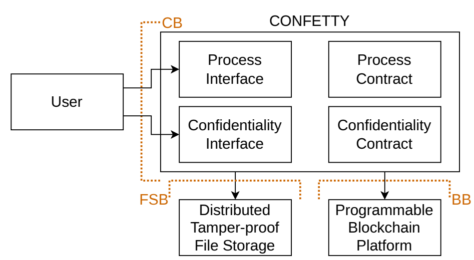


Fig. 6: The CONFETTY threat model, based on the components in Fig. 3

The **CONFETTY Boundary (CB)** indicates that CONFETTY is the trusted component in the communication with a user (let it be a certifier, a process participant or owner, or an auditor). In this case, a spoofing attack could result in a user impersonating another. Still, this attack is prevented since we assume that each user is linked to the Blockchain Platform

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-032-02929-4_14

credentials. As for tampering, trust is based on the fairness of the user who sends the data. It cannot be the case for a user to repudiate an action because all of them are recorded on the blockchain with a transaction signed with the user's private key. An attacker could sniff the data a user sends to CONFETTY. However, we resort to TLS channels for those communications, specifically designed to prevent information leakage [28]. DoS attacks are, in principle, feasible unless we pre-filter the range of potential clients invoking the CONFETTY platform. Indeed, all functionalities foresee an initial data input from users (see, e.g., step 1 in Fig. 5) which malicious nodes could exploit with multiple calls and bulky payloads. However, DoS attacks would lead to a disruption or degradation of the system's availability. Yet, all data, state, and other process instances' information can be retrieved from the Blockchain Platform and the Distributed Tamper-proof File Storage to reactivate or replicate the system without any data loss. Information disclosure is prevented because all sensitive information is encrypted via MA-ABE. Elevating a user's privileges is impossible since attributes and permissions are linked to a blockchain account.

The Blockchain Platform is the trusted component in the **Blockchain Boundary (BB)**. In this case, a spoofing attack could result in a malicious version of CONFETTY impersonating the original software object. This attack is prevented since CONFETTY operates as a gateway for transactions ultimately signed by users via their private keys. Information disclosure can be achieved only on public data or resource locators of sensitive data. However, the latter is stored upon MA-ABE-encryption, making its content unreadable by malicious users. Other possible attacks (DoS, tampering, repudiation, elevation of privileges) are counteracted by the design of blockchain protocols. As for the Distributed Tamper-proof **File Storage Boundary (FSB)**, tampering or DoS attack and elevation of privileges are counteracted by design, too. Spoofing and repudiation attacks are ineffective since the authorship of the provided information is proven by signatures of transactions stored on-chain. Since we resort to MA-ABE encryption, information disclosure is negated since the sensitive data stored is encrypted.

5.2 Implementation

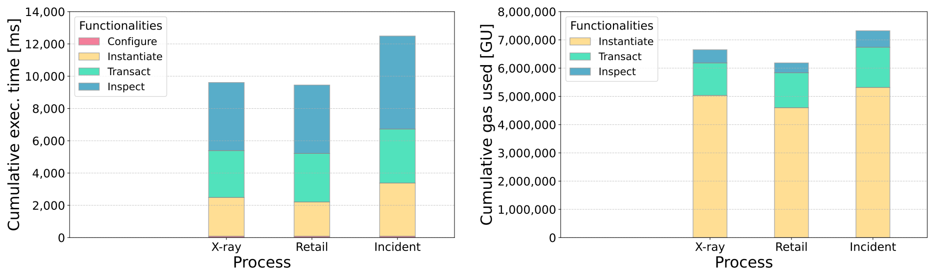
CONFETTY is deployed on three main tiers. An application server hosts the Confidentiality Interface and the Process Interface, developed in Python 3.6.9 and Java SDK-13, respectively. Users communicate with it via TLS communication channels [28]. We utilize IPFS[4] as the Distributed Tamper-proof File Storage. We employ Sepolia and Ganache as Programmable Blockchain Platforms with their Ethereum Virtual Machine (EVM) as a test environment and local RPC, respectively, connected via Web3.js[4].

5.3 Execution Analysis

Motivated by Req. R2, we evaluated our prototype in terms of time and cost according to the four key functionalities of CONFETTY (see Sect. 4.1). We set the number of MA-ABE authorities to four and performed the experiments on a machine equipped with a processor Intel(R) Core(TM) i7-13700H CPU @ 2.40 GHz with 14 cores and 32 GB of RAM. To isolate the architecture functionalities, we excluded human interactions and the time needed

[4] IPFS: ipfs.tech/; Sepolia: sepolia.dev; Ganache: archive.trufflesuite.com/ganache; Web3.js: web3js.readthedocs.io/en/v1.10.0/. Accessed: September 17, 2025.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14



(a) Time analysis

(b) Cost analysis

Fig. 7: Performance comparison

for blockchain-specific operations. Such operations strictly depend on the blockchain protocol and the platform adopted, representing technical choices that are out of the scope of this work.

Execution Performance and Expenditure. We gauged the time and cost of our prototypical implementation against data simulating the execution and message exchange of processes from the literature. Figure 7 illustrates the results, averaging over runs that took the longest path. We gathered data using a local RPC as a stub for the Programmable Blockchain Platform.

Firstly, let us focus on simulations of our motivating scenario (Sect. 2), which we will henceforth name X-ray process. For the time evaluation, we ran an instance for the longest path five times. We employed the Sepolia testnet as the Blockchain Platform and passed input data based on real-world files in the healthcare domain upon anonymization (available in our codebase). The *configure* functionality turns out to be the fastest, requiring an average of 71 ms. For the *instantiate* functionality, we let four process participants interact with the tool as specified in the choreography and it averaged 2689 ms. This higher amount depends on the need to authenticate process participants and prepare the data to send on-chain.

For the *transact* functionality, the runtime required an average of 2476 ms in total, with the encryption of single messages averaging 247 ms. Finally, the *inspect* functionality required an average of 4390 ms including the key request management. However, notice that this step was performed only once and required 2695 ms. Once the key is received, the single reading and decryption operations for a message averaged 169 ms. Overall, if compared to the expected time for Ethereum block creation (around 12 s), the overhead introduced by CONFETTY and related operations (e.g., encrypt and decrypt of message payloads) is orders of magnitude smaller (milliseconds) and can therefore be considered acceptable. To analyze the cost, we considered the sole operations that involve the blockchain, thus, *instantiate*, *transact*, and *inspect*. The highest cost is associated with the *instantiate* functionality, consuming around 5 million Gas Units (henceforth, GU), which corresponds to the total amount for the instantiation of the public state (2931698 GU) and other related operations (each averaging 90577 GU). *Transacting* consumed 1689190 GU in total for all the information exchanges, with a cost for a single transaction averaging at 115096 GU. Another relevant cost is the deployment of smart contracts, consuming around 5 million GU. However, this operation is performed only once during the bootstrapping of CONFETTY and does not pertain to single

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

process instantiations. Finally, *inspect* required 89535 GU for the key request management. The reading operations do not produce transactions, thus incurring no fees. In total, the execution of the X-ray process, without considering the deployment, consumed a total of 6884376 GU which translates to 0.105035017ETH in fees. However, to evaluate the actual cost in fiat currency, the fee value has to be related to a specific blockchain platform, as this aspect significantly influences the multiplicative factor. Considering Polygon blockchain, the above fee of 0.105035017 POL would result in 0.02 US$[5], representing an affordable cost considering the obtained benefits. Taking as a reference another blockchain-backed entity like a standard Ethereum Non-Fungible Token (NFT), it requires around 2 million GU for deployment and 50000 to 200000 GU for transfer or minting operations [5]. In this context, CONFETTY exhibits a higher deployment cost (performed only once for kickstarting the entire CONFETTY system) given by the more complex smart contracts logic, while its execution cost is in line with an NFT, resulting in a cost-efficient performance.

Let us now compare the above results with two other processes from the literature, namely retail [6] and incident management [8]. The X-ray, incident and retail processes have similar structures in the number of messages (10, 13 and 12, respectively), gateways (5, 3 and 2) and involved participants (4, 5 and 3) and exhibit similar trends from both runtime and cost standpoints. The incident management process is the slowest and most expensive one with an 8 % and 10 % increase in cumulative time and cost compared to the X-ray process, respectively. The experiments show that CONFETTY performs comparably to literature tools and scenarios. Indeed, ChorChain [8] required approximately 5 million GU for the incident management process, while MultiChain [6] required around 6 million GU for the retail process. Notice that such approaches only focus on public execution and enforcement without incorporating any confidential mechanisms and that they deploy a contract for each new instance.

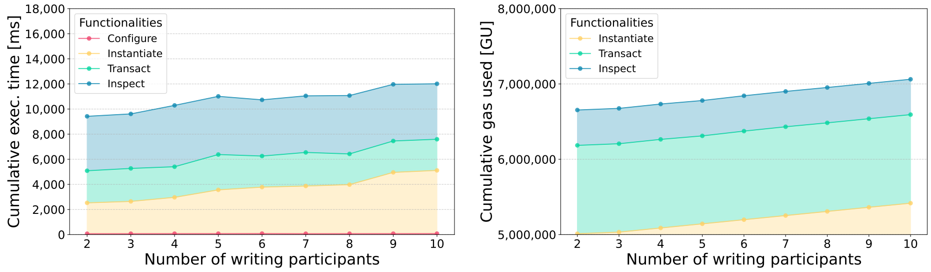
Scalability Assessment. Lastly, we analyzed the scalability of CONFETTY. To this end, we considered the dependence of time and cost taking as independent variables the dimensions of the process affecting the input size for our tool. In this section, we focus on the number of process participants and the choreography model size due to space restrictions as they evidence the most salient results. Further experiments including other variables like the number of gateways (parallel and exclusive) and the message payload size can be found in the online repository. The default values for the variables adhere to the example provided in Fig. 1. For every test, we let one variable change the value and keep the others fixed, assigned with the respective defaults. Figures 8 and 9 dissect the registered timings and costs to serve the different functionalities concurring to the total by stacking the respective subtotals.

Figure 8 depicts the performance trend varying the number of process actors actively sending messages from two (the minimum for a multi-party process), to ten (the maximum admissible within the X-ray choreography, with one different sender per message and activity). Figure 8(a) depicts the trend for cumulative time performance, showing a total variation of 3.9 s, mostly related to the *instantiate* functionality as it includes the authentication of all the participants. The costs shown in Fig. 8(b) follow a comparable linear trend.

Figure 9

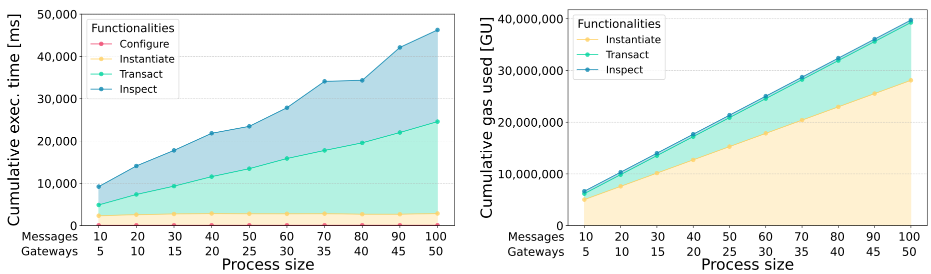
[5]The conversion rate is of 0.21 US$ at the time of writing (March 14, 2025)

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14



(a) Time analysis (b) Cost analysis

Fig. 8: Performance with increasing process participants



(a) Time analysis (b) Cost analysis

Fig. 9: Performance with increasing choreography size

displays the results observed varying the process model size. We progressively increased it by sequentially concatenating the whole choreography diagram multiple times to derive a new one, thus scaling the size up to ten times the original one. Figure 9(a) shows a linear growth in the runtime of the *transact* and *inspect* functionalities, without a significant impact on the remaining one, *instantiate*. The execution time increases with the number of messages due to additional read and write operations. Figure 9(b) shows the cost increase, with a linear growth registered for the *instantiate* and *transact* functionalities. More elements increase on-chain storage needs and hence gas cost. Read operations remain constant as they only handle one-time key management on-chain (hence the flat cost associated with *inspect*).

6 Conclusions

In this work, we presented CONFETTY, a blockchain-based architecture for PAISs. CONFETTY enables transparency and public enforcement for process execution while preserving the confidentiality of sensitive exchanged data. A public state saved on-chain maintains process data for enforcement and auditability purposes. The evolution of the public state is mediated by smart contracts implementing the business process control- and data-flow logic. We use MA-ABE encryption to hide sensitive information and grant access only to authorized parties based on their roles, encoded as attributes in access policies and embedded directly in their decryption keys. We confirmed its security guarantees with a threat model analysis, and demonstrated the time and cost performance of our implemented prototype with processes taken from the literature.

In future work, we envision a field study to assess the practical adoption and usability with real-world stakeholders. Another interesting avenue is the full decentralization of the architecture (to remove the existing points of trust residing in off-chain components) and governance (to cater for multiple process owners during the instantiation phase). Conducting a security analysis of our approach against stronger adversarial models is an intriguing future endeavor, too. Finally, we envision a study and design of a solution to enable automated decision support via computation held on encrypted data.

Acknowledgments. This work was partly funded by projects PINPOINT (B87G22000450001) under the PRIN MUR program, and Health-e-Data, funded by the EU-NGEU under the Cyber 4.0 NRRP MIMIT programme.

References

1. Benet, J.: IPFS - content addressed, versioned, P2P file system. CoRR **abs/1407.3561** (2014)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: SP. pp. 321–334 (2007)
3. Bramm, G., Gall, M., Schütte, J.: BDABE - blockchain-based distributed attribute based encryption. In: SECRYPT. pp. 99–110. INSTICC, SciTePress (2018)
4. Chase, M.: Multi-authority attribute based encryption. In: TCC. pp. 515–534. Springer (2007)
5. Choi, W., Woo, J., Hong, J.W.K.: Gas cost analysis of fractional nft on the ethereum blockchain. In: International Conference on Blockchain and Cryptocurrency. pp. 1–6. IEEE (2023)
6. Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Scala, E., Tiezzi, F.: Model-driven engineering for multi-party business processes on multiple blockchains. Blockchain: Res. Appl. **2**(3), 100018 (2021)
7. Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Tiezzi, F.: A flexible approach to multi-party business process execution on blockchain. Fut. Gen. Comp. Syst. **147**, 219–234 (2023)
8. Corradini, F., Marcelletti, A., Morichetta, A., et al.: Engineering trustable and auditable choreography-based systems using blockchain. ACM Trans. Manage. Inf. Syst. **13**(3) (2022)
9. Di Ciccio, C., Cecconi, A., Dumas, M., García-Bañuelos, L., López-Pintado, O., Lu, Q., Mendling, J., Ponomarev, A., Binh Tran, A., Weber, I.: Blockchain support for collaborative business processes. Informatik Spektrum **42**, 182–190 (2019)
10. Di Ciccio, C., Meroni, G., Plebani, P.: On the adoption of blockchain for business process monitoring. SoSyM **21**(3), 915–937 (2022)
11. Ghadge, A., Bourlakis, M., Kamble, S., Seuring, S.: Blockchain implementation in pharmaceutical supply chains: A review and conceptual framework. IJPR **61**(19), 6633–6651 (2023)
12. Hong, L., Zhang, K., Gong, J., Qian, H.: A practical and efficient blockchain-assisted attribute-based encryption scheme for access control and data sharing. Security and Communication Networks **2022**(1), 4978802 (2022)
13. Khan, S.N., Loukil, F., Guegan, C.G., Benkhelifa, E., Bani-Hani, A.: Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-Peer Netw. Appl. **14**(5), 2901–2925 (2021)
14. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: SP. pp. 839–858. IEEE (2016)

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-032-02929-4_14

15. Köpke, J., Meroni, G., Salnitri, M.: Designing secure business processes for blockchains with SecBPMN2BC. Future Gener. Comput. Syst. **141**, 382–398 (2023)
16. Lin, C., He, D., Zeadally, S., Huang, X., Liu, Z.: Blockchain-based data sharing system for sensing-as-a-service in smart cities. ACM Trans. Internet Technol. **21**(2) (2021)
17. Liu, Z., Jiang, Z.L., Wang, X., et al.: Multi-authority ciphertext policy attribute-based encryption scheme on ideal lattices. ISPA/IUCC/BDCloud/SocialCom/SustainCom pp. 1003–1008 (2018)
18. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., Ponomarev, A.: Caterpillar: A business process execution engine on the Ethereum blockchain. Softw., Pract. Exper. **49**(7), 1162–1193 (2019)
19. Marangone, E., Di Ciccio, C., Friolo, D., Nemmi, E.N., Venturi, D., Weber, I.: Enabling data confidentiality with public blockchains (2023)
20. Marangone, E., Di Ciccio, C., Friolo, D., Nemmi, E.N., Venturi, D., Weber, I.: MARTSIA: Enabling data confidentiality for blockchain-based process execution. In: EDOC. pp. 58–76 (2023)
21. Marangone, E., Di Ciccio, C., Weber, I.: Fine-grained data access control for collaborative process execution on blockchain. In: BPM Blockchain and RPA Forum. pp. 51–67. Springer (2022)
22. Mendling, J., et al.: Blockchains for business process management - challenges and opportunities. ACM Trans. Manag. Inf. Syst. **9**(1), 4:1–4:16 (2018)
23. Mühlberger, R., Bachhofner, S., Ferrer, E.C., et al.: Foundational oracle patterns: Connecting blockchain to the off-chain world. In: BPM: Blockchain and RPA Forum. pp. 35–51 (2020)
24. Pham, V.D., Tran, C.T., Nguyen, T., Nguyen, T.T., et al.: B-box - a decentralized storage system using IPFS, attributed-based encryption, and blockchain. In: RIVF. pp. 1–6 (2020)
25. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT. p. 457–473. Springer-Verlag (2005)
26. Scandariato, R., Wuyts, K., Joosen, W.: A descriptive study of microsoft's threat modeling technique. Requir. Eng. **20**(2), 163–180 (2015)
27. Stiehle, F., Weber, I.: Blockchain for business process enactment: A taxonomy and systematic literature review. In: BPM Blockchain and RPA Forum. LNBIP, vol. 459, pp. 5–20. Springer (2022)
28. Thomas, S.: Ssl and tls essentials: Securing the web (2000)
29. Tran, A.B., Lu, Q., Weber, I.: Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In: BPM Demos. pp. 56–60. Springer (2018)
30. Uddin, M.: Blockchain medledger: Hyperledger fabric enabled drug traceability system for counterfeit drugs in pharmaceutical industry. International Journal of Pharmaceutics **597**, 120235 (2021)
31. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger (2014)
32. Xu, X., Weber, I., Staples, M.: Blockchain Patterns, pp. 113–148. Springer Int. Publishing (2019)
33. Yan, L., Ge, L., Wang, Z., Zhang, G., Xu, J., Hu, Z.: Access control scheme based on blockchain and attribute-based searchable encryption in cloud environment. JCC **12**(1), 61 (2023)

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-032-02929-4_14

This document is a pre-print copy of the manuscript
(Marcelletti et al. 2025)
published by Springer (available at link.springer.com).

The final version of the paper is identified by DOI: 10.1007/978-3-032-02929-4_14

References

Marcelletti, Alessandro, Edoardo Marangone, Michele Kryston, and Claudio Di Ciccio (2025). "Balancing Confidentiality and Transparency for Blockchain-Based Process-Aware Information Systems". In: *BPM Forum*. Ed. by Arik Senderovich, Cristina Cabanillas, Irene Vanderfeesten, and Hajo A. Reijers. Vol. 564. Lecture Notes in Business Information Processing. Springer, pp. 238–255. ISBN: 978-3-032-02929-4. DOI: 10.1007/978-3-032-02929-4_14.

BibTeX

```
@InProceedings{   Marcelletti.etal/BPMF2025:CONFETTY,
  author        = {Marcelletti, Alessandro and Marangone, Edoardo and
                  Kryston, Michele and Di Ciccio, Claudio},
  booktitle     = {BPM Forum},
  title         = {Balancing Confidentiality and Transparency for
                  Blockchain-Based Process-Aware Information Systems},
  year          = {2025},
  pages         = {238--255},
  crossref      = {BPM2025Forum},
  doi           = {10.1007/978-3-032-02929-4_14},
  keywords      = {Business process management; Distributed ledger
                  technologies; Blockchain; Attribute-based encryption;
                  Security; Privacy}
}
@Proceedings{     BPM2025Forum,
  title         = {Business Process Management Forum - {BPM} 2025 Forum,
                  Seville, Spain, August 31 - September 5, 2025,
                  Proceedings},
  year          = {2025},
  editor        = {Arik Senderovich and Cristina Cabanillas and Irene
                  Vanderfeesten and Hajo A. Reijers},
  isbn          = {978-3-032-02929-4},
  publisher     = {Springer},
  series        = {Lecture Notes in Business Information Processing},
  volume        = {564}
}
```