# A Hyperledger Fabric Based Organizational Decentralized Access Control Solution

Sangat Das
*Pune Institute of Computer Technology, Pune, India*
sangatdas5@gmail.com

Chinmay Saraf
*Pune Institute of Computer Technology, Pune, India*
chinmay.saraf98@gmail.com

Devashish Pradeep Khairnar
*Pune Institute of Computer Technology, Pune, India*
khairnardevashish@gmail.com

***Abstract*—Currently, various organizations rely on third-party access control and resource management applications or develop their centralized application for resource management. Considering the pitfalls of present centralized applications, the proposed solution presents a decentralized approach towards organizational resource management and access control provisioning within or across organizational units. The research attempts to provide solutions for problems and risks associated with a centralized resource and access control management systems. This approach leverages the advantages of the permissioned Blockchain framework - Hyperledger Fabric and peer-to-peer network. The proposed solution also provides Single Sign-On authentication mechanism (SSO) for various organizational resources. Using the approach presented in the paper we can develop an efficient way of Team management with which we can achieve a higher degree of autonomy of subdivisions within an organization.**

## I. Introduction

When it comes to giving access to employees in various organizations, the said organizations rely on third party access control applications or develop their own application for the same. But, even though these applications are claimed to be very secure, they use centralized servers to maintain access records, access keys, etc. This creates a single point of failure and vulnerability for malicious hackers to get unauthorized access to confidential data or perform Denial of Service(DoS) attacks [1].

We believe using decentralized applications for such an access control mechanism makes the system more secure from the perspective of a single point of failure and attack. By using blockchain technology [2], which is also base technology for Bitcoin Cryptocurrency [3], we not only can ensure confidentiality of organizational data but also can increase availability for authenticated users and eliminate single point of failure by introducing multiple nodes in a network.

Hyperledger Fabric is a blockchain platform to create a private and permissioned ledger [4]. The consensus in hyperledger fabric consists of three phases namely Endorsement, Ordering, and Validation, which can be used to develop applications for the above-described scenarios. Hyperledger Fabric also has provisions to define policies for access control on various resources like chain code. The concept of "Channels" in hyperledger fabric allows us to create logical divisions within the organization which enables the creation of multiple subunits within the organization with each subunit having access only to their own transactions [5].

In the presented approach, we are utilizing the above-mentioned features of the hyperledger fabric framework to create a decentralized access control mechanism that can be deployed as a solution by an organization on its own infrastructure. In this approach, an organization can create its own verified peers and form a private permissioned network that can be trusted to provide access control to various resources owned by the organization.

## II. Related Work

Similar work in this field has been developed by Ellcrys [6], where researchers have proposed using a public blockchain network to manage open source repositories. Instead of using traditional consensus algorithms like Proof-of-Work (PoW) and Proof-of-Stake (PoS), they have resorted to using a hybrid approach, Proof-of-Activity [7] as a consensus model. In their work, they have addressed the need for a decentralized way to control open-source projects and have discussed a way through which collaborators can be rewarded in the form of their own native coins (Ell). They have also modeled their Transaction Endorser similar to the Hyperledger endorser concept which enables them to use general-purpose languages for writing smart contracts to solve the problem of non-determinism.

Another group of researchers in their paper [8], have addressed the need for a decentralized system to verify contributions to open source projects and maintain secure versions of software available to the general public. This paper has proposed how blockchain technology can be used to maintain secure source code binaries protected from attacks that can destroy repositories, divert updates, or inject malicious code snippets.

To implement flexible, diverse, and dynamic access control, this paper [9] introduced attribute-based access control (ABAC) into the blockchain. The access control logic of ABAC engines determines who should have access to what resources under what circumstances by taking what actions and is useful for an open environment, such as blockchain. The resource sharing platform in the paper can support flexible and diverse permission management, as well as a verifiable and transparent access process that integrates Transaction Based Access Control(TBAC) with standard ABAC and blockchain.

978-0-7381-0504-8/20/$31.00 ©2020 IEEE

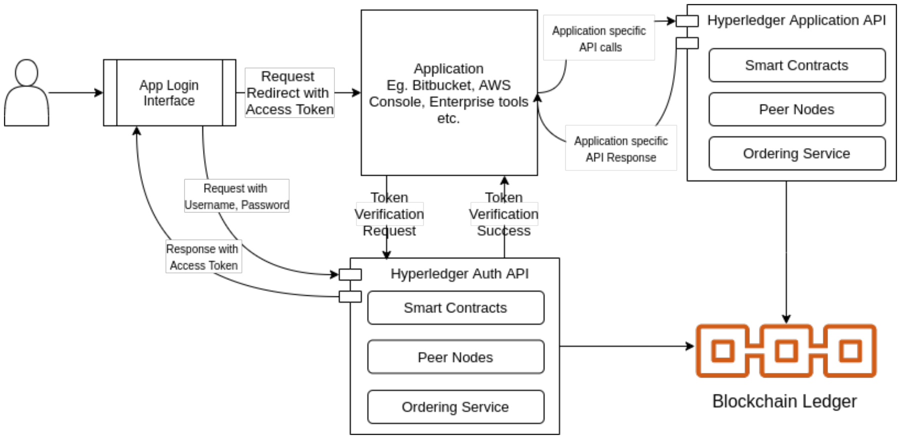
App Login Interface
Request Redirect with Access Token
Application Eg. Bitbucket, AWS Console, Enterprise tools etc.
Application specific API calls
Hyperledger Application API
Smart Contracts
Peer Nodes
Ordering Service
Application specific API Response
Request with Username, Password
Token Verification Request
Token Verification Success
Response with Access Token
Hyperledger Auth API
Smart Contracts
Peer Nodes
Ordering Service
Blockchain Ledger

Fig. 1: Application Diagram

Four different types of transactions are presented to describe the TBAC access control procedure and provide the instances of these transactions corresponding to subject registration, object escrowing and publication, access request, and grant. The guarantee authorization relationship among these transactions is assured by Bitcoin-type cryptographic scripts.

Similar work is proposed in paper [10] where the right to access a resource can be easily transferred from a user to another through a blockchain transaction created by the last right owner, without the intervention of the resource owner; the right is initially defined by the resource owner through a transaction, and all the other transactions representing the right transfers are published on the blockchain. The advantage of this solution is that any user can inspect transactions at any time in order to check who currently holds the rights to perform a given action on a given resource. Consequently, a user who had its access request denied can check whether the entity in charge of verifying the existence of the required right actually made the right decision.

A group of researchers has done research on the Version Control System using the Ethereum Blockchain platform and IPFS which is a distributed file system [11]. Their research primarily focuses on the elimination of third-party authenticators between developers and approvers. The proposed application utilizes the main features of the Blockchain of decentralization and security. The version control system is based on the consent of approvers which involves sharing of documents between developers and approvers using the hash of the document stored on the IPFS file system. The registration of a new developer/approver is based on the consent of two-third of all registered developers and approvers. All the transactions are handled by smart contracts.

In another research for decentralized scientific communities collaboration [12], the solution for version control is proposed using public blockchain. In this application, the use of Blockchain to store commit hash is proposed. On storing the commit hash on blockchain, it will be pushed to a centralized repository management system like Git, Bitbucket, etc. The approach mainly focuses on avoiding the authorship risk due to centralized repositories. The application introduces Blockchain before the commit is pushed to a centralized repository.

## III. Application

In the proposed solution, a single point authentication for multiple applications is provided with the help of a decentralized application that is useful in achieving transparency, security, and independence from third-party applications. As shown in fig:1, when a user logs into the system, a unique token is created after successful login by Hyperledger Auth API which can be further used to access various applications. In the authentication process, the proposed solution is using Hyperledger based Auth API to authenticate the users and also check their access to various organization owned resources. Along with this, the solution also provides the provision to keep track of API calls to any application and its associated response with the help of Hyperledger Application API. Using this functionality, the activity associated with an application can be stored in the form of logs on ledger blockchain. Since data stored on the blockchain is immutable, it won't be possible for any malicious user to change the logs to hide his/her activity.

## IV. Design

### A. Overview

In the proposed system, dedicated channels are created for every Team/division within the organization. In the application, Teams are considered as "Organizations" in hyperledger fabric [5] with each Team comprising various Team Units which can be thought of as "Organizational Units (OUs)" in hyperledger fabric. Also, just like in the real world organization structure [13] where the Human Resource department hires and onboards employees in organization, similarly HRCA in the application will provide identity to various participants in the network. HRCA is a "Certification Authority" in hyperledger fabric which will be owned by the organization. The manager of a Team has administrative authority i.e. admin role in hyperledger fabric. Only the "Admin" role has permissions to add/remove users/Team members with the help of MSP (Membership Service Provider).

### B. System Architecture

*1) Components:*

- `Channel`: A Hyperledger Fabric channel is a private "subnet" of communication between two or more specific network members, to conduct private and confidential transactions. In the proposed application, each Team will be communicating over a dedicated channel. The configuration of the channel can be updated by Admin nodes.
- `Team`: Team in the proposed application behaves like an organization of hyperledger fabric.
- `Team Units`: Team Units in the proposed application behaves like organizational units of hyperledger fabric. These organizational units are part of an organization.
- `Ordering Service`: Orderer node is responsible for ordering transactions executed within the network. All orderer nodes together form an ordering service.
- `HRCA`: It is a certification authority owned by the organization used to issue identities by implementing a public key infrastructure [14] that can be used to prove identity.
- `MSP`: The MSP is the mechanism that allows that identity to be trusted and recognized by the rest of the network without ever revealing the member's private key. MSP uses certificates generated by Certificate Authorities which represent identities, to allow the user to be part of network permissioned identities.
- `Peer Node`: Peers are a fundamental element of the network which host ledgers and smart contracts. A peer can be part of an organizational unit or an organization.
- `Admin Node`: Admin node is responsible for updating channel configuration, adding or removing members in a channel.(Please refer fig: 2)

*2) Participants:*

- `OrgAdmin`: This participant will have the right to create new channels and also can give rights to other participants to do the same.
- `Team Manager`: These participants have access to admin nodes and have the right of adding or removing members in the channel.
- `Team Members`: These participants are part of Team Units and have access to peer nodes and have associated policies for access control.
- `HRs`: These participants give identity certificates to all participants of the network.

*3) Workflow:* This section explains workflow of the proposed application: The application has an interface for HRCA which will allow HRs to issue identity certificates to new employees. After an employee gets an identity certificate, a Team Manager will provide employee membership with the help of the channel MSP of his/her Team through an interface created for Team Managers. Here, each Team will have a dedicated Channel associated with it and there will be many Team Units like Developers, Testers, Database managers, etc associated with each Team. Each Team unit will have a number of peers with multiple users. The Team Manager will add users/employees in their respective Team Units. Also, there can be access control policy defined for each peer as per their access rights to company resources. In order to achieve this, the application will be using the CouchDB database to store user-specific access information. Please refer fig: 3

After a user is assigned to a Team unit, he/she can perform operations on Team owned applications. But, these operations must follow policy specified. For example, users won't be able to perform a write operation on a resource for which he/she is not allowed to, as per policy defined. These policies can be defined for all Team resources through the Team Manager interface.

In order to achieve single point authentication, the application provides Hyperledger based Auth APIs. In this mechanism, when the user first logs in to the system, a JWT token is created for the session and stored in the state database (CouchDB). This replicates Single Sign-On [15] mechanism. This token is returned to the client login interface which will then forward it to the third party application (Bitbucket, AWS Console, etc.), which will in return verify from the auth application whether the token is valid or not. This will ensure that only valid users are able to access organizational resources (in this case organization accounts) and since the tokens have an expiration time associated with it, even if a malicious user gets hold of a token, he/she won't be able to cause much damage to the integrity of the system.

The application also gives provision to store logs associated with the activity of users on company-owned applications. In order to implement this functionality, the application uses a hyperledger based chain code API to store logs in the state database.

### C. Chaincode Flow

fig: 4 shows the execution of commit requests by users in Git with the execution of chaincode. In the proposed solution, any API calls to company-owned applications will be done through the execution of chaincode. When a peer requests for commit request by calling chaincode, before executing commit operation, access of the user to perform a write operation on git is checked from state database CouchDB. The request to commit is granted only if the user has access to do the same else the request is rejected. Similarly, for any company-owned application, the same validation can be performed before executing the API call. If a user has the grant to perform API calls (here commit request), then the logs of the request are stored in the state database.

### D. Scalability

The application is built using a hyperledger fabric platform [5] that can be deployed on the cloud and local machines. However, the application can be scaled. Hyperledger Fabric does not use the Proof of Work algorithm and also the mining process is not involved. This makes the application faster and

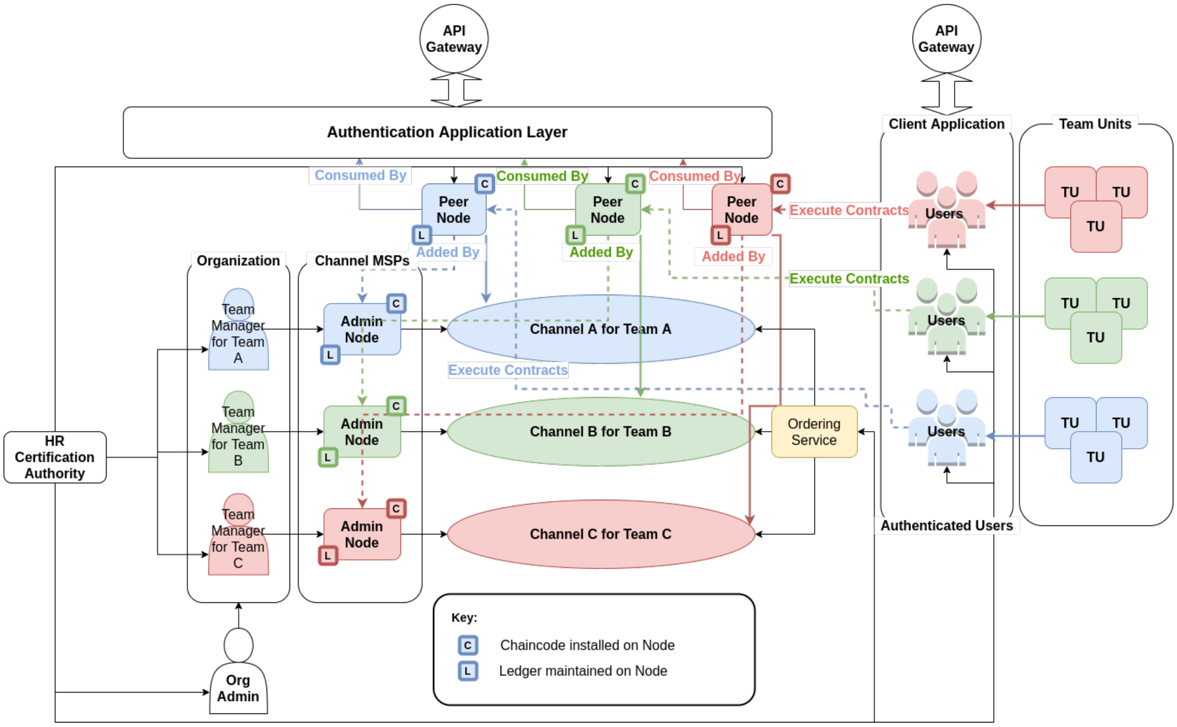

Fig. 2: System Architecture Diagram

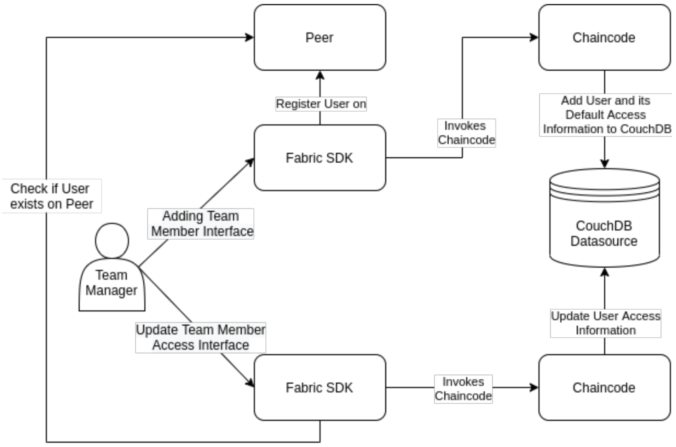
Peer
Chaincode
Register User on
Invokes Chaincode
Add User and its Default Access Information to CouchDB
Fabric SDK
Check if User exists on Peer
Adding Team Member Interface
CouchDB Datasource
Team Manager
Update Team Member Access Interface
Update User Access Information
Fabric SDK
Invokes Chaincode
Chaincode

Fig. 3: Team Manager Action Flow Diagram

more scalable [16]. There is no limit on adding the number of peers in the network.

The size of the network will be directly proportional to the size of the organization. Furthermore, we are dividing the peers to be part of separate channels, which reduces the load and network complexity of a single channel. Since for a particular peer we can have multiple users with different roles there is no need to increase numbers of peers to match the number of users. In this way, the application is highly scalable.

### *E. Hyperledger Transaction Flow*

Following are the steps involved in hyperledger transaction processing (Please refer fig: 5):

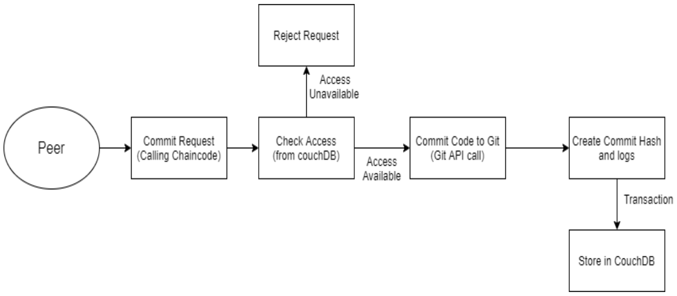
Reject Request
Access Unavailable
Peer
Commit Request (Calling Chaincode)
Check Access (from couchDB)
Access Available
Commit Code to Git (Git API call)
Create Commit Hash and logs
Transaction
Store in CouchDB

Fig. 4: Chaincode Flow Diagram

1) Client application request a transaction.
2) SDK application generates a transaction proposal and sends to endorsing peers.
3) Endorsing peers verify:
    a) The transaction proposed is well in form
    b) Not submitted before
    c) Signature is valid
    d) The client is authorized to perform the concerned operation.
4) The transaction proposal inputs as argument are passed to chaincode or smart contract and further to SDK application.
5) The application verifies the signature and compares the proposal responses to be same.
6) Application broadcast the transaction containing read/write sets and signature to ordering service.
7) Ordering peers prepares the block and transmit it to all

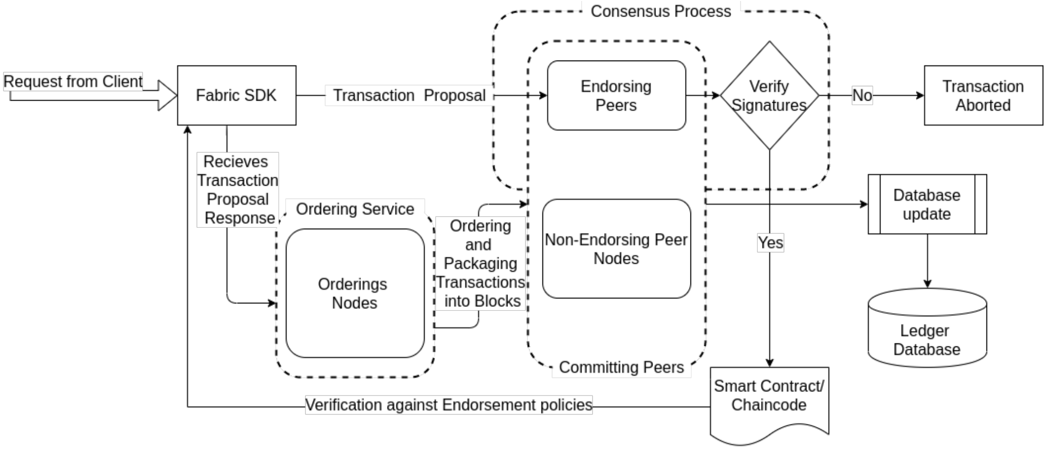

Fig. 5: Hyperledger Fabric Transaction Flow

the peers.

8) Each peer appends the block to the chain, and for each valid transaction the write sets are committed to the database.

### F. Implementation

1) Peer Adding - HRCA only issues identity certificates to a peer. But, to assign a team to the peer, the manager of a team needs to onboard the peer to the team channel. The Peer adding algorithm checks if the peer has a valid identity certificate issued by HRCA. If a new team has to be created then orgAdmin creates a new channel and adds an admin node that can govern the channel. A Team Manager can be added to this admin node who can further add other members of the team to channel. If the channel already exists then the peer is added to the channel by Team Manager. (Please refer algorithm:1)

2) Assign Access Rights - The application allows the team manager to give access rights to an individual from the team. The "Assign Access Rights" algorithm assigns default access to a user if he/she is a new user. If the user already exists then the manager can update the access rights of the user. (Please refer algorithm:2)

3) Use Access Rights - A user needs to have access rights to API else API request will be rejected. The algorithm checks if a user is authenticated and has access to the requested API call. If the user has access then the API request is processed else rejected. (Please refer algorithm:3)

4) Activity Logs Sharing - The application stores the activity logs of API calls by a user. The algorithm for this stores the successful API call response, timestamp, userInfo, output into CouchDB. In case of failed API call it stores failure status, timestamp of API call, userInfo. (Please refer algorithm:4)

**Algorithm 1** Peer Adding

```
procedure AddPeer($fabricSDK$)
    $HRCA$ gives **Identity Certificate** to $Peer$
    if new **channel** creation request then
        $OrgAdmin$ creates new channel
        $OrgAdmin$ adds new peer
        $OrgAdmin$ assign role of Team Manager to a peer
    end if
    if $peer$ request to add in new **channel** then
        $Team - Manager$ adds $peer$ to new **channel**
    end if
end procedure
```

**Algorithm 2** Assign Access Rights

```
procedure AssignAccess($user$, $access - rights$)
    if $user$ is new then
        $user.access - rights \leftarrow default.access - rights$
        $couchDB \leftarrow user.access - rights$
    end if
    if $User$ already registered then
        $user.access - rights \leftarrow access - rights$
        $couchDB \leftarrow user.access - rights$
    end if
end procedure
```

**Algorithm 3** Use Access Rights

```
procedure UseAccessRights($user$, api-request)
    $user$ requests for api-request to an application
    if $user.authenticated \wedge user.hasAccessToApiCall$ then
        api-request processed
    else
        api-request rejected
    end if
end procedure
```

**Algorithm 4** Activity Logs Storing

```
procedure StoreActivityLogs($user$, res)
    if API call succeeds then
        $OP.timeStamp \leftarrow res.timestamp$
        $OP.Body \leftarrow res.body$
        $OP.userInfo \leftarrow res.userInfo$
        $couchDB \leftarrow OP$
    else
        $OP.timeStamp \leftarrow res.timestamp$
        $OP.failureStatus \leftarrow res.failureStatus$
        $OP.userInfo \leftarrow res.userInfo$
        $couchDB \leftarrow OP$
    end if
end procedure
```

## V. Conclusion and Future Scope

Considering present scenarios and pitfalls of centralized systems for resource management, we have proposed a decentralized solution that uses the hyperledger fabric platform which is based on blockchain technology. With the help of the proposed solution, the organization will be able to manage authentication and access control of company resources in a feasible and secure way. Also, the solution provides immutable storage of activity logs of various resources which enables an organization to keep track of malicious activity which is useful for the organization to perform security audits.

However, there is definitely the scope of improvement. The application scope can be extended to open-source repositories with the help of IPFS [17] i.e. distributed file system. Third-party applications like Bitbucket, JIRA, Jenkins can be incorporated with the proposed application to store more details of employee activities in the ledger database. Also, the scope can be enhanced to include employees from vendor organizations and their access controls as per an agreement signed between the organization and the vendor. We are planning to append these features in the coming future.

## References

[1] C. Inc. (2020). What is a denial-of-service (dos) attack? [Online]. Available: https://www.cloudflare.com/.

[2] IBM. (2020). What is blockchain technology? [Online]. Available: https://www.ibm.com/.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2018.

[4] C. Saraf and S. Sabadra, "Blockchain platforms: A compendium," in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, IEEE, 2018, pp. 1–6.

[5] T. L. Foundation. (2020). A blockchain platform for the enterprise — hyperledger-fabricdocs master documentation, [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.0/.

[6] E. org. (2019). Ellcrys technical white paper (v1), [Online]. Available: https://github.com/ellcrys/papers.

[7] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[8] K. Alhamed, M. C. Silaghi, I. Hussien, and Y. Yang, "Security by decentralized certification of automatic-updates for open source software controlled by volunteers," in *Workshop on Decentralized Coordination*, 2013.

[9] Y. Zhu, Y. Qin, G. Gan, Y. Shuai, and W. C.-C. Chu, "Tbac: Transaction-based access control on blockchain for resource sharing with cryptographically decentralized authorization," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, vol. 1, 2018, pp. 535–544.

[10] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, Springer, 2017, pp. 206–220.

[11] N. Nizamuddin, K. Salah, M. A. Azad, J. Arshad, and M. Rehman, "Decentralized document version control using ethereum blockchain and ipfs," *Computers & Electrical Engineering*, vol. 76, pp. 183–197, 2019.

[12] V. Lenko, N. Kunanets, V. Pasichnyk, and Y. Shcherbyna, "Decentralized blockchain-based platform for collaboration in virtual scientific communities," *ECONTECHMOD: An International Quarterly Journal on Economics of Technology and Modelling Processes*, vol. 8, 2019.

[13] L. Learning. (2013). Common organizational structures, [Online]. Available: https://courses.lumenlearning.com/.

[14] C. Inc. (2020). How does public key encryption work? — public key cryptography and ssl, [Online]. Available: https://www.cloudflare.com/.

[15] OneLogin. (2020). How does single sign-on work, [Online]. Available: https://www.onelogin.com/learn/how-single-sign-on-works.

[16] IBM. (2020). Does hyperledger fabric perform at scale? [Online]. Available: https://www.ibm.com/.

[17] I. team. (2020). Ipfs powers the distributed web, [Online]. Available: https://ipfs.io/.