# Fine-grained Data Access Control for Collaborative Process Execution on Blockchain

Edoardo Marangone[1][0000−0002−0565−9168], Claudio Di Ciccio[1][0000−0001−5570−0475], and Ingo Weber[2][0000−0002−4833−5921]

[1] Sapienza University of Rome, Rome, Italy
{edoardo.marangone,claudio.diciccio}@uniroma1.it
[2] Software and Business Engineering, Technische Universitaet Berlin, Germany
ingo.weber@tu-berlin.de

**Abstract.** Multi-party business processes are based on the cooperation of different actors in a distributed setting. Blockchains can provide support for the automation of such processes, even in conditions of partial trust among the participants. On-chain data are stored in all replicas of the ledger and therefore accessible to all nodes that are in the network. Although this fosters traceability, integrity, and persistence, it undermines the adoption of public blockchains for process automation since it conflicts with typical confidentiality requirements in enterprise settings. In this paper, we propose a novel approach and software architecture that allow for fine-grained access control over process data on the level of parts of messages. In our approach, encrypted data are stored in a distributed space linked to the blockchain system backing the process execution; data owners specify access policies to control which users can read which parts of the information. To achieve the desired properties, we utilise Attribute-Based Encryption for the storage of data, and smart contracts for access control, integrity, and linking to process data. We implemented the approach in a proof-of-concept and conduct a case study in supply-chain management. From the experiments, we find our architecture to be robust while still keeping execution costs reasonably low.

**Keywords:** Attribute Based Encryption · Blockchain · Business Process Management · IPFS

## 1 Introduction

Blockchain technology is gaining momentum, among other reasons because it allows for the creation and enactment of business processes between multiple parties with low mutual trust [29,26]. The distributed nature of public permissionless blockchains allows every user in the network to have a copy of the ledger and therefore all the data is freely accessible. This transparency, together with the permanence of data and non-repudiability of transactions granted by the technology, motivate the use of blockchains as a reliable ground for verifiable and trustworthy interactions.

Especially in cases wherein the parties lack trust in one another, though, hiding some data from the majority of users can be useful. As a matter of fact, security and privacy are at the centre of the debate when considering blockchain technology [33,9]. For example,

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-031-16168-1_4

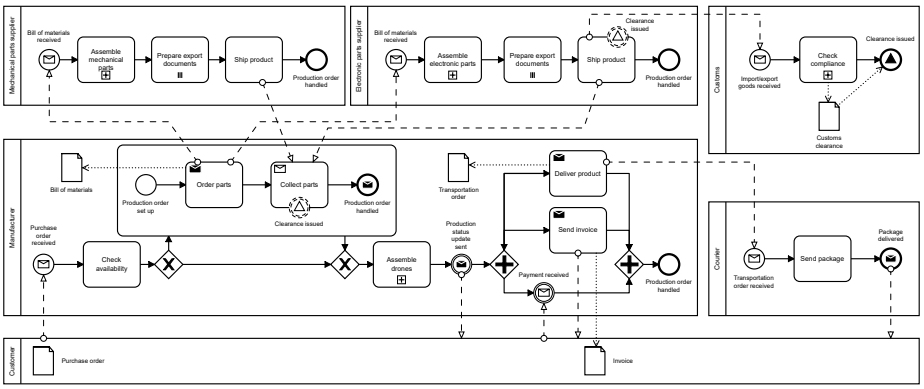


Fig. 1: BPMN collaboration diagram of a multi-party process

Corradini et al. [5] point to security and privacy aspects as relevant points. The authors underline that the encryption of the payload of messages (a solution already present in the literature) does not preserve the secrecy of information. Sharing a decryption key among process participants does not allow data owners to selectively control the access to different parts of a single message. Using the public key of a recipient forces the sender to create multiple copies of every message (one per intended reader) and severely hampers the traceability of the process. Another proposed solution is the usage of permissioned blockchains. However, this scheme entails strong complexity and management issues.

Our work aims to close the gap by proposing a technique that guarantees data privacy among parties. With this architecture, the parties can exchange information in a secure way and can also hide data (or parts thereof) from other players with whom they do not want to share it. As such, this paper introduces a novel approach to address security and privacy problems by presenting an architecture that allows for the ciphering of selected data using Attribute-Based Encryption (ABE) [25] so as to control fine-grained read and write access to data.

In the following, Sect. 2 presents a running example, to which we will refer throughout the paper, and illustrates the problem we tackle. Sect. 3 outlines the fundamental notions that our solution is based upon. In Sect. 4, we describe our approach in detail. In Sect. 5, we present our proof-of-concept implementation and illustrate the results of the experiments we conducted therewith. Sect. 6 presents the related work in the literature. Finally, Sect. 7 concludes the paper and draws some avenues for future works.

## 2 Running example and problem illustration

Figure 1 depicts a Business Process Model and Notation (BPMN) collaboration diagram representing the supply chain behind the production of drones. We will use this scenario as a running example throughout our paper. We assume process execution is backed by a blockchain-based infrastructure as illustrated in [8].

A new process instance begins when a *Customer* orders one or multiple drones from a *Manufacturer*. After checking the availability of the mechanical and electronic

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-031-16168-1_4

components in the warehouse, the *Manufacturer* orders the missing ones from a local *Mechanical parts supplier* and an international *Electronic parts supplier*, respectively. After the assemblage of the required parts, the suppliers prepare the shipment documents, the package, and send the products. *Customs* then check the documents of the international supplier and release the custom clearance after the verification of compliance concludes positively. Upon the receipt of the parts, the *Manufacturer* proceeds with their assemblage. After sending a notification about the stage reached by the production process, the *Manufacturer* sends an invoice to the paying *Customer*, and requests a *Courier* to deliver the package. The process concludes with the consignment of the ordered product.

We highlight the information artefacts we are going to primarily focus on for our examples as paper documents with twisted corners, namely *(i)* purchase order, *(ii)* bill of materials (BoM), *(iii)* customs clearance, *(iv)* invoice (for the customer), and *(v)* transportation order. First and foremost, we observe that the exchanged information in this process should not be fully accessible outside of the involved counterparts in the process execution. Notice that, instead, a non-encrypted communication through the blockchain allows every node (not necessarily involved in the process either) to disclose the full content of all data attached to transactions. If all parties knew a secret key, they could store the data on-chain once encrypted with that key to ensure nobody outside their circle can read through them, yet ensuring that the data are notarised by the blockchain. However, we remark that although the information exchanges involve multiple actors in collaborative processes, it is rare that *every* actor is supposed to read *all* the exchanged data in their entirety – particularly in this scenario, it never is the case. For example, the invoice details should be undisclosed to any other party that is not the *Customer* or the *Manufacturer*, just like the purchase order. Likewise, the transportation order should be fully accessible only to the *Manufacturer* and the *Courier*.

Whenever a message sender and recipient are single players who know one another in advance, the data producer could encrypt the message and give the access key to the sole expected consumer. However, this may not be a reasonable assumption in cases like the one we discuss here. The customs clearance, for instance, should be known to more than two parties, as the *Electronic parts supplier* and the *Customs* are directly involved but the *Manufacturer* should also be made aware of the result at the end of the border controls. Besides, not only the operators in the *Customs* office involved in the first inspection should be granted access – this restriction would impede future checks.

Another example of non-binary communication channel pertains to the bill of materials. The section of the BoM for the *Mechanical parts supplier* should not be read by any other party but the recipient of the production order and the *Manufacturer*. Notice that, albeit the *Electronic parts supplier* is also a producer of basic components for the *Manufacturer*, it should access the sole part of the BoM referred to its area of competence. Therefore, different parts of a shared data artefact should be accessible to different players. In contrast, the section of the BoM with the identifying data of the *Manufacturer* should be visible to both suppliers.

In the last few years, research work flourished for blockchain-based control-flow automation and decision support for processes like the one in this section [29,27,16,17]. Our investigation complements this body of research by focussing on the secure infor-

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

Table 1: Requirements and corresponding actions in the approach

| | Requirement | Approach |
|---|---|---|
| **R1** | Access to parts of messages should be controllable in a fine-grained way (attribute level), while integrity is ensured | We use Attribute-Based Encryption (ABE) to encrypt messages, which are stored off-chain while their locator and hash are kept on chain. Access is mediated by a component that decrypts messages only if the requester has the necessary attributes. |
| **R1.1** | Access policies should be linked to individual (parts of) information artefacts | Access policies associate granted classes of users to the sole messages or sections (slices) thereof that pertain to them |
| **R1.2** | Access policies should control access levels for authenticated users | The policies are fine-grained, and the component that decrypts messages does so as per on-chain information |
| **R1.3** | Non-authorised access is prevented | Data is kept in an encrypted form, and only authorised requests allow for decryption; salting prevents leakage of information through hashes |
| **R2** | Information artefacts should be written in a permanent, tamper-proof and non-repudiable way | We use hashed, permanent off-chain storage in combination with hashes on-chain |
| **R3** | The system should be independently auditable with low overhead | On-chain information is publicly available to users of the system, and through hashes integrity of off-chain data becomes auditable |

mation exchange among multiple parties in a collaborative though partially untrusted scenario. We list the key requirements for our approach in Table 1. Next, we focus on the background knowledge that to which our approach resorts.

## 3 Background

In this section, we give an overview of the fundamental notions upon which our approach is built. The fundamental building blocks of our work are Distributed Ledger Technology (DLT), particularly programmable blockchain platforms, and Attribute-Based Encryption (ABE). Next, we outline the basic notions they build upon and relate them to our running example.

Distributed Ledger Technologies (DLTs) realise protocols for the storage, processing and validation of transactions among a network of peers. Their distributed nature entails that no central authority or intermediary are involved in the management of the data. To all these transactions a timestamp and a unique cryptographic signature are attached. To produce signatures, a public/private key scheme is adopted. Every user holds an account with a unique address to which the public and private keys are associated. The shared database is public so all participants in the network can have access to the data. **Blockchain** is a type of DLT, wherein segments of the ledger are collated into blocks and those blocks are backward-linked together forming a chain. DLTs in general and the blockchain in particular cannot be tampered with thanks to a blend of cryptographic techniques, including the hashing of blocks themselves, the inclusion in every block of the hash of the previous one, and the distributed validation of transactions. Public blockchain platforms such as Bitcoin [20], Ethereum [31] and Algorand [4] require fees to be paid in order to let transactions be submitted and processed by the platform. More recent blockchain protocols such as Ethereum and Algorand include the opportunity to run **Smart Contracts**, namely programs deployed, stored and executed in the blockchain [7]. Smart contracts are invoked via transactions. The execution is spread

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

among the nodes without the involvement of a trusted third party so that the overall behaviour can be verified and trusted. Moreover, smart contracts can also trigger the next steps of a workflow when some conditions are met [29]. As with transactions, the execution of smart contract code is subject to costs that in the Ethereum nomenclature fall under the name of *gas*. These costs depend on the complexity of the invoked code and on the amount of data exchanged and stored. To reduce the invocation costs of smart contracts, external Peer-to-peer (P2P) systems are typically employed to save larger bulks of data [32]. One such system is the **InterPlanetary File System (IPFS)**, a distributed system for the storage and access to files. Having it a Distributed Hash Table (DHT) at its core, the stored files are scattered among several nodes. Akin to DLTs, no central authority or trusted organisation retaining the whole bulk of data is thus involved. IPFS makes use of content-addressing to uniquely identify each file in the network. The data saved on IPFS are hash-linked by resource locators that are then sent to contracts that store them on chain [15]. Thereby, the hash of external data together with their remote handle are permanently stored on chain to link them to the ledger.

In a multi-party collaboration scenario like the one we described in Sect. 2, the blockchain creates a layer of trust: the ledger operates as an auditable notarisation infrastructure to certify the occurrence of transactions among the involved actors (e.g., the purchase orders or custom clearances), the smart contracts guarantee that the workflow is followed as per the agreed behaviour, as illustrated in [8,18]. Documents such as purchase orders, bill of materials and custom clearances can be stored on IPFS and linked to the transactions that report on their submission. However, those data are accessible to all peers on chain. Techniques to cipher the data and control their accessibility to predefined users become necessary so as to take advantage of the security and traceability guarantees of blockchain while managing read and write grants on the stored information.

**Attribute-Based Encryption (ABE)** is a type of public-key encryption in which the *ciphertext* (i.e., an encrypted text derived from a *plaintext*) and the corresponding private key to decipher it are linked together by attributes [25]. In particular, the Ciphertext-Policy ABE (CP [3]) is such that every potential user is associated with a number of *attributes* over which *policies* are expressed. Attributes, in particular, are propositional literals that are affirmed in case a user enjoys a given property. In the following, we shall use the teletype font to format attributes and policies. For example, user `0xE756[...]b927` is associated with the attributes `Supplier`, to denote their role, and `14548487`, to specify their involvement in process instance number `14548487`. For the sake of brevity, we omit from the attribute name that the former is a role and the latter a process instance identifier (e.g., `Supplier` in place of `RoleIsSupplier` or `14548487` instead of `InvolvedIn14548487`) as we assume it is understandable from the context. Policies are associated to messages and expressed as propositional formulae on the attributes (the literals) to determine whether a user is granted access (e.g., `Courier or Manufacturer`).

All users can attain a unique *secret key* (*sk*). The *sk* is a fixed-length numeric sequence (typically of 512 bits) generated on the basis of the user attributes and a pair of keys, namely a *master public key* (*mpk*) and a *master key* (*mk*). In turn, *mpk* and *mk* are generated through a cryptographic parametric algebraic structure (e.g., a pairing group). A message is encrypted via the *mpk* and the policy. The users can decrypt the ciphertext

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

by using the *mpk* and their own *sk*. It follows that an unauthorised user would not have the suitable *sk* as per the policy. Furthermore, without knowledge of the *mpk*, the user cannot read the encrypted data either. Notice that *mpk* alone would not allow for the generation of new *sk*s as the master key (*mk*) is also necessary. To conclude, we remark that the generation of keys, the encryption of plaintexts, and the deciphering thereof, are operations that are algorithmically handled and thus no trusted party is needed – any peer with access to the required credentials could run the necessary code.

In our setting, intuitively, users are process participants, messages are the data artefacts exchanged during the process execution, ciphertexts are the encrypted data artefacts, policies determine which artefacts can be access by whom, and keys are the instruments that are granted to the process parties to try and access the artefacts. Next, we explain how we combine the use of blockchain and CP-ABE to build an access-control architecture for data exchanges in blockchains that meet the requirements listed in Table 1.

## 4 The CAKE approach

In this section, we describe our approach, named Control Access via Key Encryption (CAKE). Figure 2 illustrates the main components of our architecture alongside their interaction by means of a UML collaboration diagram. The involved parties are:

1. the Data Owner, who wants to cipher the information artefacts (henceforth also collectively referred to as *plaintext*) with a specific access policy (e.g., the *Manufacturer* who wants to restrict access to the bill of materials to the sole intended parties, i.e., the suppliers); we assume Data Owner is equipped with a public/private key pair;
2. one or more Readers, interested in some of the information artefacts (e.g., the *Manufacturer*, the *Electronic parts supplier*, the *Mechanical parts supplier*); we assume every Reader to keep their own public/private key pair;
3. the Attribute Certifier, specifying the attributes characterising the potential readers of the information artefacts; we assume the Attribute Certifier to hold a blockchain account;
4. the Secure Data Manager (SDM), a stateless software component ciphering the plaintext with the policy received from Data Owner; we assume the Data Owner to hold a blockchain account;
5. the Secure Key Manager (SKM), a stateless software component generating access keys for Readers and that the Readers invoke to decrypt messages; we assume the SKM comes endowed with a pair of public and private encryption keys and to hold a blockchain account;
6. IPFS, used to store the ciphertext (i.e., the ciphered plaintext); and finally
7. the Smart Contract, used to safely store the resource locator to the ciphertext saved on IPFS and the information about potential readers of the information artefacts; at deployment time, the Smart Contract is associated with the blockchain account addresses of the SDM, of the SKM, and of the Attribute Certifier, so as to accept invocations only by those components.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

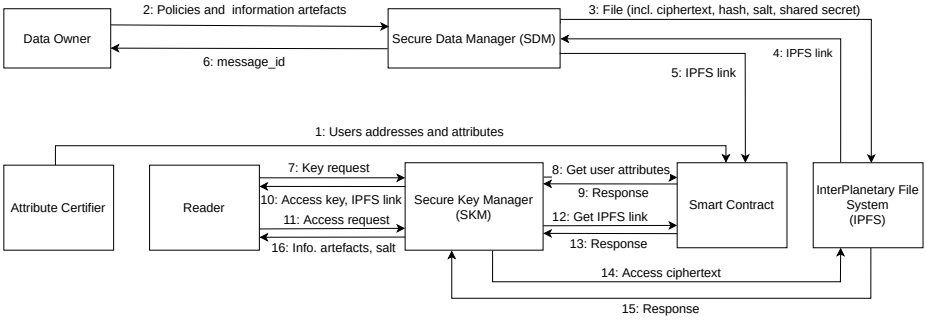


Fig. 2: The key component interactions in the CAKE approach

Using the enumeration schema of Fig. 2, action (1) is a preliminary operation in which the Attribute Certifier transmits the attributes and the identifying blockchain account addresses of the Readers to the Smart Contract so as to make them publicly verifiable on chain. To this end, the Attribute Certifier operates as a push-inbound oracle [19]. The Attribute Certifier stores on chain the attributes that determine the role of the Reader and, optionally, the list of process instances in which they are involved. For example, the Attribute Certifier stores on chain that `0x906D[...]Dba8` is the address of a user that holds the `Manufacturer` attribute, determining the role, alongside the numeric identifier `14548487` for the running process (the so-called *case id*), specifying the participation of the manufacturer in that particular process instance. Also, it registers that `0xE756[...]b927` and `0xE2C8[...]A2810` are Readers endowed with the `Supplier` and `14548487` attributes, and that the `Electronics` and `Mechanics` attributes belong to the first and the second Reader, respectively.

Thereafter (2), the Data Owner sends the plaintext (i.e., an information artefact such as the bill of material) and the access policies to the SDM, so that the latter can make use of the ABE algorithm to cipher the plaintext with the policy. The access policy declares the conditions according to which a user can be granted access to the ciphered information.

Notice that a message can be separated into multiple *slices*, and each of those can be associated to a different policy. For example, the bill of materials of process instance `14548487` is partitioned as follows (see Table 2): a slice is accessible to all suppliers and manufacturers involved in the process instance, as the policy reads `14548487 and (Manufacturer or Supplier)`; another one pertains to the sole production order of mechanical parts – i.e., `14548487 and (Manufacturer or (Supplier and Mechanics))`; a third slice is specific for the electronic parts supplier – i.e., `14548487 and (Manufacturer or (Supplier and Electronics))`. Notice that actors who are granted access to the data do not necessarily need to be directly involved in a process instance. It is the case of *Customs*, e.g., in our running example: the policy reads, indeed, `Customs or (14548487 and ([...]))`. Therefore, *Customs* are authorised to access data across all instances with their key, unlike *Manufacturer*s. In other words, the inclusion of a `case_id` as an attribute in the policy determines a design choice on

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

Table 2: Message policy examples

| Message | Slice | Policy |
|---|---|---|
| Purchase order | 1 | `14548487 and (Customer or Manufacturer)` |
| Bill of materials | 1 | `14548487 and (Manufacturer or (Supplier))` |
| | 2 | `14548487 and (Manufacturer or (Supplier and Electronics))` |
| | 3 | `14548487 and (Manufacturer or (Supplier and Mechanics))` |
| Customs clearance | 1 | `Customs or (14548487 and (Manufacturer or (Supplier and Electronics)))` |
| Invoice | 1 | `14548487 and (Manufacturer or Client)` |
| Transportation order | 1 | `14548487 and (Manufacturer or Courier)` |

whether a Reader can use the *sk* across different process instances or not. If the `case_id` is specified, different access keys are generated for separate instances.

We assume every slice to be associated with a unique identifier (henceforth, `slice_id`). Table 2 lists the policies used in our running example. The semantics of access policies meets **R1.1**, as they are at the fine-grain level of slices within messages.

Then (3), the SDM runs the algorithm for the generation of the ABE master public key (*mpk*) and master key (*mk*). It uses the master key (*mk*) and the policies to encrypt the plaintext and attain the ciphertext. Thereafter, it generates a unique identifier for the message (`message_id`), such as `17071949511205323542`. For every slice, it builds a unique identifier (`slice_id`) and a random number (named *salt*) to be additively used for hashing. Finally, it stores on IPFS the `message_id` and, for each slice, the `slice_id`, ciphertexts, hash of the slice's plaintext combined with the corresponding salt, and the following data encrypted with the public key of the SKM, which we collectively refer to as *shared secret*: *(i)* the *mpk*, *(ii)* the *mk*, *(iii)* additional parametric metadata for the cryptographic algebraic structure (for every slice). In our approach, the SDM forgets both *mpk* and *mk* after storing them as it is stateless. Also, notice that a new pair of keys is created for every message (i.e., IPFS file) to address **R1.2**. As a result (4), the IPFS returns the resource locator (i.e., the link to the IPFS file) to the SDM, which the SDM stores in the Smart Contract (5). Next (6), the SDM returns the `message_id` to the Data Owner. The Data Owner can send the `message_id` to the interested parties to let them know the content is ready for retrieval. For example, the *Manufacturer* sends the suppliers the information that `17071949511205323542` is the identifier to use to fetch the bill of material.

As said, the SDM stores the association between the message and the resource locator on chain via the Smart Contract (5). Thus, we have data stored off-chain that is linked with the blockchain ledger, as per **R2**. Table 3 illustrates the messages we described in our running example in Sect. 2 as saved on IPFS by the SDM. Every IPFS file in our approach consists of a header with the address of the sender (i.e., the Data Owner), the `message_id`, and the encrypted pair of keys (*mpk* and *mk*). The body consists of slices, each with its identifier (`slice_id`), hash, ciphertext, salt and metadata. We recall that salt and metadata are encrypted with the public key of the SKM. Furthermore, notice that the plaintext is encrypted, and albeit being stored semi-publicly on IPFS, it is unreadable even to the Data Owner (unless a party obtains a suitable key, which can be granted only by the SKM). Thereby, we meet **R1.3**.

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

Table 3: Examples of messages encoded by the CAKE system.

| Message | Original data | File header | File body (slices) |
|---|---|---|---|
| Purchase order | `Company_name: Alpha`<br>`Address: 34, Alpha street`<br>`E-mail: cpny.alpha@mail.com`<br>`Quantity: 5`<br>`Price: $5000` | `sender: 0x989a [...] FeaD,`<br>`message_id: 22063028103945568 65,`<br>`pk: {"g": "\\u00af [...] 00f4},`<br>`mk: {"beta": "\\u009b [...] 009a}` | `slice_id: 10322929677141064041,`<br>`hash: 0x2958 [...] fb611,`<br>`salt: "\\u008d [...] 01bd",`<br>`metadata: {"c1": [...] 4Els},`<br>`cipherText: "qp2I [...] 7Ue9Q"` |
| Bill of materials | `Manufacturer_company: Beta`<br>`Address: 82, Beta street`<br>`E-mail: mnfctr.beta@mail.com` | `sender: 0x906D [...] Dba8,`<br>`message_id: 17071949511205323542,`<br>`pk: {"g": "\\u0087 [...] 00ca},`<br>`mk: {"beta": "\\u00b2 [...] 00fb}` | `slice_id: 781610580582830690 1,`<br>`hash: 0x953a [...] f8d8,`<br>`salt: "Zu00 [...] u004",`<br>`metadata: {"c1": [...] 00a0},`<br>`cipherText: "oT2W [...] MQ=="` |
| | `Frames_quantity: 8`<br>`Propeller_quantity: 80`<br>`PropellerGuard_quantity: 63`<br>`Camera_quantity: 30`<br>`Controller_quantity: 4`<br>`Amount_paid: $12000` | | `slice_id: 684789586295986359 2,`<br>`hash: 0x12es [...] ig23,`<br>`salt: "bw32 [...] b464",`<br>`metadata: {"c1": [...] asq2},`<br>`cipherText: "AS2w [...] btwd"` |
| | `IMU_quantity: 6`<br>`ESC_quantity: 40`<br>`Engines_quantity: 9`<br>`Batteries_quantity: 25`<br>`Amount_paid: $9850` | | `slice_id: 314789976496645986 6,`<br>`hash: 0xj4rs [...] ne3d,`<br>`salt: "nsiw [...] mej4",`<br>`metadata: {"c1": [...] 23rs},`<br>`cipherText: "ht3r [...] asf3"` |
| Customs clearance | `Date: 2022-05-10`<br>`Sender: Beta`<br>`Receiver: Alpha` | `sender: 0x0182 [...] 8DC0,`<br>`message_id: 57575788878230570 98,`<br>`pk: {"g": "\\uKu00 [...] 00b9},`<br>`mk: {"beta": "\\u004d [...] 0d2r}` | `slice_id: 460701105413554429 0,`<br>`hash: 0x4ee6 [...] 2386,`<br>`salt: \u0010 [...] 0013 ,`<br>`metadata: {"c1": [...] 00c2},`<br>`cipherText: "udBA [...] IA=="` |
| Invoice | `Gross_total: $5000`<br>`Company_VAT: U12345678`<br>`Issue_date: 2022-05-12` | `sender: 0x906D [...] Dba8,`<br>`message_id: 67960037019529364 28,`<br>`pk: {"g": "\\u00dc [...] 00a2},`<br>`mk: {"beta": "\\u00be [...] 00c0}` | `slice_id: 126417826144933959 49,`<br>`hash: 0xad46 [...] 0f79,`<br>`salt: "o9\u [...] 01e5,`<br>`metadata: {"c1": [...] u09a},`<br>`cipherText: "7QsM [...] KVRS"` |
| Transportation order | `E-mail: cpny.alpha@mail.com`<br>`Arrival_date: 2022-06-06` | `sender: 0x906D [...] Dba8,`<br>`message_id: 98466976843684 36866,`<br>`pk: {"g": "\\ur25d [...] 3a7s},`<br>`mk: {"beta": "\\u001q [...] 08q2}` | `slice_id: 865535701700786046 6,`<br>`hash: 0xeide [...] 3f9f,`<br>`salt: "\bvNA [...] 011n",`<br>`metadata: {"c1": [...] 00b4},`<br>`cipherText: "opBX [...] J709"` |

When the Reader (e.g., the *Electronic parts supplier*) wants to read the data of a message (e.g., the section of interest in the bill of materials), it requests a key from the SKM (7). Then, the SKM retrieves the Reader data (the blockchain address and attributes) from the Smart Contract (8,9). Notice that these pieces of information were previously stored by the Attribute Certifier at step (1). Equipped with these pieces of information and with the shared secret (including the `pk` and `mk`), it produces an ABE secret key (*sk*) for the Reader and sends it back (10) together with the IPFS link corresponding to the requested message (e.g., the one identified by `17071949511205323542`). Notice that the shared secret (including the *mpk* and *mk*) is saved on IPFS encrypted with the public key of the SKM, so that only the SKM can use its private key to retrieve the necessary information and produce the *sk*. Also, we remark that the SKM is stateless, so it retains no information after it responds to the Reader.

Equipped with their own access key (*sk*), the Reader can begin the message decryption procedure. As per the ABE paradigm, the *sk* alone is not sufficient to decipher messages though. The *mpk* is also necessary, though it is encrypted in the IPFS file with the public key of the SKM. Therefore, the Reader makes an access request to the SKM (11). In turn, the SKM asks for the IPFS link from the Smart Contract (12,13). Then, the SKM retrieves the ciphertext from IPFS (14) and decrypts it with the *sk* of the user and the shared secret, extracted and deciphered from the requested message. If the decryption is successful, the SKM component sends the information artefacts back (15). Otherwise the Reader request is denied.

Recall that a message can be composed of multiple slices. In the case of the bill of material, e.g., message `17071949511205323542` consists of three slices (see Tables 2 and 3). The first slice contains information available to all suppliers, the second one only for *Electronic parts supplier*, and the third one only for *Mechanic parts supplier*. Therefore, with the *sk* of the *Electronic parts supplier*, its attributes and the shared secret kept by the SKM, the latter can decipher only the first and second slice, but not

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

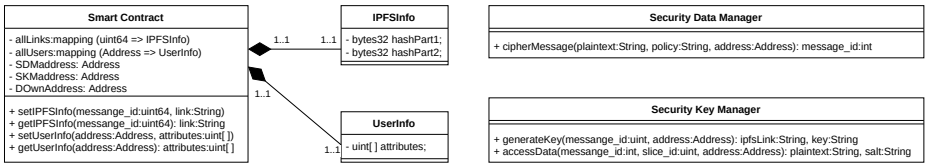


Fig. 3: The implemented components of the CAKE system

the third one – as per the specified policies. The SKM component thus returns those slices only (16). The controlled, fine-grained data access in CAKE is designed to meet the requirements regarding auditability (**R3**), integrity and control (**R1**) and specifically granularity (**R1.1**).

We conclude this section with a few more remarks about security and integrity. When a Reader has received the information artefacts, they may want to verify that the data is not counterfeit. This is the reason why the SKM component returns the (decrypted) `salt` along with the information artefacts to the Reader (16). With the received deciphered data and the salt, the Reader can compute the hash and check if it is equal to the one stored on IPFS by the SDM at step (3) or not. We remark that the Reader had received the IPFS link along with the key at step (8), so that they could directly access the data on IPFS to check the integrity of the information artefacts received from the SKM later on. This design contributes to meeting **R1.3**. The data on IPFS is ciphered and only the SKM can decipher it. The usage of the salt prevents leakage of information, like dictionary attacks.

Also, we remark that the communication backbone outside of blockchain and IPFS for the information exchanges between components is based on the Secure Sockets Layer (SSL) protocol, so as to avoid packet sniffing from malicious third parties that could intercept the data. Furthermore, we assume that the communication from Data Owner to SDM, and from Reader to SKM, are preceded by an initial authentication phase to address **R1.2**. During a preliminary handshake, the SDM and the SKM send a random value to the callers. The callers responds with that value signed with their own private key, so as to let the invoked components verify their identity. Notice that, without this measure, any malicious peer could request the *sk* in place of the real Reader by knowing their address and guessing a file they could be granted access to.

## 5 Implementation and evaluation

This section describes the proof-of-concept implementation of our approach and the test runs we conducted to assess its affordability for data access control and audits.

Figure 3 depicts the core CAKE components in the form of a UML class diagram. The code of our prototype can be found at `https://github.com/apwbs/CAKE` together with the detailed results of our experiments. We implemented the SDM, SKM and the communication channels in Python. We encoded the Smart Contract in Solidity as we employ the Ethereum testnet Ropsten for the deployment of our blockchain components: all transactions directed to the CAKE Smart Contract instance

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

Table 4: Gas consumption and total cost of test transactions

| | `setIPFSInfo` | | | `setUserInfo` | | | |
|---|---|---|---|---|---|---|---|
| | **gasUsed [unit]** | **gasPrice [wei]** | **Total cost [EUR]** | **gasUsed [unit]** | **gasPrice [wei]** | **Total cost [EUR]** | **ETH/EUR exchange** |
| avg. | 67 486.52 | 1 399 400 015 | 0.164 38 | 40 755 | 1 370 810 611 | 0.097 34 | 1746.35 |
| min | 67 484.6 | 1 000 000 007 | 0.123 78 | 40 755 | 1 000 000 007 | 0.074 75 | 1650.68 |
| max | 67 487 | 1 649 000 034 | 0.185 87 | 40 755 | 1 644 053 019 | 0.111 91 | 1834.14 |

we used for our tests can be freely inspected at `https://ropsten.etherscan.io/address/0x2D9EAe20E1E7515d47fBB9A5d454Ce7Be59cA03f`. To manage the public/private key pair system for the Data Owner, Readers and SKM, we resort to the Rivest–Shamir–Adleman (RSA) algorithm [24]. In our software prototype, the length of the pair of keys amounts to 2048 bits.

To test our system, we called the methods of the deployed Smart Contract to measure gas consumption. More in the detail, we focussed on the invocations that require the payment of gas fees, namely *(i)* the storage of the address and attributes of Readers (`setUserInfo(...)` in Fig. 3), and *(ii)* the storage of the IPFS link associated to a message (`setIPFSInfo(...)`). The data we used to run our experiments are taken from our running example (see Table 3). We executed fifteen calls per day in five consecutive days. Out of the fifteen calls, ten were directed to `setUserInfo` and five to `setIPFSInfo`. The higher numerosity of the former is due to the fact that the latter has a rather fixed input format as the length in bits of IPFS locators is constant. As `setUserInfo` takes as input arrays, the variability of inputs is potentially higher.

Table 4 summarises the results. For every call we provide the average, minimum and maximum of *(i)* units of gas used to run the code, *(ii)* price in wei paid for the gas consumption (using the Ropsten default setting), *(iii)* total cost in Euros based on the daily exchange rate with an Ether. The costs are relatively limited and range between ten and twenty Euro cents, which can be considered a reasonably low amount in light of the permanency and security guarantees provided by the system. Most importantly, the size of the information artefacts do not have a significant impact on the price paid to store them.

To save on gas expenditures, we have adopted a few mechanisms that reduced the size of input and output data. Among them, we recall the following two. First, we have turned IPFS links from their native base-58 encoded format in strings of 46 characters to pairs of `bytes32` elements (the `IPFSInfo` struct in Fig. 3). This allowed for a saving of approximately 30 000 units of gas per call of `setIPFSInfo`. Secondly, we have encoded attributes into numeric identifiers to avoid the usage of strings for denumerable entities, thereby saving more gas units as the length of the attribute array increases. Further gas-consumption optimisation techniques may be achievable especially for the attribute checking. This challenge paves the path for future work.

## 6 Related work

Over the last few years, several research endeavours have been dedicated to the automation of collaborative processes based on blockchain. Weber et al. [29] present a technique

Pre-print copy of the manuscript published by Springer (available at `link.springer.com`)
identified by DOI: `10.1007/978-3-031-16168-1_4`

that resorts to blockchain technology to execute business process between parties who do not trust each other. In their seminal work, they show how the actors can find a mutual agreement on the enacted behaviour without the need to trust a central authority for its enforcement. López Pintado et al. [16] present Caterpillar, a blockchain-based BPMN execution engine. Caterpillar allows users to create instances of a process and to monitor their status. Tran et al. [27] introduce Lorikeet, a model-driven engineering (MDE) tool to implement business processes on chain for the management of assets (e.g., cars, houses), thereby proposing a solution for a scenario that traditionally relies on central authorities. Di Ciccio et al. [8] describe how to design and run business processes where several parties are involved, present the building blocks of model-driven approaches for blockchain-based collaborative business processes with a comparison between Caterpillar and Lorikeet. López Pintado et al. [15] present a model to dynamically bind the actors in a multi-party business process to roles and a specification language for binding policies. CAKE can handle dynamic role binding as the attributes are set by the Attribute Certifier possibly at run time or deploy time. Access keys are generated upon request and not before the process starts. Madsen et al. [17] investigate distributed declarative workflow execution where the collaboration is among adversaries. In such settings, the involved parties do not trust each other and they can also suspect that a party might not act like established. In this work, the authors demonstrate that the execution of the distributed declarative workflow could be implemented as a Smart Contract while ensuring the enforcement of workflow semantics and notarisation of the execution history. Corradini et al. [5] present ChorChain. It takes a BPMN choreography model as input and outputs its translation into a Solidity Smart Contract. ChorChain also allows auditors to obtain ex-post and runtime information on the process instances. These works undoubtedly contribute to the integration of blockchain and process management thus unlocking security and traceability opportunities. However, they do not include mechanisms to ensure fine-grained access control to data saved on a public platform. In contrast, our work precisely focuses on this aspect in a collaborative business process scenario.

Another branch of research work that pertains to our investigation area is the privacy and integrity of data stored on chain. Several papers in the literature document the adoption of encryption to this extent. Hawk [12] is a decentralised system that automatically implements cryptographic devices based on user-defined private Smart Contracts. We take inspiration from this work in that we resort to policies backed by Smart Contracts to cipher messages. Bin Li et al. [13] present RZKPB, a privacy protection system for transactions in shared economy built upon blockchain. This method does not require third trusted parties and preserves transaction privacy as it does not store the financial transactions publicly on chain. Their methodology relates with ours in that we resort to external data stores to save data too, yet we link it with transactions on the ledger. In [14], the authors describe FPPB, a fast privacy protection method based on licenses. It uses zero-knowledge proof, secret address and encryption primitives in the blockchain. Thanks to these features, it grants consistency without disclosing data. This architecture can be used in several shared economic applications. Rahulamathavan et al. [23] propose a new privacy-preserving blockchain architecture for IoT applications based on Attribute-Based Encryption techniques. We employ ABE too, yet with the objective of enhancing

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

existing architectures with our approach. In contrast, this model aims at changing the blockchain protocol at its core. Benhamouda et al. [2] present a solution that allows a public blockchain to act as a repository of secret data. In their system, at first, a secret is stored on chain, then the conditions under which to release it are specified and, finally, the secret is disclosed if and only if the conditions are met. In our approach, we employ shared secrets among components but we do not use the blockchain as a storage for secret data nor expect to disclose the secret. Differently from the techniques above, we tackle the problem of controlled data access in a multi-party process scenario, wherein several information artefacts are exchanged and different actors can access (parts of) messages based on fine-grained policies.

Wang et al. [28] present a secure electronic health record system wherein they combine ABE, Identity-Based Encryption (IBE) and Identity-Based Signature (IBS) with the blockchain technology. This architecture differs from CAKE because in this case the hospital owns the data about patients, and patients specify the policies. In our case, no authority is intended to manage the data except the data owners themselves – in healthcare processes, e.g., they would be the patients. Pournaghi et al. [22] provide a scheme based on blockchain technology and attribute-based encryption, named MedSBA. Their architecture differs from ours for two main reasons. Firstly, MedSBA makes use of two private blockchains, whereas we consider a public-blockchain scenario. Secondly, they cipher the data with AES symmetric cryptography with a random key and then they cipher that random key via ABE. By ciphering with the AES encryption scheme, MedSBA does not allow different users to read the same message, or slices thereof.

## 7 Conclusion and future remarks

In this work, we have proposed CAKE, an approach that combines blockchain technology and Attribute-Based Encryption (ABE) to control data access in the context of a multi-party business process. Our approach also makes use of IPFS to store information artefacts, access policies and meta-data. We employ Smart Contracts to store the user attributes, determining the access granted to the process actors, and the link to IPFS files. CAKE provides a fine-granular specification of access grants, data integrity, permanence and non-repudiability, allowing for auditability with minor overheads.

An important aspect to analyse in future studies is the integration with alternative encryption methods. For example, Odelu et al. [21] propose an RSA-based CP-ABE scheme with constant-size secret keys and ciphertexts (CSKC). Their approach targets high efficiency for limited-battery devices. The adoption of CSKC could be of help to integrate IoT devices in the management of blockchain-based processes. Key-Policy Attribute-Based Encryption (KP-ABE) [10] seems a promising asset for a more agile management of the process instance identifiers (case ids). With KP-ABE, attributes are associated with the ciphertext while the policy is associated with users, so the latter can decrypt the ciphertext only if the attributes of the encrypted text satisfy the user policy.

We also plan to overcome existing limitations of our approach. If a Data Owner wants to revoke access to data for a particular Reader, e.g., they can change the policy and cipher the messages again. However, the old data on IPFS would still be accessible. To overcome this limitation, we are considering the usage of InterPlanetary Name System

Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

(IPNS), as it allows for the replacement of existing files, hence the substitution of a message with a new encryption thereof. Furthermore, we plan to turn the SDM and SKM into distributed components in order to make our architecture more robust. We are investigating the adoption of secure multi-party computation schemes [6] to this end.

The integration of CAKE with existing blockchain-based process automation toolkits such as Caterpillar [16], Lorikeet [27] and ChorChain [5] is an interesting research avenue as well. CAKE can complement the control-flow-centric perspectives of the above tools with the data access control facilities it provides. To this end, the automated translation of task-based authorisation constraints to policies would be part of the endeavour [30]. Lorikeet specifically includes methods for on-chain data management, which CAKE can complement for confidential off-chain data. As we resort to IPFS to store data, though, the integration should include oracles that permit Smart Contracts to interact with off-chain data [19,1]. The system designer would then be able to determine the trade-off between full transparency on the decision process and access control, by balancing the on-chain and off-chain storage of data as discussed in [11]. Finally, we aim to implement this system with other public blockchains in the future (e.g., Algorand [4]) and test this system with real-world multi-party business processes in production.

**Acknowledgements.** The work of E. Marangone and C. Di Ciccio was partially funded by the Cyber 4.0 project BRIE and by the Sapienza research projects SPECTRA and "Drones as a service for first emergency response".

## References

1. Basile, D., Goretti, V., Di Ciccio, C., Kirrane, S.: Enhancing blockchain-based processes with decentralized oracles. In: BPM (Blockchain and RPA Forum). pp. 102–118 (2021)
2. Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T., Reyzin, L.: Can a public blockchain keep a secret? In: TCC (2020)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: SP. pp. 321–334 (2007)
4. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theor. Comput. Sci. **777**, 155–183 (2019)
5. Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Tiezzi, F.: Engineering trustable and auditable choreography-based systems using blockchain. ACM Trans. Manage. Inf. Syst. **13**(3) (feb 2022)
6. Cramer, R., Damgård, I.B., et al.: Secure multiparty computation. Cambridge University Press (2015)
7. Dannen, C.: Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. Apress (2017)
8. 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 (May 2019)
9. Feng, Q., He, D., Zeadally, S., Khan, M.K., Kumar, N.: A survey on privacy protection in blockchain system. Journal of Network and Computer Applications **126**, 45–58 (2019)
10. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. IACR Cryptol. ePrint Arch. p. 309 (2006), http://eprint.iacr.org/2006/309

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-031-16168-1_4

11. Haarmann, S., Batoulis, K., Nikaj, A., Weske, M.: Executing collaborative decisions confidentially on blockchains. In: BPM (Blockchain and CEE Forum). pp. 119–135 (2019)
12. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 839–858 (2016)
13. Li, B., Wang, Y.: RZKPB: A privacy-preserving blockchain-based fair transaction method for sharing economy. In: TrustCom/BigDataSE. pp. 1164–1169 (2018)
14. Li, B., Wang, Y., Shi, P., Chen, H., Cheng, L.: FPPB: A fast and privacy-preserving method based on the permissioned blockchain for fair transactions in sharing economy. In: IEEE Intl. Conf. TrustCom/BigDataSE. pp. 1368–1373 (2018)
15. López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Controlled flexibility in blockchain-based collaborative business processes. Inf. Syst. **104**, 101622 (2022)
16. 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)
17. Madsen, M.F., Gaub, M., Høgnason, T., Kirkbro, M.E., Slaats, T., Debois, S.: Collaboration among adversaries: Distributed workflow execution on a blockchain. In: FAB. pp. 8–15 (2018)
18. Mendling, J., Weber, I., Aalst, W.V.D., et al.: Blockchains for business process management - challenges and opportunities. ACM Trans. Manage. Inf. Syst. **9**(1), 4:1–4:16 (Feb 2018)
19. Mühlberger, R., Bachhofner, S., Ferrer, E.C., Di Ciccio, C., Weber, I., Wöhrer, M., Zdun, U.: Foundational oracle patterns: Connecting blockchain to the off-chain world. In: BPM 2020 Blockchain and RPA Forum. pp. 35–51. Springer (2020)
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf
21. Odelu, V., Das, A.K., Khurram Khan, M., Choo, K.K.R., Jo, M.: Expressive cp-abe scheme for mobile devices in iot satisfying constant-size keys and ciphertexts. IEEE Access **5**, 3273–3283 (2017)
22. Pournaghi, S., Bayat, M., Farjami, Y.: Medsba: a novel and secure scheme to share medical data based on blockchain technology and attribute-based encryption. Journal of Ambient Intelligence and Humanized Computing **11** (11 2020)
23. Rahulamathavan, Y., Phan, R.C.W., Rajarajan, M., Misra, S., Kondoz, A.: Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption. In: ANTS. pp. 1–6 (2017)
24. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems (reprint). Commun. ACM **26**(1), 96–99 (1983)
25. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT. p. 457–473 (2005)
26. Stiehle, F., Weber, I.: Blockchain for business process enactment: A taxonomy and systematic literature review. In: BPM Blockchain Forum (Sep 2022)
27. 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 (2018)
28. Wang, H., Song, Y.: Secure cloud-based ehr system using attribute-based cryptosystem and blockchain. Journal of medical systems **42**(8), 152 (2018)
29. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: BPM. pp. 329–347 (2016)
30. Wolter, C., Schaad, A.: Modeling of task-based authorization constraints in BPMN. In: BPM. vol. 4714, pp. 64–79 (2007)
31. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger (2014), https://ethereum.github.io/yellowpaper/paper.pdf
32. Xu, X., Weber, I., Staples, M.: Architecture for Blockchain Applications. Springer (2019)
33. Zhang, R., Xue, R., Liu, L.: Security and privacy on blockchain. ACM Comput. Surv. **52**(3) (2019)


Pre-print copy of the manuscript published by Springer (available at link.springer.com)
identified by DOI: 10.1007/978-3-031-16168-1_4

This document is a pre-print copy of the manuscript
(Marangone, Di Ciccio, and Weber 2022)
published by Springer (available at link.springer.com).

The final version of the paper is identified by DOI: 10.1007/978-3-031-16168-1_4

## References

Marangone, Edoardo, Claudio Di Ciccio, and Ingo Weber (2022). "Fine-Grained Data Access Control for Collaborative Process Execution on Blockchain". In: *BPM Blockchain, RPA and CEE Forum*. Ed. by Andrea Marrella, Raimundas Matulevičius, Renata Gabryelczyk, Bernhard Axmann, Vesna Bosilj Vukšić, Walid Gaaloul, Mojca Indihar Štemberger, Andrea Kő, and Qinghua Lu. Lecture Notes in Business Information Processing. Springer, pp. 51–67. ISBN: 978-3-031-16168-1. DOI: 10.1007/978-3-031-16168-1_4.

## BibTeX

```
@InProceedings{   Marangone.etal/BPMBCF2022:AccessControlBlockchain,
  author        = {Marangone, Edoardo and Di Ciccio, Claudio and Weber,
                   Ingo},
  booktitle     = {BPM Blockchain, RPA and CEE Forum},
  title         = {Fine-Grained Data Access Control for Collaborative Process
                   Execution on Blockchain},
  year          = {2022},
  pages         = {51--67},
  crossref      = {BPMBCRPACEE2022},
  doi           = {10.1007/978-3-031-16168-1_4},
  keywords      = {Attribute Based Encryption; Blockchain; Business process
                   management; IPFS}
}
@Proceedings{     BPMBCRPACEE2022,
  title         = {Business Process Management: Blockchain, Robotic Process
                   Automation, and Central and Eastern Europe Forum, {BPM}
                   2022 Blockchain, {RPA}, and {CEE} Forum, M{\"{u}}nster,
                   Germany, September 11-16, 2022, Proceedings},
  year          = {2022},
  editor        = {Marrella, Andrea and Matulevi{\v{c}}ius, Raimundas and
                   Gabryelczyk, Renata and Axmann, Bernhard and Bosilj
                   Vuk{\v{s}}i{\'{c}}, Vesna and Gaaloul, Walid and Indihar
                   {\v{S}}temberger, Mojca and K{\H{o}}, Andrea and Lu, Qinghua},
  isbn          = {978-3-031-16168-1},
  publisher     = {Springer},
  series        = {Lecture Notes in Business Information Processing}
}
```