A Secure and Privacy-Preserving Student Credential Verification System Using Blockchain Technology

Jayana Kaneriya* and Hiren Patel

Abstract—**Advancements in digital technologies have made the storage, sharing, and verification of educational credentials extremely important for entities such as students, universities, institutions, and companies. Digital credentials play an important role in students' lives as a lifelong learning passport. The educational field is experiencing numerous issues such as academic record forgery, record misuse, credential data tampering, time-consuming verification procedures, and issues related to ownership and control. Modern-day technology, Blockchain, is an appropriate alternative to resolve these issues and increase trust among entities. In this research, we intend to propose a Blockchain-based educational digital credential issuance, and verification model that addresses these issues in the education system using Ethereum Blockchain and smart contracts. The method we propose offers a way to demonstrate the correctness of specific credential attributes without revealing other attributes, thereby leading to ownership, minimal disclosure, and control. We offer an interface for storing massively encrypted academic records in a decentralized file system like Interplanetary File System (IPFS). Furthermore, Ethereum provides tamper-resistant chains to maintain the integrity of digital credentials. Finally, in comparison with the time it requires to issue credentials, our model safely accelerates the verification process by about 8%.**

Index Terms—**Blockchain, Ethereum, interplanetary file system, smart contract, digital credential**

I. Introduction

The potential growth in smart cities is due to advancements in technologies such as Internet of Things, Machine Learning, Cloud Computing, Data Science, Virtual Reality, Big Data Analytics, Artificial Intelligence, Neural Networks, Deep Learning, and others. These cutting- edge technologies have influenced most daily activities because of their obvious advantages like quick response, ease of use, and extensive availability. However, online sharing and verification of educational documents cannot always guarantee security, privacy and integrity in secured networks.

Recent developments in Blockchain introduce a decentralized way to achieve security, transparency and integrity for numerous application domains. Some of the latest domains where Blockchain has proven its importance are real time supply chain for project deliveries [1], secured data harvesting for agriculture [2], an electronic healthcare record system [3], a transport monitoring system [4], smart cities [5], a fog computing platform [6], a land registry system [7], and Smart Grid Systems [8]. Moreover, education is the backbone of society in which various stakeholders interact with each other in trustless environment.

Existing educational systems do not offer complete control and independence of credentials such as mark sheets, degree certificates, letters of recommendation, training certificates, and character certificate to the real identity owner, i.e., students. These credentials are building blocks for students' career and selection criteria for recruiters. So, smooth and secure sharing of such credentials with recruiters and other institutions is required to showcase the ability of a student. A similar level of complexity arises for a verifier to validate the integrity and authenticity of a credential submitted by a student [9].

Cryptographic techniques used in Blockchain enhance the security and integrity of transactions recorded by a distributed ledger. Blockchain solves the problem of lack of trust by maintaining transaction records to each participating node. Transactions are recorded in a block which is added by a miner using a consensus algorithm. In addition, the Merkle tree generates a cryptographic fingerprint of the entire set of transactions for a block to ensure its integrity and inclusion. The chain is created by storing the cryptographic fingerprint of the previous block. Blockchain is immune to attacks because the entire ledger is chained, and altering one transaction requires, subsequent blocks to be altered, which is nearly impossible [10]. Therefore, Blockchain can effectively solve existing problems such as the issuance, maintenance, integrity, privacy, and authenticity of credentials in the education domain.

In a recent work, we proposed a general framework for privacy preserving self sovereign identity with selective disclosure using Blockchain [11]. In this paper, we extend and refine our initial work in multiple dimensions by adding the following novelties:

- Introducing the usage of a general framework for the education domain with inter-contract communication and contract-service interaction.
- Facilitating the process of individual or group of elements' verification without disclosing other attributes.
- Due to the random placement of attributes in the tree, two different entities having the same number of attributes will have a completely different tree structure. This results in unpredictability for an attacker, making the estimation much more complex.
- Due to the inherent nature of Blockchain ttechnology, transactions once committed or mined shall never be altered which leads to the enhancement of security and transparency among legitimate stakeholders.

Manuscript received April 13, 2023; revised May 4, 2023; accepted May 18, 2023.

Jayana Kaneriya is with Kadi Sarva Vishwavidyalaya, Sarva Vidyalaya Kelavani Mandal, LDRP Institute of Technology & Research, Gandhinagar, Gujarat, India.

Hiren Patel is with Kadi Sarva Vishwavidyalaya, Sarva Vidyalaya Kelavani Mandal, Vidush Somani Institute of Technology and Research, Kadi, Gujarat, India.

*Correspondence: jayana102006@gmail.com (J.K.)

- Avoiding human or manual intervention in the process of smart contract execution results in increased trust and a reduction in possible malicious activities.

The rest of the paper is structured as follows. In Section II, there is a description of the literature survey. Section III describes the design and details about smart contracts and core services. In Section IV, we present the work flow of the scheme with its main procedures. Section V, contains the details of the experimental results and the security analysis of the proposed solution. Finally, in Section VI, we draw conclusions and outline future work.

II. Literature Survey

The digitization of the education domain drives economic growth, innovation, and the development of society. This section elaborates on various Blockchain-based educational models to ensure decentralization, immutability, integrity, privacy, and security.

Arenas and Fernandez [12] introduce CredenceLedger, a permissioned Blockchain-based mobile application for digital credentials of students. Arndt and Guercio [13] propose transcript storage and verification using a graph -based database, Neo4j, which utilizes the Cypher query language to search transaction details from the Blockchain. In addition, model uses BigChainDB to securely store and retrieve transcript records. Bessa *et al.* [14] present a Hyperledger Composer Blockchain-based repository architecture for educational credentials. Han *et al.* [15] discuss an Ethereum and smart contract-based framework to store academic records for students. Furthermore, the work implies that cost-concerned applications are the major area, as cost is reduced in Blockchain-based applications compared to cloud-based applications. Srivastava *et al.* [16] introduce a token-based credit system for educational institutions to transfer credit to students. The system utilizes Ark Blockchain to maintain a chain for credentials generated by a multi signature scheme.

Oganda *et al.* [17] present a business model with components to integrate Blockchain technology and educational institutions for online education. Lizcano *et al.* [18] propose a Blockchain-based approach to create a trust model between students and employers for reliable credential verification. Guo *et al.* [19] discuss a hybrid model that uses a private Blockchain to store multimedia educational resources and a public Blockchain to store digital certificates. Third party organizations use public Blockchain to verify the authenticity of a digital certificate.

Turkanovic *et al.* [20] present and implement a prototype, EduCTX, which is built upon the Ark Blockchain platform to share educational credit of students with different institutions. Ark provides a flexible way to select the Blockchain type as permissionless or consortium. The prototype uses the Delegated Proof-of-Stake consensus algorithm [21] so that new nodes have to confirm their identity to join a network. The authors describe basic scenarios such as (a) procedural steps for a new node to join an existing network, (b) the process for student enrolment by an institute, (c) issuance of credit to enrolled students, and (d) credit record verification. Students use a wallet to store credits in the form of ECTX [20] tokens although communication is done over a private channel which can be prone to attack.

EduRSS is proposed by Li and Han [22], which focuses on secure storage and sharing of educational records using Blockchain and an encryption scheme. The scheme provides smart contract-based access control for records stored on the storage server and Blockchain with security analysis. The server keeps records in encrypted form, and the Blockchain maintains a chain record hash. However, they do not discuss retrieval methods for encrypted records and usage of decentralized off-chain data storage.

A heterogeneous Blockchain-based framework is discussed for lower latency and higher throughput using private and consortium Blockchain [23]. Private Blockchain is deployed on each educational campus to store students' private data, and consortium Blockchain stores the hash of that data, thus it is used by recruiters and other institutions to verify students' data. They analyze the operating cost for centralized, decentralized and heterogeneous systems and compare storage space for consortium Blockchain and heterogeneous Blockchain. However, they do not mention smart contract structures and security analysis.

A Blockchain-based crowd sourcing platform for underprivileged students is proposed with interactions between the preliminary stakeholders of the system i.e., students, fundraisers, and sponsors [24]. Students submit an application form for scholarships, loans, or donations along with academic and personal information through a web interface. Fundraisers and sponsors are able to view the list of students. The platform provides flexibility to sponsors for full or partial sponsorship and the role of the fundraiser is to commit a transaction. The authors propose architectural details, the workflow of the system, and prototype implementation using Java. Although, they do not discuss smart contract latency and deployment cost.

A transparent and reliable academic credit issuance and verification platform is proposed for the Brazilian educational system to increase security and decrease bureaucracy [25]. An educational certificate verification model is presented to ensure privacy, authorization, and ownership using the Hyperledger Fabric Framework [26].

To address issues in higher educational systems in low-income countries, a novel blockchain-based solution is introduced with security, immutability, and transparency [27]. The authors discuss a comparative study of a non-smart contract-based solution with a smart contract-based solution. Additionally, they mention implementation challenges and cost feasibility to run the prototype model.

Ali *et al.* [28] discuss three models for student information systems to maintain transactions on the blockchain with higher data availability. The different models use stateful and stateless data to match the requirements of small-scale organizations to large-scale organizations. The student information system is able to provide security, reliability, and trust among active stakeholders. The authors do not provide smart contract implementation details.

Mishra *et al.* [29] propose a tamper-proof, non-repudiable, and privacy-protected smart solution for secure sharing of educational credentials. They design nine smart contracts to automate data handling in a decentralized application. The solution is tested and validated using the public blockchain, Ethereum, with security and cost analysis.

III. Proposed Work

The overview of the proposed framework was shown in our earlier paper [11], and this work is an extension of that by applying a general framework to a specific use case i.e., education. In this section we describe the smart contracts, communication between smart contracts, and communication between smart contracts with core services of our proposed framework.

A. Smart Contract Introduction

Enrolment Contract (EMC): The contract is used to enroll various entities, such as (a) Issuer (university), (b) Owner (students), and (c) Verifier (company). The process of enrolment requires (i) entity address, (ii) entity's public key, (iii) class (issuer / owner / verifier), (iv) link to the private key, and (v) time stamp for the enrolment.

Algorithm 1 outlines the process to enroll the issuer, owner, and verifier to the system. The steps include: (i) checking if the caller of this function has the authority to call the function, (ii) adding related data of an entity to an Entity structure, (iii) adding entity address to entityChain array, (iv) appends a list based on class (issuer, owner, verifier) of an entity, and (v) finally, enrolling of an entity will trigger the event LogEntityEnrol () to notify the system about the completion status.

Algorithm 1: Algorithm to add entity to the system

```
Inputs:    _address, _publicKey, _class, _privateKeyLink

Outputs:   LogEntityEnrol ()
1   if     msg.sender! = owner then
2              exit
3   else
4              Entity [_address].entityAddress ← _address;
5              Entity [_address].proposerAddress ← msg.sender;
6              Entity [_address].publicKey ← _ publicKey;
7              Entity [_address].timeStamp ← now;
8              Entity [_address].class ← _class;
9              Entity [_address].privateKeyLink ←_privateKeyLink;
10             entityChain.push(_address);
11             entityListByClass[_class].push(_address);
12             Emit the event LogEntityEnrol
               (msg.sender,_address,_publicKey,_class)
13  end
```

Credential Issuance Contract (CIC): The contract is used to maintain a credential chain for students. The issuance process requires (i) owner address, (ii) issuer address, (iii) credential ID, (iv) link to the credential, (v) credential attributes, (vi) credential hash, and (vii) status. The issuance process uses the method proposed in our earlier paper [11]. The Algorithm 2 outlines the process to issue a credential to the owner.

The steps for issuing a credential are as follows: (i) The CIC interacts with the EMC to validate the caller of the function (who must be an issuer); (ii) The function assigns the related values to a Credential structure; (iii) To provide randomization in tree construction, the function calculates the mole value of each attribute using the ASCII value of each character. Although, different equations have been used to calculate the mole value of each attribute to ensure randomization of attributes, as described in our earlier paper [11]; (iv) The mole values are rearranged using a sorting algorithm. Quick sort has been used to arrange the mole values in ascending order. However, we offer to choose any sorting algorithm like bubble sort, insertion sort, heap sort, selection sort; (v) The function rearranges the attribute values by changing the position of the attributes; (vi) The function creates a hash tree based on the updated attribute values; (vii) The function appends the owner and issuer lists by address as a key and value as a credentialID; (viii) Finally the issuance of a credential triggers the event LogCredentialIssuance () to notify the system about the completion status.

Algorithm 2: Algorithm to add credential details

```
Inputs:    _ownerAddress, _credentialID, _credentialLink,
           _credential [], _credentialHash, _status
Outputs:   LogCredentialIssuance ()

1   if     entityEnrolment.getEntityClass(msg.sender) != issuer
2              Exit
3   else
4              Credential [_credentialID].ownerAddress ←
               _owneraddress;
5              Credential [_credentialID].issuerAddress ← msg.sender;

6              Credential [_credentialID].credentialID ← _credentialID;
7              Credential [_credentialID].credentialLink ←
               _credentialLink;
8              Credential [_credentialID].credentialHash ←
               _credentialHash;
9              Credential [_credentialID]. status ←_status;
10             _mole ← calculatemole(_credential)
11             _sortedMole ← sort (_mole);
12             updatedCredentialList ← Transpose (_sortedMole,
               _mole, _credential);
13             Credential[_credentialID].hashTree ←
               MerkleTree(updatedCredentialList);
14             credentialListByOwner
               [_owneraddress].push(_credentialID);
15             credentialListByIssuer [msg.sender].push(_credentialID);
16             Emit the event LogCredentialIssuance(_ownerAddress,
               msg.sender, _credentialID, _credentialHash, _status)
17  end
```

Consent Contract (COC)*:* This contract is responsible for maintaining a chain for students who want to provide consent for their credential to a legitimate verifier. This process requires (i) consent ID, (ii) owner address, (iii) verifier address, (iv) credential ID, (v) attribute index, (vi) validity, and (vii) random token.

Algorithm 3 outlines the process to add a transaction related to providing consent to the verifier to verify a particular attribute value from a credential. The steps include: (i) The COC interacts with the EMC to validate the verifier and the COC interacts with the CIC to verify the caller of this function (which must be an the owner of a credential); (ii) The function adds the related data of a consent to a Consent structure; (iii) The function generates a proof using the CIC for a particular attribute value to a verifier students may only want to share their result (pass/fail) with a company, as explained in Algorithm 4; (iv) The function appends the owner list and verifier list by address as a key and value as a consentlID; (v) At the end, LogConsent () event notifies the system regarding the completion status.

Algorithm 3: Algorithm to add consent

```
Inputs:    _verifierAddress, _consentID, _credentialID, _ index,
           _token, _validity
Outputs:   LogConsent()
1   If     entityEnrolment.getEntityClass (_verifierAddress) != verifier && msg.sender!=credentialIssuance.
           getOwnerAddress (_credentialID) then
2          exit
3   else
4          Consent [_consentID].ownerAddress ← msg.sender;
5          Consent [_consentID].verifierAddress ← _verifierAddress;
6          Consent [_consentID].consentID ← _consentID;
7          Consent [_consentID].credentialID ← _credentialID
8          Consent [_consentID].token ← _token;
9          Consent [_consentID].validity ← now + (_validity * 1 days);
10         _proof ← proofGeneration(_credentialID,_index);
11         Consent [_consentID].credentialproof ← _proof;
12         consentLlistByOwner[msg.sender].push(_consentID);
13         consentLlistByVerifier[_verifierAddress].push(_consentID);
14         Emit the event LogConsent(_verifierAddress, msg.sender, _consentID, _credentialID, _proof)
15  end
```

Algorithm 4: Algorithm to generate proof

```
Inputs:    _credentialID, _index
Outputs:   _proof
1    ....
2    _hashTree ← credentialIssuance.getHashTree (_credentialID);
3    for i←0 to _proofsize do
4        if  i==0 then
5            if  __index % 2 == 0 then
6                _proof [i] ← _hashTree [_index + 1];
7            else
8                _proof [i] ← _hashTree [_index - 1];
9            end
10       else
11           for j ← _startIndex to _endIndex do
12               if  _index == j || _index == j+1 then
13                   if _group % 2 == 0 then
14                       _updatedIndex ←_layerNodes+ _visitedNodes + _group + 1;
15                   else
16                       _updatedIndex← _layerNodes +_visitedNodes + _group - 1;
17                   end
18               else
19                   j← j+1;
20                   _group ← _group+1;
21               end
22           end
23           _proof [i] ← _hashTree [_updatedIndex];
24           _visitedNodes ← _visitedNodes + _layerNodes;
25           _layerNodes ← _layerNodes/ 2;
26           _startIndex ← _endIndex;
27           _endIndex ← _startIndex + _group;
28           _index ← _updatedIndex;
29               _group ← 0;
30       end
31   End
32   return  _proof
```

The Algorithm 4 outlines the process to generate proof for a specific attribute of a credential. The steps include: (i) The COC interacts with the CIC to get the hash tree of a credential, (ii) The function calculates the proof size as log (n), where n is the total number of leaf nodes in the tree, (iii) The hashes stored in the intermediate nodes are added in proof which are enough to reach up to the root, (iv) At the end, the algorithm returns proof.

Verification Contract (VC): This contract is responsible for maintaining a verification chain using (i) consent ID, and (ii) status of credential verification. This contract interacts with the COC and the CIC for consent and credential verification.

Algorithm 5 outlines the process to verify attribute value of a credential. The steps include: (i) The VC interacts with the COC to validate the caller of this function (who must be a verifier); (ii) The function adds related data to a Verification structure; (iii) The function calls a function to verify consent token value and validity of a token; (iv) gets root hash from the CIC contract using function getHashRoot (); (v) finds the hash value of an attribute value; (vi) calculates root hash using proof values; (vii) at the end, LogConsent () event notifies the system regarding the completion status.

Algorithm 5: Algorithm to verify proof

```
Inputs:    _consentID, _token, _proof,
           attributeValue, _Index
Outputs:   LogVerification (), _result
1   if     consent.getVerifierAddress(_consentID
           ) != msg.sender then
2          exit
3   else
4          credentialID = consent.getCredentialID
           (_consentID);
5          Verification [_consentID].consentID
           ←_consentID;
6          if  !( verifyTokenValidity(_consentI
               D,_token)) then
7              Verification [_consentID].status
               ← false;
8              return false;
9          else
10             _root ← credentialissuance.getHashTreeR
               oot (credentialID);
11             _hash ← Hash (attributeValue);
12             for i← 0 to _proof.length do
13                 _proofElement ← _proof[i];
14                 if  _index % 2 == 0 then
15                 _hash ← Hash (_hash,
                   _proofElement));
16                 else
17                     _hash ← Hash
                       (_proofElement, _hash));
18                 end
19                 _index ← _index / 2;
20             end
21             if  _hash== _root then
```

```
22                  Verification [_consentID].status ← true;
23                  return true
24              else
25                  Verification [_consentID].status ← false;
26                  return false;
27              end
28          end
29      Emit the event LogVerification ( msg.sender, _consentID, result)
```

Algorithm 6 outlines the process to verify token value and the validity of consent. The steps include: (i) The VC interacts with the COC to obtain the token value and the time period during which a verifier can only verify the credential value, (ii) compares the retrieved values from the COC contract to the values submitted by the verifier, (iii) returns a comparison result, at the end of the algorithm.

Algorithm 6: Algorithm to verify token and validity

```
Inputs:     _consentID,_token
Outputs:    _result
1   _flag ← true;
2   (token,validity) = consent.getConsent (_consentID);
3   if    !(token==_token && now <= validity) then
4         _flag ← false;
5   End
6   return _flag;
```

B. Inter Contract Communication

Our framework comprises four basic contracts such as Enrolment Contract, Credential Issuance Contract, Consent Contract, and Verification Contract. Smart contracts are responsible for maintaining the chain of transactions, including the entity enrolment chain, credential chain, consent chain, and verification chain. This section demonstrates the interaction between these chains. Fig. 1 shows the basic interaction between the smart contracts.

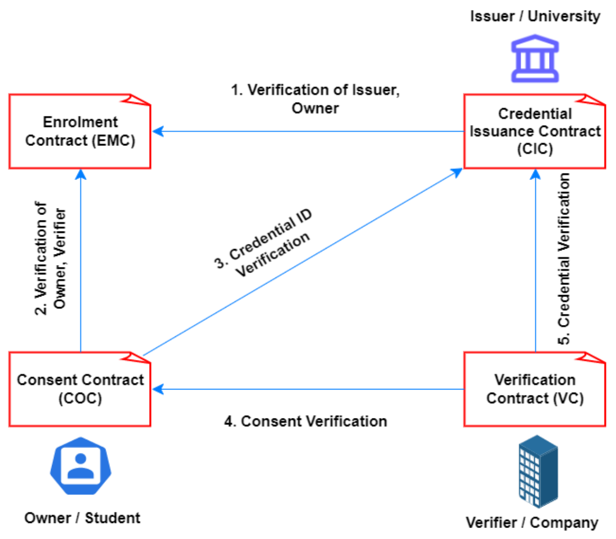

Issuer / University
1. Verification of Issuer, Owner
Enrolment Contract (EMC)
Credential Issuance Contract (CIC)
2. Verification of Owner, Verifier
3. Credential ID Verification
5. Credential Verification
Consent Contract (COC)
Verification Contract (VC)
4. Consent Verification
Owner / Student
Verifier / Company

Fig. 1. Inter-contract communication.

- CIC as a caller contract and EMC as a target contract: When a university wants to issue a credential to a student using the CredentialIssuance () function of the CIC, it can verify the identity of the student and the university through the getEntityClass () function of the EMC.
- COC as a caller contract and EMC as a target contract: When a student wants to share their consent for the verification of some credential using the addConsent () function of the COC, verification of identities of the student and the verifier can be done through the getEntityClass () function of the EMC.
- COC as a caller contract and CIC as a target contract: Verification of an issued credential ID can be done using the getCredentialID () function of the CIC.
- VC as a caller contract and COC as a target contract: The verifier only can verify the credential of a student, when it receives the consent for that credential using the Verification () function of the VC. It communicates with the getConsent () function of the COC for the consent verification.
- VC as a caller contract and CIC as a target contract: The VC communicates with the getCredential () function of the CIC to verify credential details.

C. Contract-Service Interaction

Our framework uses four services such as Off-Chain Data Service (OCS), Cryptographic Service (CS), Tokenization Service (TOS), and Smart Contract Service (SCS). Fig. 2 shows interaction between smart contract and service with required files.

Off-Chain Data Service (OCS): This service is capable of storing encrypted content to off-chain data storage like IPFS, Filecoin, Storj, Cloud storage (e.g. AWS), DBMS (e.g. Oracle) etc. In our work, we stored the encrypted private key to the decentralized data storage, IPFS. OCS returns an encrypted private key link (content identifier) which is used by EMC. This service also returns encrypted JSON object credential values link for the CIC.

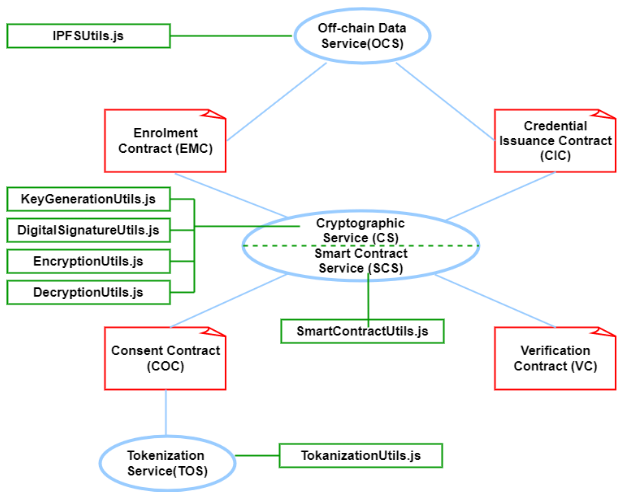


Fig. 2. Contract-service communications.

Cryptographic Service (CS): The cryptographic service performs various cryptographic functions for smart contracts. Communications between all entities are encrypted using cryptographic keys for security reasons. This service is responsible for generating keys and encrypting the private key for the EMC. CS is also used to generate a digital signature for credentials and to encrypt credential values for the CIC. Furthermore, it can be used to encrypt consent details for the COC. This service helps the VC to verify digital signature and decrypt the required credential values.

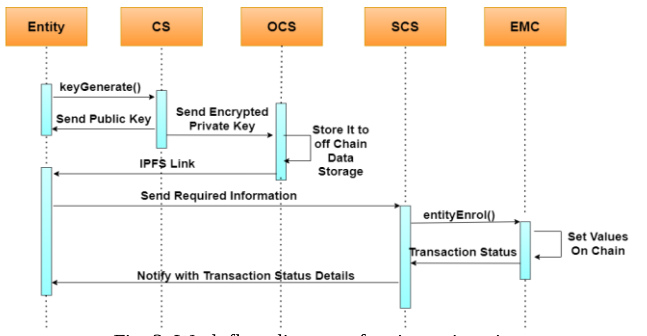
Tokenization Service (TOS): This service generates random consent token to secure our framework from replay attack for the COC.

Smart Contract Service (SCS): This service provides a web3 instance to connect with the Ethereum node using host address and port number. Functionalities of all the deployed contracts can be called using this service.

IV. Scheme Overview with Main Procedures

In this section of the paper, we present the four main processes of the education system, which include, entity registration, credential issuance, consent, and credential verification. To provide security against various attacks, each message transmission between entities utilizes the functionalities of CS.

Entity Registration: The registration process requires three core services and one smart contract to interact with each other, namely: (i) CS, (ii) OCS, (iii) SCS, and (iv) EMC. CS is used for generating a pair <P_U, P_R> for asymmetric key encryption and encrypting the private key for security concerns. The encrypted private key is stored on IPFS through off-chain data service (OCS). SCS provides a smart contract instance to interact with smart contracts on the Ethereum Blockchain. The EMC maintains an entity chain on the Blockchain. An entity follows the registration procedure to enroll valid entities (University, Student, and Company) to the system. Fig. 3 illustrates the system flow for adding a new entity to the system.



Entity
CS
OCS
SCS
EMC
keyGenerate()
Send Public Key
Send Encrypted Private Key
Store It to off Chain Data Storage
IPFS Link
Send Required Information
entityEnrol()
Transaction Status
Set Values On Chain
Notify with Transaction Status Details

Fig. 3. Work flow diagram of entity registration.

In the proposed framework, registering an entity involves the following steps: (i) an entity calls a cryptographic service to generate asymmetric keys using the Rivest-Shamir-Adleman (RSA) public key algorithm. We also offer the option to choose other cryptographic algorithms like DSA, elliptic curve etc.; (ii) CS sends the generated public key to the entity; (iii) Then, CS encrypts the private key with a secret key and sends it to OCS, (iv) OCS stores the encrypted private key on the decentralized data storage (Interplanetary File System, IPFS); (v) IPFS returns the content identifier (CID) to an entity; (vi) Entity sends request containing address of an entity, public key and CID to SCS for registration; (vii) SCS calls entityEnrolment () function of the EMC; (viii) the EMC validates entity information and adds transaction to the blockchain confirms transaction on Blockchain; (ix) EMC sends transaction status details to SCS; (x) SCS notifies the entity with the transaction status details.

Credential Issuance: Credential issuance process requires three core services and two smart contracts to interact with each other, namely: (i) CS, (ii) OCS, (iii) SCS, (iv) CIC, and (v) EMC. CS encrypts a JSON object which contains attribute values of a credential using the public key of a student. CIC maintains various chains on the Blockchain. Fig. 4 illustrates the system flow for adding a transaction of a credential.

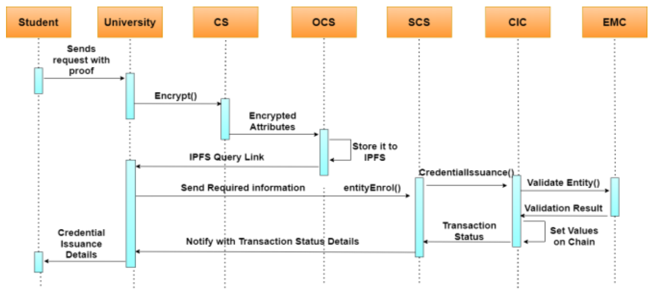


Fig. 4. Work flow diagram of credential issuance.

In the proposed framework, issuing a credential involves the following steps: (i) the student sends a request to the university to generate credential with proof; (ii) The university calls CS to encrypt the credential attribute values using the public key of a student; (iii) Then, CS sends the encrypted values to OCS; (iv) OCS stores the encrypted values on decentralized data storage (IPFS); (v) IPFS returns a content identifier (CID) to the university; (vi) The university sends a request containing the required parameters to SCS for the issuance process; (vii) SCS calls CredentialIssuance () function of the CIC; (viii) The CIC interacts with EMC to check enrolment details for a student; (ix) The EMC checks the enrolment status of the student on its entity chain and replies to CIC with the status; (x) The CIC stores all values in form of a transaction on the Blockchain; (xi) The Blockchain replies to SCS with transaction details through the CIC; (xii) SCS notifies the university with transaction status; (xiii) The university sends details of the credential issuance to the student in the form of a transaction.

Consent: Attribute sharing process requires two core services and three smart contracts to interact with each other, namely: (i) TOS, (ii) SCS, (iii) COC, (iv) EMC, and (v) CIC. TOS generates a random token which can be used only one time by verifier. The COC maintains a chain of consent details on the Blockchain. Fig. 5 illustrates the system flow for adding a transaction of a credential with consent.

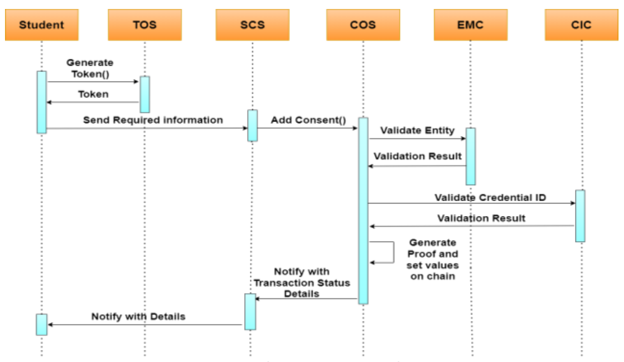


Fig. 5. Work flow diagram of consent.

In the proposed framework, sharing a credential involves following steps: (i) The student calls TOS to generate a random token; (ii) TOS applies different functions to generate a random token and sends it to student; (iii) Then, the student sends the required parameters to SCS; (iv) SCS calls the Consent () function of COC; (v) COC interacts with EMC to check verifier details; (vi) EMC checks the verifier's details against its entity chain and replies to COC with the status of a verifier; (vii) COC interacts with CIC for credential ID validation; (viii) CIC replies to COC with validation results; (ix) COC stores all relevant values in the form of a transaction on the Blockchain; (x) The Blockchain replies to SCS with transaction details through COC; (xi) SCS notifies the student with transaction status.

Credential Verification: Credential verification process requires one core service and three smart contracts to interact with each other, namely: (i) SCS, (ii) VC, (iii) COC, and (iv) CIC and VC verifies consent details and maintains chain of verification details on the Blockchain. Fig. 6 illustrates the different steps involved in the verification process.

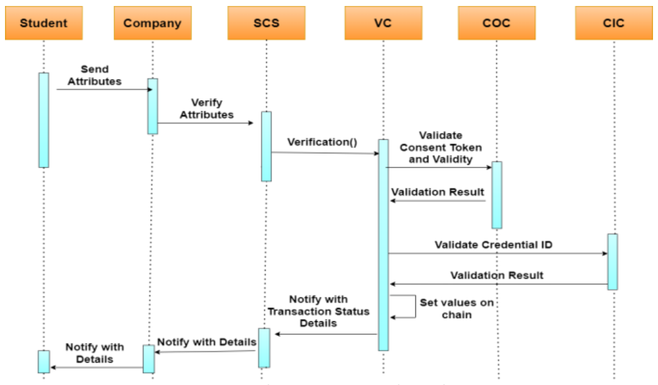


Fig. 6. Work flow diagram of verification.

In the proposed framework, the verification process involves the following steps: (i) the student shares a credential attribute value with a consent token to the company; (ii) The company sends received attributes for verification to SCS; (iii) SCS calls the Verification () function of VC; (iv) VC interacts with COC to check consent details; (v) COC checks it and replies to VC with verification details; (vi) VC interacts with CIC for credential validation, (vii) CIC replies to VC with verification results; (viii) VC stores all values in the form of a transaction on the Blockchain; (ix) The Blockchain replies to SCS with transaction details through VC; (x) SCS notifies the company with the transaction status; (xi) The company notifies the student with the verification status.

V. Experimentation and Results

A. Implementation Setup

We simulated our model using a machine with an Intel Core i3 processor, 4 GB of primary memory, 465 GB of secondary memory, and a 64-bit Windows operating system. For the implementation of our proposed framework, we opted Geth Blockchain [30], which is an Ethereum client implemented in Go language. REST APIs, Data API, Web3 and JSON RPC are responsible for smooth interactions between core services, smart contracts and the Blockchain. Node provides a highly scalable programming environment that satisfies multiple concurrent requests. Node package manager is a software registry used to install packages. The Solidity compiler compiles smart contracts and generates byte codes and application binary interfaces required for the deployments of the smart contracts. IPFS is used as a decentralized off chain data storage to store large amounts of data like private key, credential document etc.

B. Performance Evaluation

In this section, we evaluate the performance of our model, which primarily depends on cryptographic primitives, off-Chain Data Storage (IPFS), the proposed method, and smart contracts.

Cryptographic Primitives: We investigated the delay in end-to-end transmission due to cryptographic methods for real-time educational systems. We analyzed the latency during key generation for RSA, DSA and RSA-PSS algorithms with different key sizes. Fig. 7 shows the latency comparison of these algorithms. RSA is suitable for encryption, decryption, digital signature creation and verification. Although computational latency is high for security reasons, we used the RSA algorithm for cryptographic operations with a key size of 2048. We also analyzed the latency during encryption/decryption and signing/verification of credentials against their size, which is depicted in Fig. 8. We can see that the computational latency for encryption is high compared to decryption and verification latency is low compared to signing.

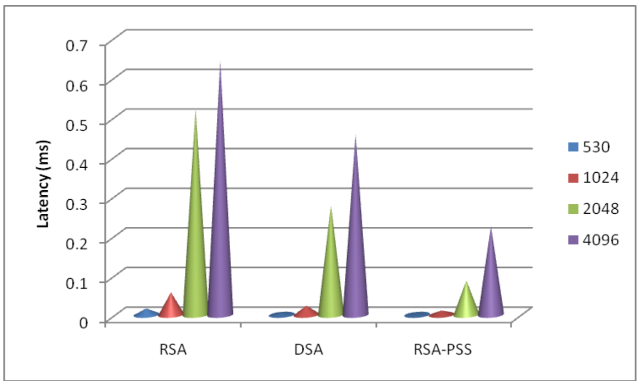


Fig. 7. Latency during key generation using public key cryptography.

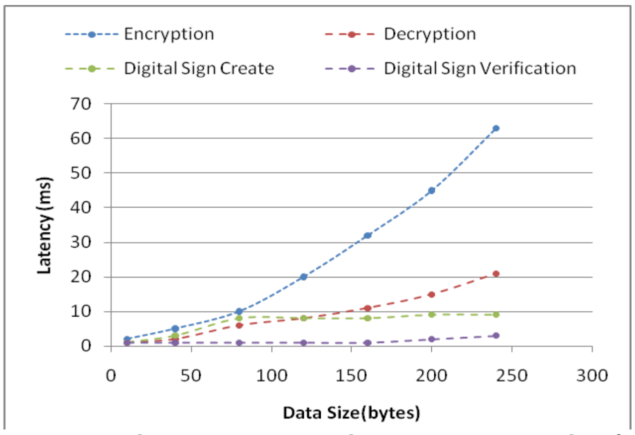


Fig. 8. Latency during RSA encryption, decryption, sign create and verify.

Off-Chain Data Storage (IPFS): The latency parameter of off-chain data storage is of utmost important because it directly affects the performance of the model. The latency comparison between storing and retrieving data in IPFS based on size of the data is shown in Fig. 9. We also observed the bandwidth utilization for IPFS network traffic to evaluate the effect of transaction time. Fig. 10 summarizes the IPFS-based storage system's input and output bandwidth utilization over time. The IPFS node required for our experiment used an average of 402 kbps bandwidth to execute send and receive transactions.

Proposed Method: Our model uses a method to place credential attribute values at random positions. Method uses four steps: transformation, succession, transposition and tree construction. The transformation step uses various equations to convert the attribute value into a mole value. We analyzed the impact of using these equations in our model in terms of the cost and time required to execute them. Fig. 11 shows that using the ASCII value requires less gas cost and time. So, in our model we used this method to convert attribute values to mole values. Fig. 12 shows the impact of the number of attributes on gas cost and execution time in the transformation, succession, transposition and tree construction.

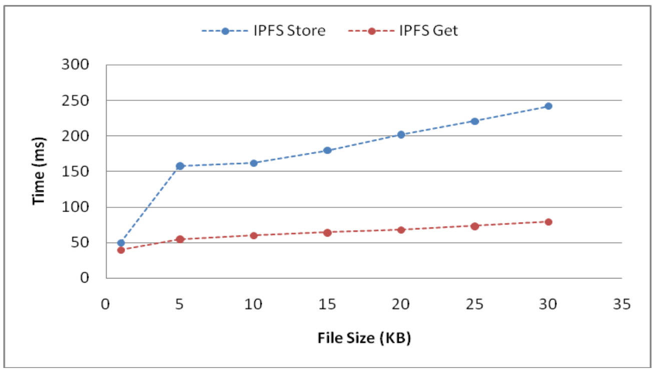


Fig. 9. Latency comparisons between IPFS store and retrieve.

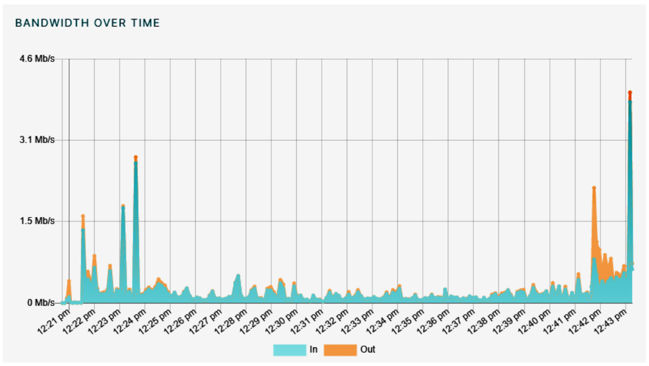


Fig. 10. Network bandwidth utilization of the IPFS.

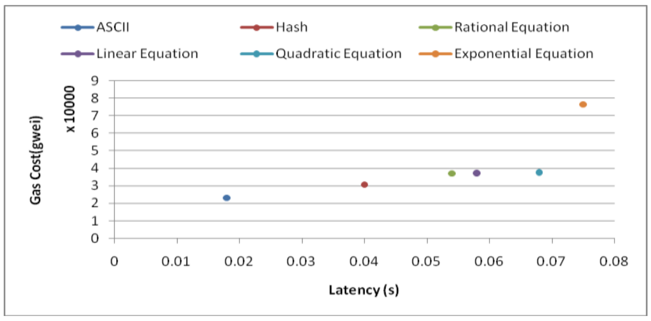
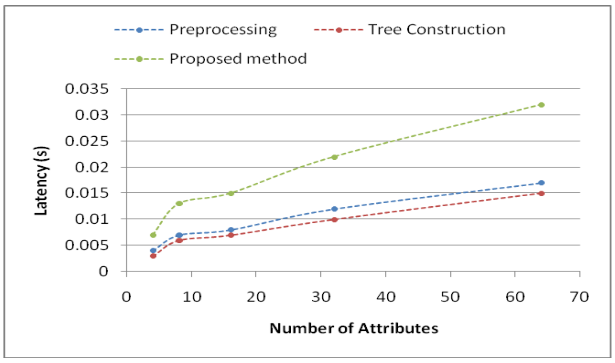
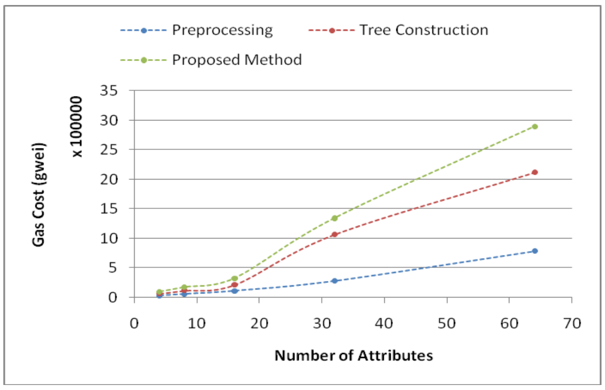


Fig. 11. Latency and Gas used to execute equations.



(a)



(b)

Fig. 12. (a) Comparative latency during execution of pre-processing and Tree construction. (b) Gas used during execution of pre-processing and tree construction.

Smart Contracts: We have implemented smart contracts in the Solidity language and the Solidity compiler compiles the solidity file and generates Application Binary Interface (ABI) and bytecode. Furthermore, we have deployed smart contracts on Ethereum using generated bytecode and ABI. ABI provides an interface to interact with Ethereum Virtual Machine (EVM) which stores executable bytecode so that we are able to interact with smart contract. Space occupied by ABI and bytecode affects the transaction time. Fig. 13 shows the space complexity of major smart contracts of our model, which shows that CIC occupies the highest space in EVM as it has implemented the proposed method. Moreover, Fig. 14 shows comparative latency and gas cost for major functions used in smart contracts which store transactions on Ethereum. The average system response time for enrollment, issuance, consent, and verification procedures is approximately 9s, 14s, 11s, and 1s respectively, as shown in Fig. 14(b). The latency of functions that are responsible for retrieving transaction values has range between 5ms to 15ms.

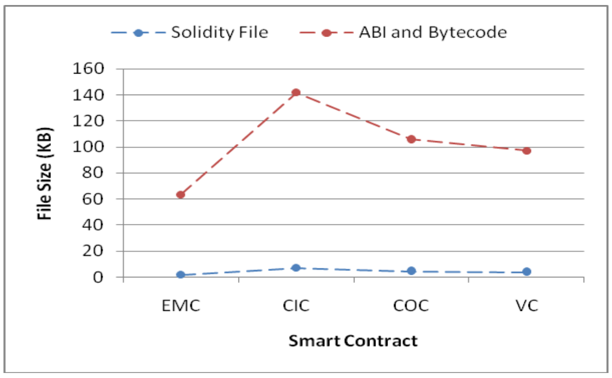
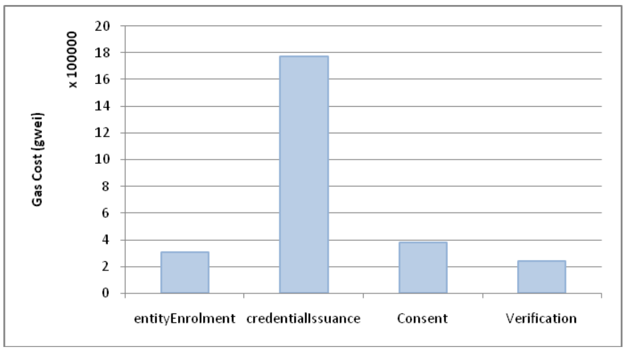
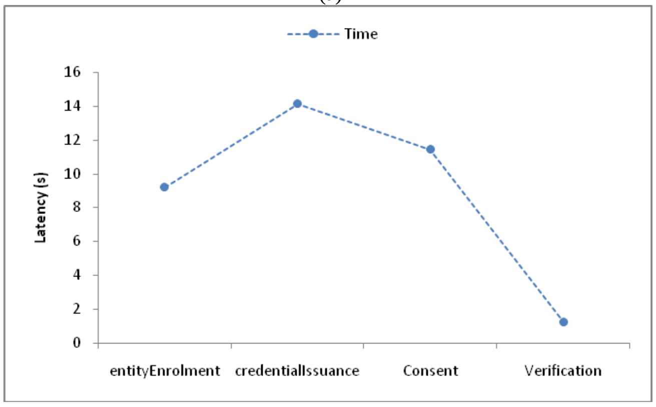


Fig. 13. Space complexity of core smart contracts.



(a)

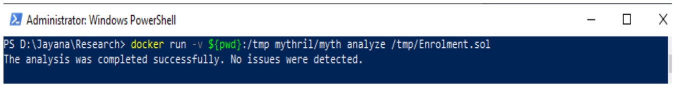


(b)

Fig. 14. (a) Gas used during execution of main functions of smart contracts. (b) Latency of execution of main functions of smart contracts.

C. *Security Analysis*

In this section we elaborate on the security aspects of the byte codes of smart contracts using the open-source tool, MyThril [31], which provides higher accuracy and is able to detect security vulnerabilities [32]. This tool is a command-line tool and written in Python. It relies on taint analysis, concolic analysis, and control flow checking of the EVM byte code to exploit security vulnerabilities in smart contracts. We used the docker image of the tool to perform security analysis. We performed the security analysis on the smart contracts used in our model using the MyThril tool. The tool detects security issues such as integer underflows, unchecked call return value, delegate call to untrusted callee, unprotected Ether withdrawal, assert violation, write to arbitrary storage location etc. Fig. 15 shows the security analysis output for Enrolment Contract and the tool returned a message stating, "The analysis was completed successfully, No issues were detected." We received the same result for the CredentialIssuance Contract, Consent Contract and Verification Contract. The results show that the smart contracts are secure against vulnerabilities.



```
PS D:\Jayana\Research> docker run -v ${pwd}:/tmp mythril/myth analyze /tmp/Enrolment.sol
The analysis was completed successfully. No issues were detected.
```

Fig. 15. Security verification of enrolment contract.

VI. Conclusion and Future Work

This work presents a new model that uses the Blockchain technology with the aim to provide privacy through selective disclosure for educational credentials. The model employs a method which randomises the credential attribute values and uses them for tree construction which can prove inclusion of a single or multiple attribute values from a tree. Our model describes the main smart contract algorithms to provide automation and core services for exchanging data between different entities providing cost reductions compared to third-party verification systems. Our model stores encrypted records on IPFS and only a hash of records on the Blockchain to achieve security. Credentials are shared with proof and consent token with validity so that verifier can verify the attribute within the provided time duration. However, we assumed that students have fulfilled the requirements of the university for credential. We provided a prototype model on the Ethereum Blockchain and evaluated its performance. The prototype has shown negligible overhead to achieve decentralization with end-to-end encryption. In the later part of this research, we intend to include monetization features in our model.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Jayana Kaneriya conducted research and wrote the manuscript. Hiren Patel enhanced the writing quality of the manuscript and manuscript language before it was to be submitted. All authors had approved the final version.

References

[1] P. Helo and A. H. M. Shamsuzzoha, "Real-time supply chain—A blockchain architecture for project deliveries," *Robot. Comput. Integr. Manuf.*, vol. 63, 101909, 2020.

[2] H. Patel and B. Shrimali, "AgriOnBlock: Secured data harvesting for agriculture sector using blockchain technology," *ICT Express*, 2021.

[3] S. Tanwar, K. Parekh, and R. Evans, "Blockchain-based electronic healthcare record system for healthcare 4.0 applications," *J. Inf. Secur. Appl.*, vol. 50, 102407, 2020.

[4] J. Zhang, M. Guo, B. Li, and R. Lu, "A transport monitoring system for cultural relics protection based on blockchain and internet of things," *J. Cult. Herit.*, vol. 50, pp. 106–114, 2021.

[5] I. Makhdoom, I. Zhou, M. Abolhasan, J. Lipman, and W. Ni, "PrivySharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities," *Comput. Secur.*, vol. 88, 101653, 2020.

[6] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, and P. D. Drobintsev, "Trust management in a blockchain based fog computing platform with trustless smart oracles," *Future Gener. Comput. Syst.*, vol. 101, pp. 747–759, 2019.

[7] S. K. Panda, G. B. Mohammad, S. Nandan Mohanty, and S. Sahoo, "Smart contract-based land registry system to reduce frauds and time delay," *Secur. Priv.*, vol. 4, no. 5, 2021.

[8] A. Kumari, A. Shukla, R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, "ET-DeaL: A P2P smart contract-based secure energy trading scheme for smart grid systems," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020.

[9] K. Kumutha and S. Jayalakshmi, "Blockchain technology and academic certificate authenticity—A review," *Expert Clouds and Applications*, Singapore: Springer Singapore, 2022, pp. 321–334.

[10] C. S. Wright, "Bitcoin: A peer-to-peer electronic cash system," *SSRN Electron. J.*, 2008.

[11] J. K. And and H. Patel. A blockchain-based conceptual framework for privacy preserving self- sovereign identity with selective disclosure. Iteejournal.org. [Online]. Available: http://www.iteejournal.org/v11no3june22_pdf4.pdf

[12] R. Arenas and P. Fernandez, "CredenceLedger: A permissioned blockchain for verifiable academic credentials," in *Proc. 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 2018.

[13] T. Arndt and A. Guercio, "Blockchain-based transcripts for mobile higher-education," *Int. J. Inf. Educ. Technol.*, vol. 10, no. 2, pp. 84–89, 2020.

[14] E. E. Bessa and J. S. B. Martins, "A Blockchain-based educational record repository," arXiv, [cs.OH], 2019.

[15] M. Han, Z. Li, J. He, D. Wu, Y. Xie, and A. Baba, "A novel blockchain-based education records verification solution," in *Proc. the 19th Annual SIG Conference on Information Technology Education*, 2018.

[16] A. Srivastava, P. Bhattacharya, A. Singh, A. Mathur, O. Prakash, and R. Pradhan, "A distributed credit transfer educational framework based on blockchain," in *Proc. 2018 2nd International Conference on Advances in Computing, Control and Communication Technology (IAC3T)*, 2018.

[17] F. P. Oganda, N. Lutfiani, Q. Aini, U. Rahardja, and A. Faturahman, "Blockchain education smart courses of massive online open course using business model canvas," in *Proc. 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS)*, 2020.

[18] D. Lizcano, J. A. Lara, B. White, and S. Aljawarneh, "Blockchain-based approach to create a model of trust in open and ubiquitous higher education," *J. Comput. High. Educ.*, vol. 32, no. 1, pp. 109–134, 2020.

[19] J. Guo, C. Li, G. Zhang, Y. Sun, and R. Bie, "Blockchain-enabled digital rights management for multimedia resources of online education," *Multimed. Tools Appl.*, vol. 79, no. 15–16, pp. 9735–9755, 2020.

[20] M. Turkanovic, M. Holbl, K. Kosic, M. Hericko, and A. Kamisalic, "EduCTX: A blockchain-based higher education credit platform," *IEEE Access*, vol. 6, pp. 5112–5127, 2018.

[21] S. M. Skh Saad and R. Z. Raja Mohd Radzi, "Comparative review of the blockchain consensus algorithm between Proof of Stake (POS) and Delegated Proof of Stake (DPOS)," *Int. J. Innov. Comput.*, vol. 10, no. 2, 2020.

[22] H. Li and D. Han, "EduRSS: A blockchain-based educational records secure storage and sharing scheme," *IEEE Access*, vol. 7, pp. 179273–179289, 2019.

[23] G. Wang, H. Zhang, B. Xiao, Y.-C. Chung, and W. Cai, "EduBloud: A Blockchain-based Education Cloud," in *Proc. 2019 Computing, Communications and IoT Applications (ComComAp)*, 2019.

[24] M. A. Rashid, K. Deo, D. Prasad, K. Singh, S. Chand, and M. Assaf, "TEduChain: A blockchain-based platform for crowdfunding tertiary education," *Knowl. Eng. Rev.*, vol. 35, no. e27, 2020.

[25] L. M. Palma, M. A. G. Vigil, F. L. Pereira, and J. E. Martina, "Blockchain and smart contracts for higher education registry in Brazil," *Int. J. Netw. Manage.*, vol. 29, no. 3, p. e2061, 2019.

[26] O. S. Saleh, O. Ghazali, and M. E. Rana, "Blockchain based framework for educational certificates verification," *J. Crit. Rev.*, vol. 7, no. 03, 2020.

[27] I. Alnafrah and S. Mouselli, "Revitalizing blockchain technology potentials for smooth academic records management and verification in low-income countries," *Int. J. Educ. Dev.*, vol. 85, 102460, 2021.

[28] S. I. M. Ali, H. Farouk and H. Sharaf, "A blockchain-based models for student information systems," *Egypt. Inform. J.*, vol. 23, no. 2, pp. 187–196, 2022.

[29] R. A. Mishra, A. Kalla, A. Braeken, and M. Liyanage, "Privacy protected blockchain based architecture and implementation for sharing of students' credentials," *Inf. Process. Manag.*, vol. 58, no. 3, 102512, 2021.

[30] Getting started with Geth. go-ethereum. [Online]. Available: https://geth.ethereum.org/docs/getting-started

[31] Mythril: Security analysis tool for EVM bytecode. Supports smart contracts built for Ethereum, Hedera, Quorum, Vechain, Roostock, Tron and other EVM-compatible blockchains.

[32] M. di Angelo and G. Salzer, "A survey of tools for analyzing ethereum smart contracts," in *Proc. 2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, 2019.