12-23-2019

Blockchain based Access Control for Enterprise Blockchain Applications

Lei Xu
The University of Texas Rio Grande Valley, lei.xu@utrgv.edu

Isaac Markus
Conduent Technology Innovation

Subhod I
Conduent Technology Innovation

Nikhil Nayab
Conduent Technology Innovation

Recommended Citation

Xu, L, Markus, I, I, S, Nayab, N. Blockchain-based access control for enterprise blockchain applications. Int J Network Mgmt. 2020; 30:e2089. https://doi.org/10.1002/nem.2089

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Blockchain based Access Control for Enterprise Blockchain Applications

Lei Xu[1] | Isaac Markus[2] | Subhod I[3] | Nikhil Nayab[4]

[1]University of Texas Rio Grande Valley, Texas, United States
[2]Conduent Technology Innovation, North Carolina, United States
[3]Conduent Technology Innovation, Bangalore, India
[4]Conduent Technology Innovation, New Jersey, United States

Correspondence
*Lei Xu, 1 W University Blvd, Brownsville, Texas, 78520, USA. Email: xuleimath@gmail.com

Present Address
1 W University Blvd, Brownsville, Texas, 78520, USA

Abstract

Access control is one of the fundamental security mechanisms of IT systems. Most existing access control schemes rely on a centralized party to manage and enforce access control policies. As blockchain technologies, especially permissioned networks, find more applicability beyond cryptocurrencies in enterprise solutions is expected that the security requirements will increase. Therefore, it is necessary to develop an access control system that works in a decentralized environment without compromising the unique features of a blockchain. A straightforward method to support access control is to deploy a firewall in front of the enterprise blockchain application. However, this approach does not take advantage of the desirable features of blockchain. In order to address these concerns, we propose a novel blockchain based access control scheme, which keeps the decentralization feature for access control related operations. The newly proposed system also provides the capability to protect user's privacy by leveraging ring signature. We implement a prototype of the scheme using Hyperledger Fabric, and assess its performance to show that it is practical for real world applications.

KEYWORDS:
blockchain; access control; privacy

1 | INTRODUCTION

Access control is one of the basic security mechanisms deployed in almost every IT system. A general access control includes authentication and authorization. Specifically, the system should be able to recognize a user, make decisions based on pre-defined rules, and enforce the decision. Many works have been done on access control to improve its efficiency and flexibility[1,2,3]. Although there are different ways to implement an access control, most existing systems share one common feature: they rely on a trusted centralized party to manage and enforce the access control policies.

The current form of access control is not compatible with the emerging blockchain technology. A blockchain is maintained by a set of participants, and each of them keeps a local copy of the whole ledger. When a new record needs to be added to the ledger, all participants run a consensus protocol and utilize their local copies of the ledger to determine whether it should be accepted or not. This construction model brings a variety of useful features, including immutability, public/unified accessibility, and resilience[4]. Since there is no centralized party to rely on, the standard paradigm is not compatible requiring novel access control protocols for users of a blockchain.

This is not a problem for cryptocurrency systems like Bitcoin[5], Primecoin[6], and Zerocoin/Zerocash[7,8], where the blockchain is leveraged for double-spending prevention and currency issuing, where all information is intentionally disclosed to everyone.

However, for permissioned blockchains used to implement enterprise applications such as supply chain management [9], enterprise identity management [10], and business process management [11,12], a lack of access control becomes a big concern. For these types of applications, there are multiple organizations involved that require an ability to transact with each other while maintaining selected information confidential from certain parties. Although each participant can still have its own access control system such as active directory service, it does not help much as it only works for the organization itself and it is hard to coordinate across the boundaries of different organizations.

To handle this problem, several existing platforms scarify the desirable features of blockchains to support certain level of separation for access control. For example, Corda [13] allows a subset of organizations to maintain a "sub-ledger" to store information they do not want to disclose to others. Hyperledger Fabric has both a channel and collections features for peer to peer communications separate from other participants [14]. Although this approach partially achieves access control, it is far away from perfect. For example, it causes information fragmentation and pushes all access control burdens to the users. Furthermore, when organizations join and leave the system, the number of "sub-ledgers" increases creating latency and resilience issues.

To overcome these challenges, we propose a **d**ecentralized ledger/blockchain based **ac**cess **c**ontrol scheme (DAcc) for enterprise applications. DAcc is built on permissioned blockchain and focuses on data protection, which utilizes several cryptography tools to enforce access control in a privacy friendly manner. DAcc is fully compatible with blockchain and can be easily integrated with different blockchain applications.

In summary, our contributions in this paper include:

- We propose the architecture of DAcc and detailed design of key protocols to protect both the data and users;
- We analyze and prove the security features of DAcc;
- We implement the prototype of DAcc using Fabric and describes details of the implementation; and
- We evaluate the performance of the prototype to demonstrate its practicability.

The rest of the paper is organized as follows: Section 2 gives a brief overview of blockchain and its applications. Section 3 describes the high-level architecture of DAcc and detailed description of key protocols are given in Section 4. We analyze the security features of DAcc in Section 5. Section 6 discusses the implementation details of DAcc. Section 7 provides performance evaluation of DAcc referencing previously studied Hyperledger Fabric capabilities. We discuss related works in Section 8 and conclude the paper in Section 9.

2 | BACKGROUND

In this section, we briefly review the background of blockchain technology. Blockchains can be roughly divided into two categories, public ledgers and permissioned ledgers. The most critical difference between these two types of decentralized systems is whether a user can directly join a network or if it needs permission and credentials to join and transact in the system. For a public blockchain, a user can join and leave the system freely. But for a permissioned blockchain, one needs to obtain an identity that can be recognized by other peers before he/she can participate in the system. This identity is critical for signing and verifying transactions. Permissioned blockchains are more suitable for enterprise applications since participants are usually predefined and exposure of internal application to the public increases liability. In this work we focus on permissioned blockchains since the underlying assumptions of a private network are needed for our solution.

Most blockchain consensus mechanisms can be used to build a permissioned blockchain, such as proof-of-work with longest chain principle, proof-of-stake [15], Byzantine fault tolerate protocol and its variants [16,17]. The underlying consensus protocol does not affect the operations of DAcc built on top of the blockchain. Considering the typical characters of an enterprise application, we choose to use Hyperledger Fabric for the prototype. Hyperledger Fabric utilizes an endorsing-and-ordering mechanism for consensus, where a digital signature is used for proof of endorsement and Kafka [18] is used for ordering service.

3 | OVERVIEW

In this section, we give an overview of privacy preserving access control scheme for enterprise blockchain applications, and discuss the security assumptions and objectives.

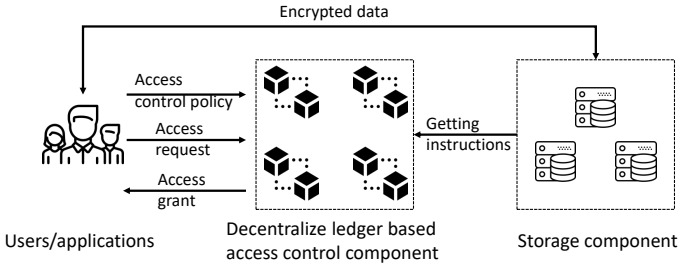

FIGURE 1 Overview of the blockchain based access control system. The blockchain is the only interface between the user and the data storage component. The system is designed to be compatible with a Hadoop distributed file system or other storage solutions that provide data resiliency through replication.

3.1 | Scheme Participants

For an enterprise application, there are different types of resources that can be protected using access control. In this paper, we focus on data access control, i.e., to control who can read and modify data managed by the system. The scheme consists of two major components:

- The blockchain based access control component, which takes care of most access control related operations and meta data management. This module is maintained by a group of nodes which work on behalf of participating enterprise organizations using the pre-defined endorsement policy.
- The data storage component, which is responsible for storing all the data managed by the system, and follows instructions issued by the blockchain based access control component to operate.

There is another type of participants of the system, who are users/applications that interact with the system to access data stored in the storage component. **FIGURE** 1 summarizes the high-level architecture of the proposed scheme.

3.2 | Security Assumptions and Objectives

We summarize the security assumptions of the three types of participants of DAcc as follows:

- The blockchain access control component is trusted as a whole, i.e., based on the selection of the underlying consensus mechanism, this component can tolerate a certain ratio of malicious nodes.
- The data storage component is semi-trusted, i.e., it follows pre-defined protocol, but will try its best to learn stored data.
- Users who own the data are fully trusted, but users who want to access others' data are not trusted.

DAcc aims at providing a blockchain based access control mechanism for data. More specifically, it achieves three objectives:

- Customizable and flexible access control rules for stored files. Data owner should be able to define the access control rules in a flexible manner, e.g., applying the rule to a group of data files, delegating the right to update the rule, and adding expiration information.
- Detecting faulty/malicious node(s) in the blockchain. In the decentralized environment, it is possible that a node fails, or responds incorrectly for access control related operations. In this case the system should detect such events and fix the node or remove it from the system.
- Privacy of access activities. When a user submits a request to access a data file, this information will be stored on the blockchain and can be accessed by all participants. The system should provide a privacy protection mechanism to hide a user's activities without affecting the access control operations.

4 | DETAILED DESIGN OF PRIVACY PRESERVING BLOCKCHAIN BASED ACCESS CONTROL

In this section, we describe the detailed design of the blockchain based access control scheme and analyze its security features.

The blockchain based access control scheme enforces access control through encryption, i.e., all shared data are stored in the form of cipher-texts, and only authorized users can get the corresponding decryption key. Therefore, data confidentiality is not a problem for the storage component. Without loss of generality, we assume each participant in the system has a public/private key pair (pk, sk) where the public key is embedded into a certificate that everyone recognizes. The blockchain based access control scheme consists of three protocols:

- Data and access policy uploading. This protocol is used by a data owner to submit his/her data file and corresponding access control policy to the system.
- Data access request submission. A user uses this protocol to prepare and submit an access request of a data file to DAcc.
- Access grant. DAcc uses this protocol to process a data access request, and the decision is made based on consensus of the system.
- Data access. This protocol is used by a user to read/write a data file if the access request is approved by DAcc.

We also discuss supporting functions like access revoking, data file deletion, and fault tolerant supporting in this section.

4.1 | Data and Access Policy Initialization

To upload a data file to the system, the user *owner* follows Protocol 1 to prepare all the information. In Protocol 1, *policy* there is a list of public keys belonging to users who can read the data, $\texttt{Enc}_{\texttt{Asym}}$ is an asymmetric encryption scheme such as RSA encryption, and `Hash` is a cryptography hash function such as SHA256. *meta* is the data structure storing information about the data file, including the unique identifier of the data file, the owner identity, creation time, and hash value of the data file.

Note that interactions between the blockchain and the storage component are not reflected in Protocol 1. After *owner* submits meta data, access policy and split keys to the blockchain, a smart contract is activated to check whether the transaction is valid, and the consensus mechanism is used to determine whether the transaction will be added to the ledger. When the *owner* submits c to the storage system, the storage system checks the ledger, and only accepts the request if the corresponding transaction is included in the ledger.

Protocol 1 Protocol for the data owner to submit data and access policy to the system.

Require: The data file *data*, the access policy *policy*, integer t and n

$dek \xleftarrow{\$} \{0,1\}^{\ell}$
$c \leftarrow \texttt{Enc}_{\texttt{AES}}(data, dek)$
$meta_{data} \leftarrow \texttt{Meta}(data)$
$K \leftarrow \texttt{Split}(dek, t, n)$, where K is the set of split keys and the size of K is $|K| = n$
owner selects a group of nodes $N = \{N_1, \ldots, N_n\}$ from the ledger
for do $N_i \in N, i = 1, \ldots, n$
    $kc_i \leftarrow \texttt{Enc}_{\texttt{Asym}}(k_i, pk_{N_i}), k_i \in K$
    $hkc_i \leftarrow \texttt{Hash}(kc_i)$
end for
Submitting $meta_{data}$, *policy*, and $\{(kc_1, hkc_1), \ldots, (kc_n, hkc_n)\}$ to the blockchain
Submitting c to the storage component

4.2 | Reading Access Request Submission

When a user needs to access a data file managed by the system, he/she has to go through the blockchain based access control system to obtain the data. If privacy is not a concern, the user can use his/her private key to generate a signature on the request, and every blockchain node can verify the signature and check the corresponding policy to determine whether the request should be accepted or not.

Protocol 2 describes the process that the *requester* prepares the request of accessing a data file in a privacy preserving way. Specifically, the function `Extract` returns a set of public keys stored in the access control policy *policy*, which must include the requester's public key. The *requester* also generates a new one-time public/private key pair (pk_t, sk_t) just for this request, and using the ring signature scheme to sign the public key pk_t as part of the request. *requester* then submits *req* to the blockchain system.

The purpose of the one-time public/private key pair is to break the potential connection between multiple access requests. If a user always uses the same key pair, it is easy for a passive adversary to learn his/her access pattern. Depending on the asymmetric encryption scheme $\mathtt{Dec}_{\mathtt{Asym}}/\mathtt{Enc}_{\mathtt{Asym}}$ used in Protocol 3 and Protocol 4, the one-time key pair is generated using different methods. For instance, if ECIES [19] is used, the requester random selects a positive integer (within a range) as the private key, and calculates the public key by calculating a scalar multiplication with the generated private key and the base point on an elliptic curve, which is a public parameter.

Protocol 2 Protocol for the requester to request data access.

Require: Identity of the data file id_d

1: Fetching the *policy* of the data file id_d from its meta data, which is stored on the blockchain
2: $PK \leftarrow \mathtt{Extract}(policy)$
3: Randomly selects a temporary public/private key pair (pk_t, sk_t)
4: $\sigma \leftarrow \mathtt{RingSign}(pk_t, PK, sk_{requester})$
5: $req \leftarrow (id_d, pk_t, PK, \sigma)$
6: **return** *req*

4.3 | Reading Access Granting

Access granting is handled by a smart contract deployed on the blockchain. The procedure is given in Protocol 3. Since the request is sent to all participating nodes of the blockchain, each node can verify whether the access request should be proved or rejected. If a malicious node proves an illegal request and tries to publish the response on the blockchain, other nodes can detect that and refuse to include the corresponding transaction. Therefore, as long as there are enough honest nodes in the system, such a transaction will not be included in the blockchain. If the unauthorized user colludes with a malicious node, the node can share its piece of secret using off-chain communication channel. However, the unauthorized user needs to collude with multiple nodes where the number is larger than the threshold to recover the data encryption key.

4.4 | Data Accessing

After the DAcc blockchain grants access to read a data file, the requester follows the Protocol 4. In this process, the requester checks the integrity of the received split keys with their hash values stored in the blockchain to detect and avoid malicious node(s).

Depending on the implementation of the storage system, the data fetching request is processed in different ways. When a typical distributed storage system working in the master-slave manner (e.g., Hadoop) is used, the fetch request is sent to the master node to process. Instead of responding to the request directly, the master node first queries the blockchain to check whether the received request has been approved or not. If the master node receives positive respond, it instructs slave nodes to send data to the requester. We also discuss the use of a fully decentralized storage system in Section 6.2.

Protocol 3 Protocol for DAcc to grant data access.

Require: Accessing request *req*
for Each node N_i on the ledger **do**
  Parsing *req* to (id_d, pk_t, PK, σ)
  if PK does not belong to the *policy* attached to the data file id_d **then**
    return `Reject`
  end if
  if The digital signature σ is NOT valid using `RingVerify` and PK **then**
    return `Reject`
  end if
  if The request passes these verification **then**
    $k_i \leftarrow \texttt{Dec}_{\texttt{Asym}}(kc_i, sk_{N_i})$
    $kc_i' \leftarrow \texttt{Enc}_{\texttt{Asym}}(k_i, pk_t)$
    Submitting kc_i' to the blockchain
  end if
end for

Protocol 4 Protocol for the requester to access the data file.

$\mathcal{K} \leftarrow \emptyset$
for $i = 1$ to n **and** $|\mathcal{K}| < t$ **do**
  $k_i' \leftarrow \texttt{Dec}_{\texttt{Asym}}(kc_i', sk_t)$
  if $\texttt{Hash}(k_i') == hkc_i$ **then**
    Adding k_i' to $\mathcal{K}$
  end if
end for
if $|\mathcal{K}| == t$ **then**
  $dek \leftarrow \texttt{Reconstruct}(\mathcal{K}, t, n)$
  Fetching c, the cipher-text of the target data file
  $data \leftarrow \texttt{Dec}_{\texttt{AES}}(c, dek)$
end if

4.5 | Data Updating Request and Processing

To supporting data updating, the owner of the data can add a privilege field to the access control policy to determine whether the user can update a data file or just read it. The updating process is similar to the data submitting protocol given in Protocol 1. Except that the user needs to provide information of the old version of the data file, and nodes on the blockchain will check whether this request is valid or not according to the access control policy of the old version data file. The meta data of the new version also includes a pointer pointing to its previous version.

DAcc works in the appending-only manner, and it does not delete the old version of the data file after a new version is submitted to the system. With this feature, DAcc allows users to track the history of a data file easily.

4.6 | Other Supporting Functions

Besides the major functions described above, there are several other functions that are necessary for DAcc.

4.6.1 | Access Revoking

The owner of a data file can submit a new access control policy for a data file to DAcc to revoke a user. If the revoked user submits access request after the new policy is accepted by DAcc, nodes on the ledger will reject the request. However, if the revoked user requested the same data file before the revoke, DAcc cannot prevent him/her from keeping a local copy of the data

encryption key to access the data file. Therefore, DAcc requires the user to use different data encryption keys for different data files and different versions of the same data file.

4.6.2 | Data File Deletion

DAcc consists of two parts, the blockchain part is completely append-only and does not support deletion operations. But the user can request to erase his/her data file from the storage component. Specifically, DAcc does the following steps: (i) The owner of the data file sends a request to delete the file to the blockchain; (ii) Nodes on the blockchain checks the request and accepts it as a new transaction if it is valid; (iii) The storage component checks the blockchain, and it deletes the data file if the deletion transaction has been confirmed on the ledger. After the deletion operation is done, even if one gets the data encryption key, he/she cannot access the data file anymore.

4.6.3 | Blockchain Node Replacement

It is possible that one or more nodes of the blockchain fail or are compromised. In this case, some data files may become not accessible especially when the data owner sets up a policy with high security level, i.e., it requires every or almost every node of the blockchain to participate the access granting process. In this case, DAcc needs to support blockchain node replacement to maintain the availability of the service. DAcc adopts a general way to handle this situation. Specifically,

- Encryption key storage. When the data owner submits a new data file and initializes related information, he/she also encrypts the secret key using his/her private key and stores the cipher-text in a transaction of the blockchain.
- When the current setup fails (e.g., some nodes failed),
  - The owner submits a new transaction to the blockchain to revoke previous configuration.
  - Based on the availability of blockchain nodes, the owner re-do the initialization and splits/distributes key pieces to available blockchain nodes that are selected.
- Existing nodes of the blockchain use the updated information to process received request.

5 | SECURITY ANALYSIS

The blockchain based access control scheme satisfies four security features: access control, integrity, confidentiality, and privacy, which map to the requirements described in Section 3. In the following we give detailed analysis of these features.

5.1 | Access Control and Data Confidentiality

Access control and data confidentiality mean only authorized users can access the data file managed by DAcc. DAcc achieves access control and protects data confidentiality through the combination of encryption and the consensus mechanism employed by blockchain. Data files are always in the form of cipher-text in the storage component, and data confidentiality is reduced to the safety of data encryption key and the size of the network. According to the design of DAcc, a data encryption key is split into multiple pieces that are kept by different nodes of the blockchain. Therefore, a larger system with more nodes will be more secure as an adversary may need to collude with a higher number of nodes to recover the data encryption key. The probability that an adversary can succeed also depends on the parameters of the secret sharing scheme (i.e., integers t and n) that the user selected. For the same n, a larger t means it is harder for the adversary to break the system.

5.1.1 | Integrity and Usefulness

A legitimate user can only access a data file if he/she collects enough split key pieces from nodes of the blockchain. If one of these key pieces is wrong, the user cannot successfully decrypt the data file he/she gets from the storage component. DAcc stores hash values of original key pieces on the ledger and the immutability feature prevents an adversary from modifying them. Therefore, the user can always verify the received key pieces with these hash values (as described in Protocol 4) to guarantee the integrity of received information.

5.2 | Privacy

DAcc leverages ring signature and one-time public/private key to protect users' privacy in the process of access data files.

- Because the public/private key pair used to receive split keys is only used once and generated in a random way, an adversary cannot connect two key pairs submitted for the access of the same or different data files. Therefore, a key pair does not compromise the user's privacy.
- The randomly generated public key needs to be certified so a blockchain node can check it against the corresponding policy. Ring signature plays a key role here to protect the user's privacy. According to the security features of a ring signature scheme, one cannot link a ring signature with a specific singing public key included in the "ring". It is easy to see that the size of the ring determines the level of privacy, i.e., the larger the ring is, the higher privacy level one can obtain. For DAcc, the size of the ring is also limited by the size of the access control list. If there is only one user is allowed to access a data file, DAcc cannot provide adequate privacy protection. To mitigate this risk, a data owner should avoid using such policies and always provides a set of dummy users to help hide real users.

5.3 | Combination of Multiple Privacy and Security Components

A common misunderstanding of cryptography primitives is that using several secure components together automatically gives all the security/privacy features of each component. Unfortunately, this is not the case and using two secure components together without careful design can compromise all the security features [20].

DAcc addresses this concern using the universal composable (UC) security framework [20]. The UC feature guarantees that if one cryptography primitive is secure under the UC framework, it can be used together with other primitives that are also secure under the UC framework. It has been approved that the digital signature scheme is secure under UC framework [21], and one can construct a ring signature scheme that is also secure under the UC framework [22], we only need to prove that the one-time key pair generation is also secure under UC framework. To prove that the one-time key pair generation is secure under UC framework, we need to demonstrate that one cannot distinguish a simulated execution of one-time key pair generation and a real execution of the process. The one-time key pair generation relies on a pseudo-random function to randomly select the key pair. In practice, we can use an HMAC based on cryptographic hash function as the pseudo-random function, which has been proved to be secure under the UC framework [23].

6 | IMPLEMENTATION

In this section, we first describe the prototype we implemented and discuss the possibilities of extending the design DAcc to support other storage systems.

6.1 | Prototype using HDFS and Hyperledger Fabric

We implement a preliminary prototype of DAcc utilizing Hyperledger Fabric [14,24,25] for the underlying blockchain and Hadoop HDFS [26] for the storage.

To demonstrate the major functions, we set up a test network consists of two organizations with two peers in each organization, and a single orderer for the entire network. As a blockchain platform targeting at enterprise applications, Hyperledger Fabric divides users into groups (i.e., channels, which are private "subnets" of communication between two or more specific network members) for coarse-grained and limited access control. In order to implement the secret sharing scheme outlined previously and associated protocols, the prototype uses a global-and-local ledger architecture as depicted in **FIGURE** 2, which is implemented using the channel feature. Under this architecture, each organization is enrolled into the main channel, and separate individual channels are set up for each organization. This architecture serves two main purposes: First it allows the split keys generated by the secret sharing scheme to remain separate and safe among organizations by storing them separately but indexing against a common key. Second, it allows for the general tracking of results generated by non-deterministic functions akin to the secret sharing scheme by local ledger being able to query global ledger for user defined business rules.

Hyperledger Fabric does not allow the execution of cross channel functions that modify a ledger (i.e. adding a block), protocols such as decryption of key splits and key reconstruction are implemented as platform wide service for aggregating and distributing

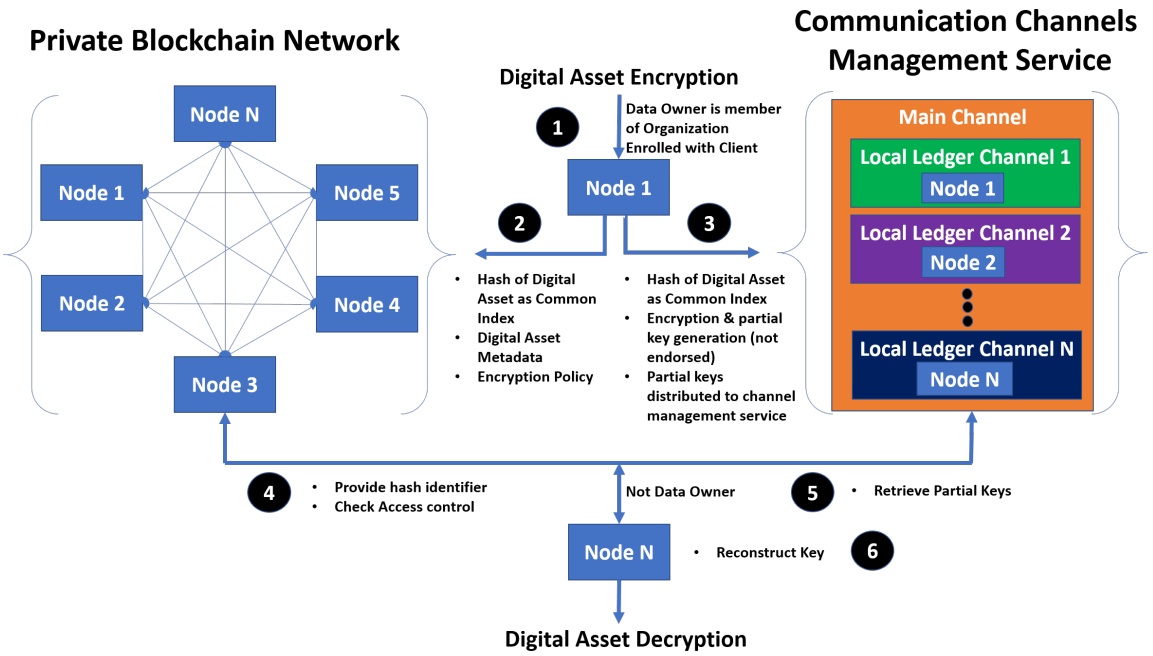

FIGURE 2 Architecture of prototype. Digital assets can represent different types of information in DAcc, e.g., the data encryption key and cipher-texts of key splits. Global distributed and local ledgers are accomplished via utilization of channels in Hyperledger Fabric to keep key splits separated and confidential among network peers.

response from queries to channels. For the posting of a key split to a local ledger, the coordination and execution service collects the information from the peer connected to data owner and disseminate to each peer the partial key information. For key reconstruction, the outside service also coordinates the query and response for each peer to aggregate individual responses. The critical aspect is that the coordination services only read in instructions endorsed and posted in the channels. Therefore, all the actions are consistent with the endorsement policy, which is part of the consensus protocol.

For the system implementation, two different chaincodes are developed in Go, and installed and instantiated into respective global and local channels. The Hyperledger Fabric `cryptogen` and `cryptoconfig` tools are used for creating and managing peer key pairs for transacting on network. Note that peer key pairs are different from those are involved in the operation of the access control mechanism. The application layer is managed utilizing the Fabric Node.js SDK to create a REST server for managing APIs to develop, deploy, and test chaincodes. For the file storage component, we use a local instantiation of Hadoop Distributed File System (HDFS). The communication with the HDFS cluster is managed via its own server composed using Haskell to setup up required REST APIs for managing directories and files in HDFS.

6.2 | Generalization of DAcc to Other Blockchain and Storage Systems

In fact, the design of DAcc and its security features do not rely on any specific blockchain or storage system. While it is straightforward to port DAcc to another permissioned blockchain platform and centralized storage system (distributed or not), it is not trivial to utilize a decentralized storage system such as IPFS[27,28]. IPFS by nature is a type of peer-to-peer storage system[27] that utilizes distributed hash table (DHT), where a data file is decomposed to chunks with fixed size and distributed to different nodes of the system. Since we only consider enterprise applications, all nodes know each other and a user can connect to any of them to access everything stored. The reading and writing operations are done in a similar way as HDFS, the only difference is that a user needs to interact with several nodes of IPFS to finish the operation instead of only interacting with the master node (name node) of HDFS.

The security assumptions also change with a purely peer-to-peer storage system like IPFS. For HDFS, it is relatively safe to make the assumption that the HDFS is well protected and follows the protocol. But it is hard to assume every node of IPFS is semi-trusted. In order to mitigate this risk, DAcc adopts different strategies for different operations:

- Reading. IPFS keeps multiple copies of the same data file using different nodes. To guarantee the correctness of a received file, an integrity tag is stored in the blockchain as part of the metadata of the data file. If a user obtains a file and finds out that the integrity tag is not consistent with the actual file, he/she can request the same file from other nodes and report to the system of the failed/malicious node(s).
- Writing. Writing/updating operation is more tricky than reading as a user cannot enforce an IPFS node to work honestly, i.e., an IPFS node can compromise the integrity of chunks of a data file that are assigned to it. To mitigate this risk, DAcc can adjust the redundancy rate of storage to prevent an adversary from compromising the submitted data file.

7 | PERFORMANCE EVALUATION

The cost of DAcc operation can be roughly divided into two categories:

- Cryptography operations, including data encryption, key partition/reconstruction, signature generation/verification. Most computation intensive operations only need to be done by a single party locally, and will not affect the overall performance. For example, data encryption and decryption are the most time consuming operations, especially when the size of the data file is large. But these will only need to be done by involved users locally. The only cryptography operations that adding an extra burden to the decentralization ledger nodes are key splits encryption and ring signature verification, and the cost of which are negligible for modern computers[29,30].
- Decentralized ledger operations. DAcc does not have any special requirements on the underlying decentralized ledger system, and the performance is mainly determined by the decentralized ledger system itself, which is Hyperledger Fabric in this work. In theory, consensus protocol is the most time consuming part of a decentralized ledger system. The version of Fabric used in the DAcc prototype utilizes the Kafka protocol for consensus purpose, which has very low latency with high throughput. In practice, many other factors need to take into consideration, such as network latency, and how busy a node. The performance of Fabric has been extensively studied by Thakkar et al.[31], which discussed impacts of transaction arrival rate, block size, endorsement policy, and number channels on performance. With optimization techniques, the underlying decentralized ledger platform is able to support more than 2K transaction per second with a typical setup[31].

8 | RELATED WORKS

In this section, we briefly review related works.

Several blockchain platforms implement access control by separating users into different groups and each group maintains its own ledger to share information[13]. Although this approach is useful for certain scenarios, it has several limitations as an access control mechanism. For example, if n users want to share different information with each other, there will be n^2 groups in the system. It also shifts the access control management burden to the users, who are responsible to decide when a new subgroup is required and manage existing subgroups.

There are also works on building access control using blockchain. Zyskind et al. developed a method that only allows the owner to access data managed by a blockchain but does not consider how to grant access to others, not to mention granting access in a privacy preserving way[32]. Ouaddah et al. discussed blockchain based access control for IoT[33,34]. The scenario considered by these works are totally different from our case as IoT devices can act according to instructions but data can always be read in our case. Maesa et al.[35] discussed general access control using blockchain but ignored the two features of privacy and error detection.

9 | CONCLUSION

The original idea of blockchain is to expose everything to the public to prevent incorrect actions such as double-spending. Although this has been proved to be a successful design for a cryptocurrency system without a trusted party, it brings a big challenge for applying blockchain for enterprise applications, where data confidentiality and privacy are critical, and cannot be disclosed to everyone. As one of the fundamental security tools, access control is a powerful tool to protect data confidentiality.

DAcc achieves the major functions of a classical access control mechanism without relying on a centralized party by utilizing a secret sharing scheme and an integration protection mechanism is used to prevent an adversary participant to release wrong information to users. To preserve users' privacy in interacting with DAcc, a ring signature scheme and a one-time key generation mechanism is used. In summary, DAcc achieves three major objectives besides basic access control function through the novel design: (i) Supporting customizable and flexible access control rules. (ii) Detecting faulty/malicious node(s) in the blockchain. (iii) Preserving privacy of access activities. The modular design of DAcc also makes it relatively independent of the underlying blockchain backbone and the storage system, which provides the end users more flexibility to build their own system. We also implement a prototype on top of Hyperledger Fabric which uses HDFS as the storage system to demonstrate the efficiency of the design in practice.

References

1. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. *Computer* 1996; 29(2): 38–47.
2. Bertino E, Bonatti PA, Ferrari E. TRBAC: A temporal role-based access control model. *ACM Transactions on Information and System Security (TISSEC)* 2001; 4(3): 191–233.
3. Frank M, Streich AP, Basin D, Buhmann JM. A probabilistic approach to hybrid role mining. In: ACM. ; 2009: 101–111.
4. Underwood S. Blockchain beyond bitcoin. *Communications of the ACM* 2016; 59(11): 15–17.
5. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system.; 2008.
6. King S. Primecoin: Cryptocurrency with Prime Number Proof-of-Work.; 2013.
7. Miers I, Garman C, Green M, Rubin AD. Zerocoin: Anonymous distributed e-cash from bitcoin. In: IEEE. ; 2013: 397–411.
8. Sasson EB, Chiesa A, Garman C, et al. Zerocash: Decentralized anonymous payments from bitcoin. In: IEEE. ; 2014: 459–474.
9. Xu L, Chen L, Gao Z, Lu Y, Shi W. Coc: Secure supply chain management system based on public ledger. In: IEEE. ; 2017: 1–6.
10. Gao Z, Xu L, Turner G, et al. Blockchain-based identity management with mobile device. In: ACM. ; 2018: 66–70.
11. Mendling J, Weber I, Aalst WVD, et al. Blockchains for business process management-challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)* 2018; 9(1): 4.
12. Diallo N, Shi W, Xu L, et al. eGov-DAO: a Better Government using Blockchain based Decentralized Autonomous Organization. In: IEEE. ; 2018: 166–171.
13. Brown RG, Carlyle J, Grigg I, Hearn M. Corda: An introduction. *R3 CEV, August* 2016.
14. Androulaki E, Barger A, Bortnikov V, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: ACM. ; 2018: 30.
15. Gilad Y, Hemo R, Micali S, Vlachos G, Zeldovich N. Algorand: Scaling byzantine agreements for cryptocurrencies. In: ACM. ; 2017: 51–68.
16. Vukolić M. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In: Springer. ; 2015: 112–125.
17. Ongaro D, Ousterhout JK. In search of an understandable consensus algorithm.. In: USENIX. ; 2014: 305–319.
18. Wang G, Koshy J, Subramanian S, et al. Building a replicated logging system with Apache Kafka. *Proceedings of the VLDB Endowment* 2015; 8(12): 1654–1655.
19. Smart NP. The exact security of ECIES in the generic group model. In: Springer. ; 2001: 73–84.

20. Canetti R. Universally composable security: A new paradigm for cryptographic protocols. In: IEEE. ; 2001: 136–145.

21. Canetti R. Universally composable signature, certification, and authentication. In: IEEE. ; 2004: 219–233.

22. Yoneyama K, Ohta K. Ring signatures: Universally composable definitions and constructions. *Information and Media Technologies* 2007; 2(4): 1038–1051.

23. Al-Kuwari S, Davenport JH, Bradford RJ. Cryptographic hash functions: recent design trends and security notions.. *IACR Cryptology ePrint Archive* 2011; 2011: 565.

24. Cachin C. Architecture of the hyperledger blockchain fabric. In: . 310. ACM. ; 2016.

25. Sousa J, Bessani A, Vukolic M. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In: IEEE. ; 2018: 51–58.

26. Borthakur D. HDFS Architecture Guide. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html; 2008.

27. Benet J. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561* 2014.

28. Chen Y, Li H, Li K, Zhang J. An improved P2P file system scheme based on IPFS and Blockchain. In: IEEE. ; 2017: 2652–2657.

29. Melchor CA, Cayrel PL, Gaborit P, Laguillaumie F. A new efficient threshold ring signature scheme based on coding theory. *IEEE Transactions on Information Theory* 2011; 57(7): 4833–4842.

30. Gupta V, Gupta S, Chang S, Stebila D. Performance analysis of elliptic curve cryptography for SSL. In: ACM. ; 2002: 87–94.

31. Thakkar P, Nathan S, Vishwanathan B. Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. *arXiv preprint arXiv:1805.11390* 2018.

32. Zyskind G, Nathan O, others . Decentralizing privacy: Using blockchain to protect personal data. In: IEEE. ; 2015: 180–184.

33. Ouaddah A, Abou Elkalam A, Ait Ouahman A. FairAccess: a new Blockchain-based access control framework for the Internet of Things. *Security and Communication Networks* 2016; 9(18): 5943–5964.

34. Ouaddah A, Elkalam AA, Ouahman AA. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In: Springer. 2017 (pp. 523–533).

35. Maesa DDF, Mori P, Ricci L. Blockchain based access control. In: Springer. ; 2017: 206–220.