Self-sovereign identity management framework using smart contracts

Komal Gilani, Fariba Ghaffari, Emmanuel Bertin, Noel Crespi

Self-sovereign Identity Management Framework using Smart Contracts

Komal Gilani
Institut Telecom, Telecom SudParis
CNRS 5157, France
komalsyeda29@gmail.com

Fariba Ghaffari, Emmanuel Bertin
Orange Innovation
Caen, France
fariba.ghaffari, emmanuel.bertin@orange.com

Noel Crespi
Institut Telecom, Telecom SudParis
CNRS 5157, France
noel.crespi@it-sudparis.eu

***Abstract*—In centralized infrastructures, users are not capable of authenticating themselves, in identity management systems, beyond their application's domain. Users are forced to trust their service providers for identification a nd d ata management. Such solutions have experienced large-scale data breaches and are cumbersome for users to remember the credentials for multiple sites. Furthermore, users have very little control over their data. The concept of decentralized identity has raised the possibility of better managing these concerns. It allows users to share only the relevant part of their personal information with a service provider to verify their digital identity. We propose OrgID, a decentralized identity and user-centric data management platform including identity registration and authorization procedures. Our approach supports self-sovereign identity architecture leveraged by blockchain. This method consists of a one-time proof-verification m echanism t hat f acilitates t he s ecure a ccess and credibility of digital information. Moreover, users can maintain their identities associated with specific a ttributes a nd r ely o n a proof mechanism using smart contracts. It deploys a platform where user registration and authorization no longer involve a central service provider. We implemented the proposed solution using smart contracts on a private Ethereum blockchain. We further analyzed the performance of these processes regarding the system's scalability to evaluate how they manage latency and the number of users. The results state that the system is highly scalable to manage a large number of users and the system's latency is adjustable based on the application needs.**

***Index Terms*—Blockchain, Self-sovereign Identity, Single Sign-on, Authentication, Identity proofing, c laim verification.**

I. Introduction

Traditional identity management systems lack a user-centric approach that would increase user-perceived trustworthiness and reduce the ownership of user data by central authorities [1] [2] [3]. To access digital services, a user is often asked to identify himself in different services which implies a multitude of identifiers f or t he u ser t o m anage [3]. M ultiple identities prevent users from gaining control over their data.

Today, several methods such as SSO-type [4] [5] solutions are available to solve this problem. These solutions are based on the delegation of authentication to Identity Providers (IdPs). One hazard of using authentication delegation is that it can pose a threat to personal data protection [3], and the users have no control over the number and nature of information communicated to digital services by an identity provider [6] [7]. Another major drawback of this approach is their vulnerability against single point of failure due to the maintenance of all users' authentication through a single application [4].

The significant drivers for the shift to decentralized digital identity includes providing users with control over their information and optimizing workflows in business interaction to decrease operational cost. The concept of *Self-sovereign identity (SSI)* provides such a user-centric approach, in which identity is created, managed, and operated by the user. In this model, the existence of the user is independent of services [8], and the users can control the sharing of their data.

Blockchain technology which emerged as a distributed cash system since the introduction of *Bitcoin* [6] [9]. It soon became the key to the inception of smart contracts [10] [11] [12]. This technology brings many unprecedented opportunities in several platforms such as IoT [13] [12], healthcare, etc. Blockchain contains information in a chain of blocks, in which a cryptographic signature is an identifier for each block. Transparency is achieved through consensus among the nodes in a network. The database of blocks in the network forms a distributed ledger with an append-only shared record of transactions [14]. In this paper we proposed a system that enables the users to register and manage their identity.

The main contributions of this paper is to proposes a decentralized identity system, entitled OrgID, that supports identity creation and verification schemes and credential management utilizing blockchain technology. This system enables identity verification through proof computation for the authorization of entities to access services or connect to other entities. The highlighted advantages of the proposed method are as follows:

1) Users are registered, and the unique identifier is sent back to the user. Users can store their signed and encrypted personal information locally, or externally, and they can share it with other users/authorities for verification and access management.
2) It facilitates users' authorization to retrieve other users' data securely and robustly.
3) The proposed system includes a proof verification that enables challenge-based proof computing and allows users to verify their identities to access services.
4) We demonstrate our work through our prototype and provide performance analysis based on standard metrics. Lastly, we discuss future improvements on how

blockchain can become a vital resource to achieve distributed and user-centric digital identity systems.

The rest of the paper is organized as follows: a brief background on relevant technologies regarding decentralized identity is provided in section II. We present the proposed system structure, its concept, and its detailed design in section III. An evaluation of the implemented system is discussed in section IV, followed by our presentation of the key benefits and salient properties of the proposed system in section V, where we conclude our work with some future research directions.

II. Related Works

Among the most frequently mentioned blockchain-based identity and credentials management systems are ShoCard and uPort. They are commercially available SSI platforms for credential management based on Hyperledger Fabric and Ethereum blockchains, respectively. In ShoCard, user identity proofing works via existing trusted credentials stored on blockchain [15]. An intermediate agency or an intermediate provider can localize a user's information. The uPort system gathers attributes from an ecosystem of identity providers and does not provide an identity verification mechanism [16].

Zhou et al. [17] proposed an identity management solution to facilitate user registration, authorization, verification, and key recovery. Zyskind et al. [6] proposed a decentralized system for personal data management in which blockchain only stores the hash pointers to users' data and access permissions for authorization purposes. However, the attribute aggregation approach and endpoint binding for identifiers and verifiable credentials are not considered. The proposed system in [7] manages the identities of medical patients using blockchain and their encrypted records are stored in the blockchain. This approach has several weaknesses regarding design and scalability. Takemiya et al. [18] proposed an identity solution based on the JSON-LD standard. In this solution, users can generate a pair of cryptographic keys and store their private key hash locally. Encrypted keys are stored on a centralized server, which negates the decentralized concept for identity management. The proposed system in [19] provides support for user registration and storing the associated data and makes use of an Ethereum blockchain for identity and attribute storage. A similar solution is proposed in [20], with a standardized identifier and credential scheme.

Inspired by an idea proposed in [7], and [21], we propose a decentralized identity portal where users can register their digital representation and store it in the blockchain network. The identifier and key pairs are fed back to the user and must be stored in a secure place. Users can verify their identity to other parties through the proposed proof-verification mechanism. Compared to the aforementioned solutions, we have focused on identity verification and credential management as a decentralized solution.

Table I presents the comparison of the proposed system with existing solutions. Based on metrics derived from the careful analogy, we can claim that the proposed system has greater advantages on decentralized identities, using verifiable claims and providing SSO.

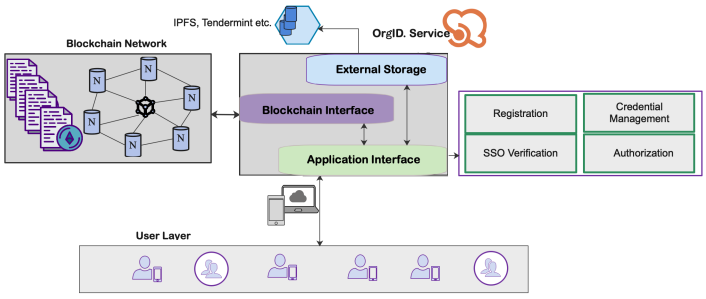


Fig. 1. High-level architecture of proposed system

III. Proposed System

We propose identity management combined with a proof-based verification mechanism for users to interact with service providers and authorities without relying on trusted delegations. In addition to provide an interface for users to create and manage their identities, this system proposes proof verification for users when they want to interact with service providers. Leveraging blockchain and smart contracts, the system aims to improve automation in the identity life cycle (i.e., registration, credential management, and verification). Blockchain can guarantee systems with no single point of failure, higher scalability, increased accountability, and reliability. Additionally, users can benefit from a variety of services and interact with other entities in a trustless environment much more robustly and securely. The novel mechanism based on the SSO concept (i.e., through combined one-time password (OTP) and blockchain) enables users to verify who they are. This method also protects the integrity of the user's associated PII.

Our proposed framework (Fig. 1) consists of users, authorities, OrgID service, blockchain network, and external storage. The private blockchain network is used to execute smart contracts. OrgID service is used for interactions between the application and network interfaces. Additionally, external storage (e.g., IPFS) can be used to store the user's digital information. It is suggested to store personal information locally or on a distributed server which allows users to search and retrieve such information. In this system, any user can register to the system and manage their digital identity representations by accessing the provided standard interface.

A. A motivating scenario

Let us consider Alice wants to use her verified identity to connect to multiple services with a single sign-on verification. Alice can register an identifier OrgID with minimum personal claims based on a pre-credential, such as her Social Security Number (SSN). A unique identifier (i.e., OrgID) is created for Alice and stored in blockchain with digitally signed encrypted personal information. Alice can share OrgID with the service A, who must check the existence of the digital personal of Alice. Upon successful verification, a hash of the computed

TABLE I
COMPARISON OF PROPOSED SOLUTION WITH EXISTING DECENTRALIZED IDENTITY METHODS

refs Features	[15]	[16]	[17]	[7]	[18]	[19]	[20]	Proposed
Decentralized Identifier	✗	✗	✗	✗	✗	✗	✓	✓
Verifiable Credentials	✗	✓	✗	✗	✗	✗	✓	✓
Credential Storage+	3	3	3	3	N/A	1	2	2,3
User-Controlled Access	✓	✓	✓	✗	✓	✓	✓	✓
Encrypted Information	✓	✓	✓	✓	✓	✗	✓	✓
Verification Scheme**	✗	✗	✗	✗	✗	✗	✗	✓

* Identifier: $Acct_{bc}$ (Using blockchain address as Identifier or combination) or DID (W3C Standard DID identifier)
+ (1) On-Chain (Store encrypted information on blockchain), (2) Locally (Store encrypted information on device of user's choice) and (3) Externally (Store encrypted information on another server e.g., IPFS)
** Allows the user to verify their identity to other users and authorities using SSO proof through smart contract and oracle

oracle proof is received by Alice, who then publishes it on the blockchain. With the oracle proof service A can match proofs and certify successful verification by adding the hash and the signature for Alice's identity in the blockchain. When Alice wants to connect to service B, she can verify the signature from service A to confirm that her identity has been verified.

With Business-to-Business (B2B) services, accessing client's information can be accomplished by relying on identity verification based on identifiers and verifiable credentials. For instance, medical company A may subscribe to a research company, B, on behalf of their employees. In this case, employees of company B do not have a direct connection with company A and thus need to verify their relationship with them.

B. Blockchain interface and smart contracts structure

In the proposed system, the blockchain interface consists of different smart contracts to provide digital identity verification and management. It mainly includes Identity Registration Contract (SC_{IRC}), Address Management Base Contract (SC_{AMBC}), Credential Storage Contract (SC_{CSC}), Service Provider Base Contract (SC_{SPBC}), and Verifier Contract (SC_{ver}). For each registered user, a profile is generated in blockchain through the Org interface, which is used as a unique identifier (OrgID) and fed back to the user along with a key pair. A secret key or secondary identifier is derived from $OgrID$ and can be used to retrieve information of identity and associated credentials. This identifier can also be used as a backup key in case of a temporary loss of the $OrgID$.

SC_{AMBC} holds the information of all the registered users and deploys several functions associated with digital identity. These functions act through contracts to fulfil the authentication of digital identities and their related information and to maintain the operational log. Each key operation performed on an identity-basis, such as access, sharing, and interaction with other entities for trustees is recorded in an SC_{WEC}. These logs are temper-proof in the blockchain, which makes it easy to monitor users' operations and to track the actions performed on digital identity. For verification, the proposed SSO mechanism code in the SC_{SSO} is also included. SC_{CMC} holds the logs of encrypted records, which represent the relationship between the digital fingerprint of the encrypted information obtained from an external source and where the corresponding sensitive information is stored locally. The structure of these records is defined in SC_{CSC}, where tuples for credentials with hashed and associated $OrgID$ are generated. SC_{SPBC} is used to demonstrate the access request based on a proposed verification in which the user can send their request to the service provider base and the service provider creates the "service code" as a challenge. Oracle generates the proof using SC_{SSO}, and then sends it to the user and stores the log on the blockchain when the user submits the proof associated with a given "service code". The service provider can verify the proof and grant user access. Our blockchain interface contains several smart contracts, enabling it to support a variety of digital identity requirements such as registration and login, information storage, verification, etc. Descriptions of these main smart contracts are listed below:

1) SC_{IRC} *contract:* This smart contract is the first step in a user's registration and their creation of an identifier in the system. The blockchain system contains several of these contracts for each user. It takes the following inputs: 1)A prefix string to represent the identifier's nature; 2) The user's account address; 3) A Boolean value to represent the user's identity (i.e., active/inactive) mode; 4) A hash value; and 5) A URI that contains a link to an external representation. It generates the tuple of decentralized identifiers, called OrgID, and stores it as the user's representation into a distributed database contract called SC_{AMBC} (this contract is discussed next. This decentralized identifier OrgID is designed in compliance with the W3C decentralized identity standard [22].
2) SC_{ABMC} *Contract:* This smart contract is a global identity contract that acts as a storage for the aforementioned contract and invokes the Creating, Reading, Updating, and Deleting (CRUD) operations on identity. It also encompasses the interaction with SC_{CMC} and SC_{IRC} contracts. These smart contracts provide an identity management base for users to perform a variety of operations on identity, including registration, update (active/inactive status), read, and transfer of ownership.
3) SC_{CS} *Contract:* This smart contract is designed to

provide users a means to store and invoke credentials. Users can store their digital information in the SC_{CSC} and invoke the CRUD operation on credentials such as Upsert, Revoke and Update SC_{ABMC}. The credential structure holds various attributes such as status, associated identifier, owner's account details, hash value(s), URI, and type of credential (e.g., education record, medical record). Furthermore, these records are associated with the identifier's OrgID and derived secret key (key_{id}). Noted that these records are stored in the ledger in the form of digital prints, and any sensitive information associated with them is to be stored locally. Users can share these records with the issuer(s) (through external means), who can provide their attestation(s) by upgrading the credential verification status in the form of duplicate records in the network.

4) SC_{SOS} *Contract:* This smart contract provides the verification mechanism with which a user can compute a challenge in the interface. This contract lets the service providers verify that an identity belongs to the same user. Using the OTP-based scheme, Oracle generates the proof and stores it in blockchain with access permission, and the user receives the proof through an interface such as SMS OTP. Firstly, the user submits the required proof in the blockchain, and then the oracle proof is recorded. This mechanism helps to avoid security threats such as spoofing and phishing attacks [23]. Upon proof submission, the service provider can verify the proof and grant/deny the user access accordingly.
5) SC_{ICC} *Contract:* This contract manages all the operations and access logs (e.g., maintains the list of permissions and the footprints of a user's activities). The authorization-related operation, defined for users, is stored in the Access Log Contract (SC_{ALC}).
6) SC_{SPBC} *Contract:* This smart contract contains the information about the various service providers that users are connected to. It includes a structure that encapsulates each service provider's address, URI, and "access_token" parameters. Individual service providers can use SC_{SPBC} to verify each proof and entertain different requests submitted by users.

C. System design

Here we present the detail overflow of our proposed identity management operation, including identifier registration, credential management and a verification scheme.

1) Identifier registration: The first step is to register the user ($acctU$) into the blockchain. This interface enables the interaction between the user and the blockchain by generating a public/private key pair (sk_U, pk_U) and corresponding blockchain address ($addr_U$). Key pairs are used to sign and encrypt personal information and claims, which are stored externally, while digital fingerprints are stored on the blockchain. These fingerprint logs are linked to $addr_U$ and real digital information. The process is detailed below (Fig. 2):

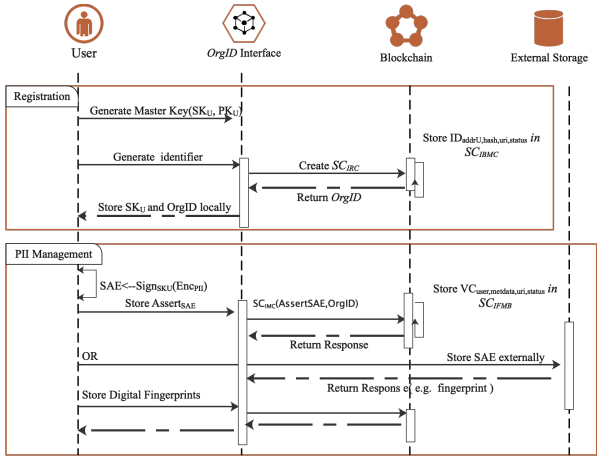


Fig. 2. The process of identifier registration and information management

- *Step 1:* A user generates the public-private key pair (i.e., sk_U, pk_U) and associated ($addr_U$) using the interface. The private key is stored locally on a device chosen by the user.
- *Step 2:* The user creates their unique identifier by creating a SC_{IRC}, which takes four parameters: the user's address (from step 1), a hash value, a URI, and a status value to represent the active/inactive status of the identifier. An OrgID and an internal key are generated from $addr_U$, and an identifier is created as follows:

$$OrgVC : Cred_{VcKey} < acctU_{subject}, acctU_{Issue}, Hash_{data}, URI >$$
$$OrgID = ID < addr_U, Key, URI, Hash_{doc} >$$

The identifier is stored in SC_{IBC}, where all the identities on the blockchain are stored. The OrgID is fed back to the user as an identifier in the proposed system.
- *Step 3:* After the successful registration of their identifier, the user can access their identifier based on the generated identifier and execute a variety of operations and to use it to extend their interaction with other entities, including other users, as well as to access different services such as those offered by education institutes, medical facilities, IoT manufacturers, etc.

2) Issuing credentials: In our proposed system, we provide a means to store identity-associated information in a blockchain system. In order to securely exchange credentials, the system allows each user to store the records of credentials on the network. However, with size limits and privacy as the main concerns, the representation of these claims does not include any likability to the actual information. Instead, these credentials are stored as hash values and the user can select a source location URI where accessibility with minimum disclosure is given. The process involves the following steps:

- *Step1:* Once the user is registered in the system and receives the decentralized identifier, she can input her PII and store the encrypted and signed digital information using a (sk_U, pk_U) pair.
- *Step 2:* The encrypted information is stored as a tuple

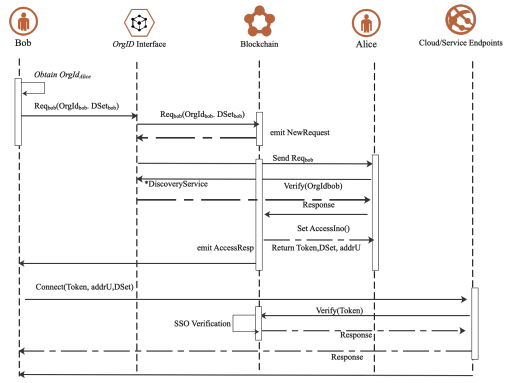


Fig. 3. Processes of Authorization. demonstration of the scenario where Bob sends the request to access Alice's information

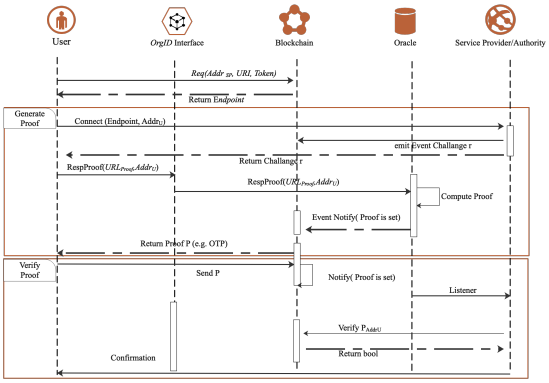


Fig. 4. Processes of SSO verification

that includes the above-mentioned identifier, the hash of metadata, and URI for the link to public information available outside of the blockchain. It is recorded on the blockchain through SC_{CSC} smart contract. The records are stored in the following form:

- *Step 3:* The user can share these credentials with authorities as a means of identity verification. Users can store their credentials by themselves, known as self-asserted claims, and use them to collect verification badges from authorities that can verify the status of these assertions.
- *Step 4:* To revoke credentials, subject or issuer calls the revoke function of $revocation - registry$ through smart contract SC_{RR}. Only the subject or issuer, following identity verification, can revoke relevant credentials.

3) Authorization (selective disclosure and verification): A user-centric authorization mechanism allows users to authorize and access other users' information. Users can set the permissions for whitelist entities in SC_{WEC}. As shown in Fig. 3, the authorization process can include the following steps:

- *Step 1:* Bob (i.e., the user to be authorized) obtains the identifier of another user (Alice). This interaction can be initiated through common services (e.g., connected social network). Bob requests to access Alice's information. In that request, Bob also shares his identifier as proof of his identity along with dataset in the following request
$$Req_{Bob} < OrgId_{Bob}, DSet_{Bob} >$$
- *Step 2:* Upon detecting the new access request (e.g., in mobile application), Alice can verify the identity of user by realizing the $OrgID - Bob$ information encapsulated in the identifier through available discovery service.
- *Step 3(a):* In case of successful verification of Bob's identity, Alice invoke the smart contract SC_{ICC} and SC_{WEC} to add him in the white-list entities. For data access endpoint, the token is generated using $OrgID_{Alice}$ and $OrgID_{Bob}$ and signed with Alice's PK. The token along with following information and a set of policies on requested data is submitted.
$$< token, DSet_{Bob}, Status_{bob} >$$
- *Step 3(b):* Alice can choose to block the permission and invoke the SC_{ICC} to deregister Bob from the white-list (if it exists already) with following information:
$< Sig_{SK}(OrgID_{Alice}, OrgIDbob, DSet, Status_{Bob} >$; where $Status_{Bob}$ for specified request is set as blocked.
- *Step 4:* Successful execution of operations in Step 3 (in both cases) notifies the response which grants/denies the access and emits the request/response logs on blockchain network.
- *Step 5:* Bob receives the response and retrieve the token from emitted event to download the data from endpoint. To perform the token validation, SSO verification is operated (this part is explained in the next step).

4) SSO-based verification mechanism: Our proposed decentralized identity verification method allows users to authorize themselves to Service Providers (SPs) using smart contract and oracle capabilities. This SSO-based verification mechanism, depicted in Fig. 4, involves the following steps:

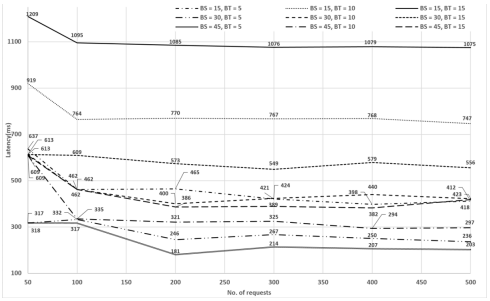
- *Step 1:* Bob sends the following request to SP by invoking the SC_{SPBC} and retrieves the endpoints to communicate with desired SP to access either service or an endpoint to retrieve Alice's data as discussed earlier.
$$Req_U < Addr_{SP}, URI, Token >$$
Upon receiving the request, SP sends the challenge code and endpoint to user and emit the request data along with challenge information on blockchain.
- *Step 2:* User invokes the SC_{SOS} contract with proof request and submits the $URL_{Proof}, Addr_U$. SC_{SOS} notifies the oracle where proof is computed and sends back to server (through secure endpoint) and emit the "proof is generated" log on the network.
- *Step 3:* User receives the notification and submits the proof (access permission applied) using SC_{SOS} and emits the "User proof is generated" log on the network.
- *Step 4:* After successful submission of user proof, the oracle proof is submitted using SC_{SOS} and emits the "Proof Added" log on the network.
- *Step 5:* After, the successful proof computation, SP can verify and emit the response on the network such as "access granted/denied", accordingly.

IV. Performance analysis

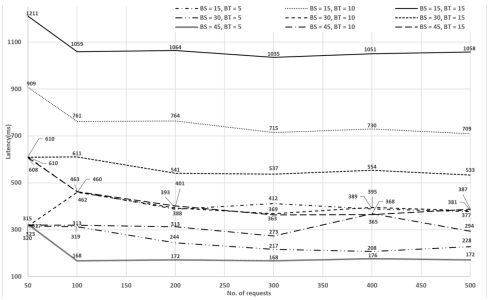
This section describes the implementation and evaluation of our proposed framework, called OrgID Services. Even though the comprehensive implementation of the system is still under development, we have implemented the blockchain interface where user's identity and credential records are stored in the private Ethereum blockchain. We developed the above-mentioned smart contracts in Solidity and deployed them in a private Ethereum Blockchain simulated by Ganache-cli v.6.12.2 and compiled by Solc v.0.8.2.

The performance analysis of the proposed method was done by evaluating the scalability of the system in terms of the increasing number of concurrent connection requests and different parameters of blockchain configuration. To measure the scalability, we assess the latency [24], [25] which establishes the average time required to handle one transaction. The scalability of the system is the changes of latency by altering a parameter [26]. We used three parameters to measure the latency deviation: $BlockTime(BT)$, $BlockSize(BS)$, and the number of $concurrent\ requests$ sent to the blockchain. BT defines the difficulty of the consensus puzzle which leads to the extraction of blocks in predefined time. In our experiments we selected $BT = [5, 10, 15]$. Due to the limitation of web3j library, we could set $BS = [15, 30, and45]$. Concurrent requests (C) are the number of requests that are sent concurrently to the system (by virtual clients). We set this parameter as $C = [50, 100, 150, 200, 250, 300, 350, 400, 450, 500]$.

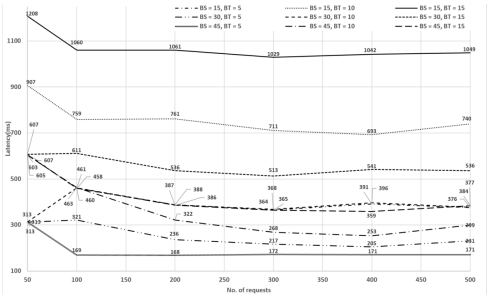
Figure 5 (a-c) depicts the latency of the identifier registration, credential issuance, and verification for different values of BS and BT, respectively. As shown in this figure, for $C \geq 100$ the average time deviation is less than $50ms$. It means, the system is not highly sensitive to the number of requests. Therefore, we can state that the system is scalable and can maintain stable latency in a large-scale environment. Another result of the experiments is that by increasing the BS and decreasing the BT to a threshold of the system requirement, the latency decreases. In the other words, the system's latency is adjustable based on the requirements of the application. It is important to mention that the higher level of security, the higher latency. It means, to achieve a more secure system, the difficulty of the consensus puzzle must be increased.



(a)



(b)



(c)

Fig. 5. System latency with different values for BT, BS, and concurrent requests for (a) identifier registration, (b) credential issuance, (c) verification.

V. Discussion and future work

We present a decentralized identity and access management method using blockchain and smart contracts. The proposed system provides the user-controlled and SSI lifecycle by enabling the global identity representations on the blockchain and exchange of information and data with other entities in the ecosystem. The main goal of the proposed system is to provide a flexible identity management solution that is sustainable and interoperable in the broader sense of adoption. The highlighted features of the system are described as follows:

- *Identity Life Cycle:* We have achieved user registration, information management, authorization, and SSO access with the help of smart contracts. The system takes into account the data size limit and defines the structure for identity and credentials to be recorded without revealing sensitive information.
- *Standardisation and Interoperability:* The defined structure of identity and information management is compliant with decentralized identity (DIDs) and thus extends the enhanced privacy by design. DIDs and verifiable claims (VCs) [8] support the global representation of identities and ensure the interoperability.
- *User-Centric Access Control:* The system allows the user to define access permissions by adding the permitted entities into a whitelist and grant access to data using access tokens.

Indeed, we aim to improve the system in future work by investigating promising means for key recovery and discovery services. Another direction will utilize a system comprised of smart contracts, APIs, and a prototype application interface to deploy this identity system to real-world use-cases.

REFERENCES

[1] T. El Maliki and J.-M. Seigneur, "A survey of user-centric identity management technologies," in *The International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007)*. IEEE, 2007, pp. 12–17.

[2] J. Torres, M. Nogueira, and G. Pujolle, "A survey on identity management for the future network," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 787–802, 2012.

[3] O. Jacobovitz, "Blockchain for identity management," *The Lynne and William Frankel Center for Computer Science Department of Computer Science. Ben-Gurion University, Beer Sheva*, 2016. [Online]. Available: https://www.cs.bgu.ac.il/ frankel/TechnicalReports/2016/16-02.pdf

[4] E. Maler and D. Reed, "The venn of identity: Options and issues in federated identity management," *IEEE security & privacy*, vol. 6, no. 2, pp. 16–23, 2008.

[5] V. Radha and D. H. Reddy, "A survey on single sign-on techniques," *Procedia Technology*, vol. 4, pp. 134–139, 2012, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017312002988

[6] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*. IEEE, 2015, pp. 180–184.

[7] T. Mikula and R. H. Jacobsen, "Identity and access management with blockchain in electronic healthcare records," in *2018 21st Euromicro conference on digital system design (DSD)*. IEEE, 2018, pp. 699–706.

[8] "Verifiable Claims Working Group Frequently Asked Questions." [Online]. Available: http://w3c.github.io/webpayments-ig/VCTF/charter/faq.html

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[10] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[11] N. Szabo, "Formalizing and securing relationships on public networks," *First monday*, 1997.

[12] L. Ismail and H. Materwala, "A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions," *Symmetry*, vol. 11, no. 10, p. 1198, 2019.

[13] O. Alfandi, S. Otoum, and Y. Jararweh, "Blockchain solution for iot-based critical infrastructures: Byzantine fault tolerance," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–4.

[14] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[15] "ShoCard." [Online]. Available: https://www.shocard.com/en.html

[16] D. C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena, "UPORT: A PLATFORM FOR SELF-SOVEREIGN IDENTITY," p. 17.

[17] T. Zhou, X. Li, and H. Zhao, "Everssdi: blockchain-based framework for verification, authorisation and recovery of self-sovereign identity using smart contracts," *International Journal of Computer Applications in Technology*, vol. 60, no. 3, pp. 281–295, 2019.

[18] M. Takemiya and B. Vanieiev, "Sora identity: Secure, digital identity on the blockchain," in *2018 ieee 42nd annual computer software and applications conference (compsac)*, vol. 2. IEEE, 2018, pp. 582–587.

[19] J. Alsayed Kassem, S. Sayeed, H. Marco-Gisbert, Z. Pervez, and K. Dahal, "Dns-idm: A blockchain identity management system to secure personal data sharing in a network," *Applied Sciences*, vol. 9, no. 15, p. 2953, 2019.

[20] "Seraph ID: Introducing Swisscom Blockchain's digital identity solution on NEO," Jul. 2019. [Online]. Available: https://neonewstoday.com/general/seraph-id-introducing-swisscom-blockchains-digital-identity-solution-on-neo/

[21] X. Fan, Q. Chai, L. Xu, and D. Guo, "Diam-iot: A decentralized identity and access management framework for internet of things," in *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 2020, pp. 186–191.

[22] "Decentralized Identifiers (DIDs) v1.0." [Online]. Available: https://www.w3.org/TR/did-core/

[23] G. B. Ayed and S. Ghernaouti-Hélie, "Privacy requirements specification for digital identity management systems implementation: towards a digital society of privacy," in *2011 International Conference for Internet Technology and Secured Transactions*. IEEE, 2011, pp. 602–607.

[24] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey on the scalability of blockchain systems," *IEEE Network*, vol. 33, no. 5, pp. 166–173, 2019.

[25] P. W. Eklund and R. Beck, "Factors that impact blockchain scalability," in *Proceedings of the 11th international conference on management of digital ecosystems*, 2019, pp. 126–133.

[26] M. Schäffer, M. Di Angelo, and G. Salzer, "Performance and scalability of private ethereum blockchains," in *International Conference on Business Process Management*. Springer, 2019, pp. 103–118.