
Blockchain DLLOC Case Study

Blockchain's application in
academic credentials verification

Need of using Blockchain for credential Verification

- A large number of counterfeit certificates in circulation is a huge problem.
- There is no assurance that the job applicant/student would provide the company/organization with accurate information.
- Verification by conventional method is very time consuming.
- An undeserving candidate might get the job/admission causing threat to organizations integrity.
- Fake degree submission constituted nearly 28% of education discrepancies in 2020 in India.
- A racket of 36,000 fake degrees was busted in Manav Bharti University (MBU) in 2021.

About Blockcert

Title: [Blockchain based Academic Certificate Authentication System Overview](#)

Project: Blockcert

MIT Media Lab's Blockcerts Project:

- Blockcerts is an open standard for building apps that issue and verify blockchain-based official records
- MIT Media Lab introduced the Blockcerts project as a solution.
- Blockcerts primarily involves linking the hash value of local files to the blockchain.

BTCert Feature

Multi-Signature Scheme for Authentication:

- Utilizing a multi-signature scheme to enhance the authentication of certificates.
- This can make it more difficult for counterfeiters to create fake certificates.

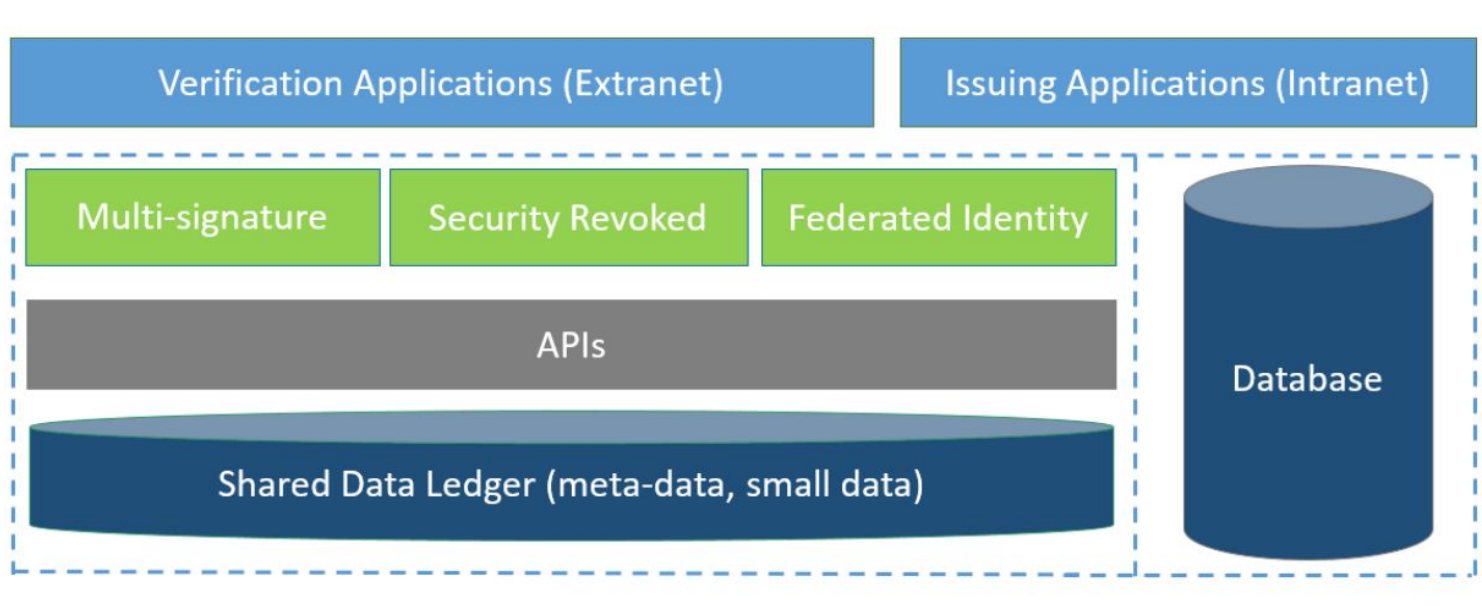
Safe Revocation Mechanism:

- Implementing a secure revocation mechanism to improve the reliability of certificate revocation.
- Ensuring that certificates can be invalidated if necessary, for example, if they are found to be fraudulent.

Secure Federated Identification:

- Establishing a secure federated identification system to confirm the identity of the issuing institution.
- This helps in verifying the authenticity of certificates and the credibility of the institution.

System Architecture Overview



System Architecture Overview

Verification Application:

- Responsible for verifying the authenticity and integrity of issued certificates.
- Performs multiple checks, including validation of authentication codes, matching hashes with local certificates, confirming hash inclusion in a Merkle tree, verifying the Merkle root on the blockchain, checking certificate revocation status, and validating certificate expiration dates.

Issuing Application:

- Manages the core business logic for certificates.
- Handles processes such as certificate application, examination, signing, issuing, revocation.

Blockchain:

- Serves as the underlying infrastructure of trust and a distributed database.

Local Database (MongoDB):

- Utilized for managing JSON-based certificates.

Participating Entities

Student:

- **The student initiates the process by applying to the school for a credential or certificate.**

Checker (Certifiers):

- **Certifiers, who are responsible for verifying and validating the students' information, examine the application.**

Issuer (Academic Committee Members):

- **The majority of the academic committee members sign the merged credential with their private keys.**
- **This multi-signature approach enhances the security and authenticity of the certificate.**

Participating Entities

System:

- It broadcasts the transaction to the blockchain network, which includes the Merkle root for all the certificates being issued.

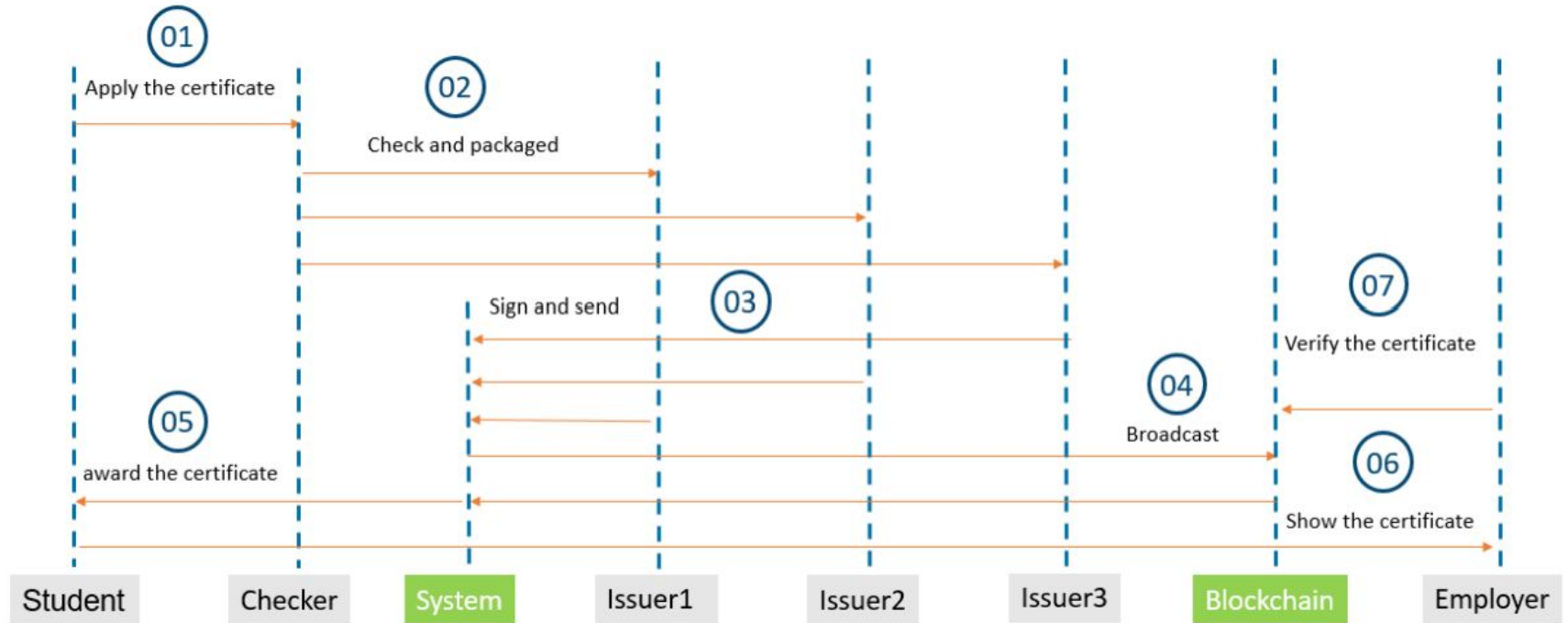
Blockchain:

- The blockchain records and stores the transaction, including the Merkle root, ensuring data integrity and security.

Employer:

- When the student applies for a job, they provide the JSON-based certificate to the employer as proof of their academic credentials.
- The employer accesses the blockchain to verify the certificate's authenticity.

Workflow



Database Architecture

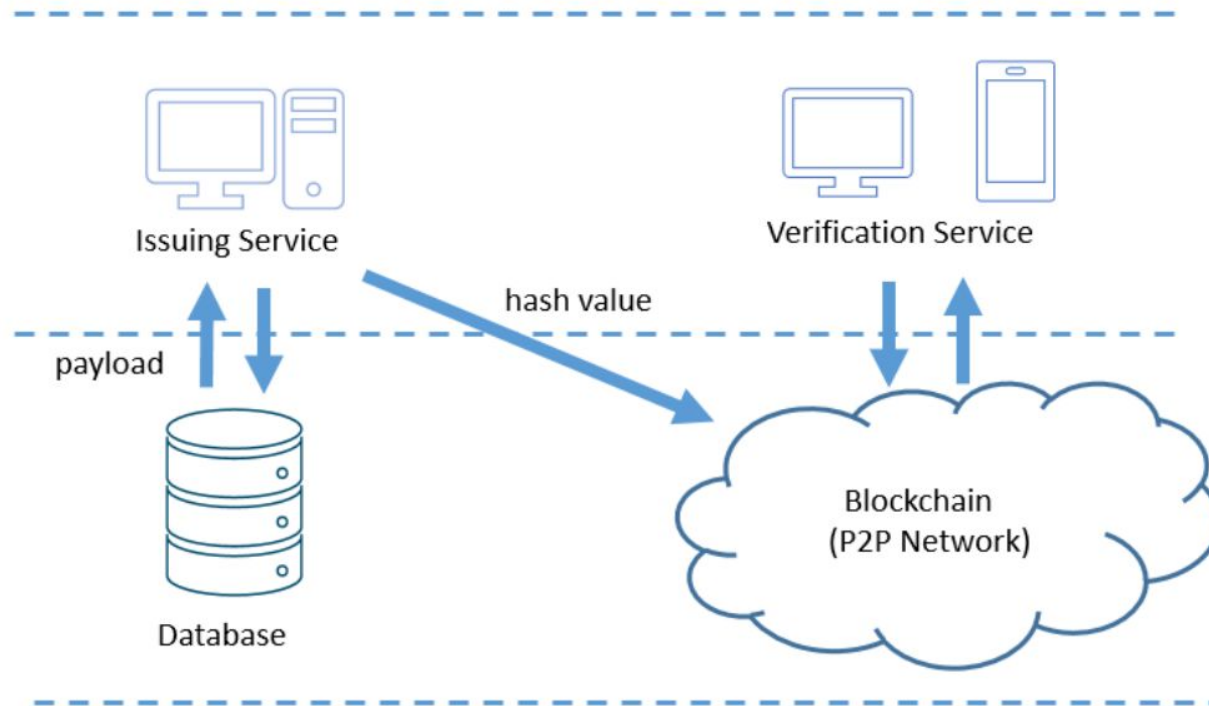
Public Authentication Data:

- This category contains data that is intended to be accessible to the public.
- The public authentication data is released to and recorded on the blockchain.
- Information in this category may include details related to certificate authentication, verification, or metadata that is meant to be publicly available for transparency and validation purposes.

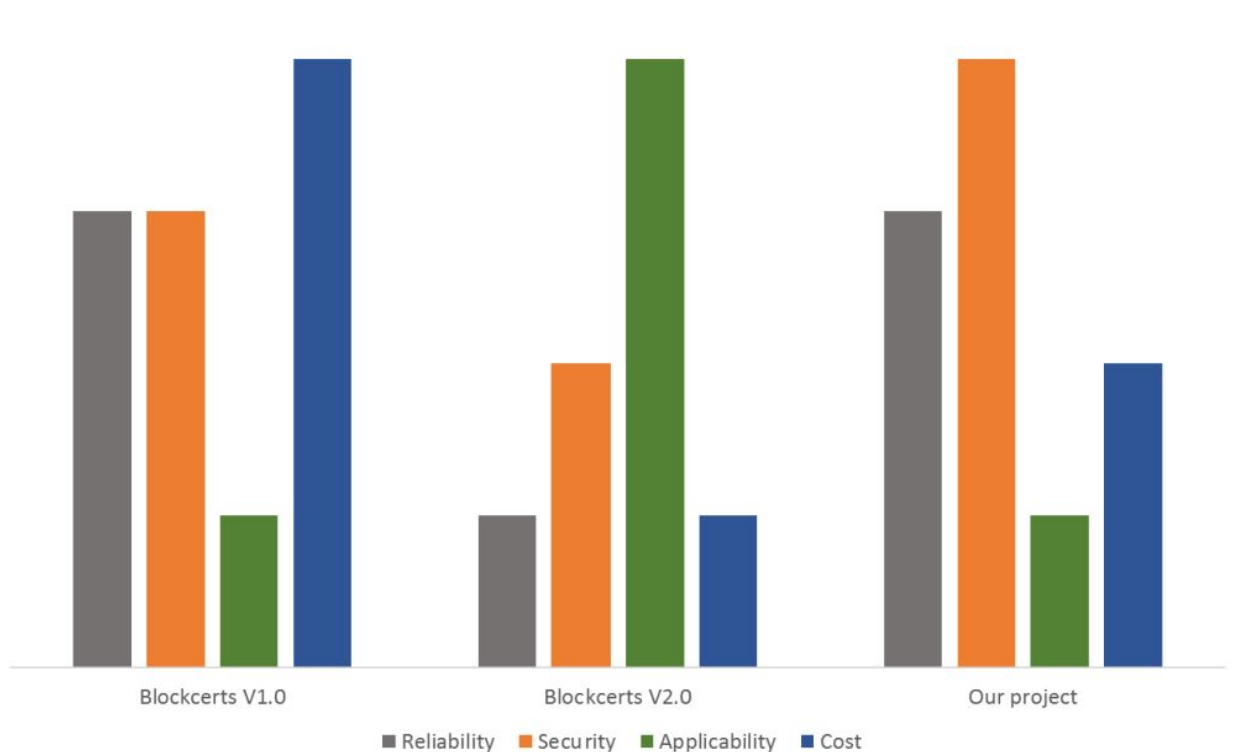
Private Certificate Data:

- Private certificate data is the second category, and it is treated differently from public data.
- This data is stored in a MongoDB database.
- It is securely protected and isolated within an intranet environment.

Database Architecture



Comparison between Blockcert and BTcert





Exp6: SmartContract

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract StudentCredentials {
5     address public owner;
6
7     // Struct to represent academic credentials
8     struct Credential {
9         string institution;
10        string degree;
11        uint256 year;
12        bool isVerified;
13    }
14
15    // Mapping to store credentials for each student
16    mapping(address => Credential) public credentials;
17
18    constructor() { 630904 gas 606000 gas
19        owner = msg.sender; // Contract creator is the owner
20    }
21
22    // Function for students to add or update academic credentials
23    function addOrUpdateCredential(string memory institution, string memory degree, uint256 year) public { infinite gas
24        credentials[msg.sender] = Credential(institution, degree, year, false);
25    }
26
27    // Function to update the isVerified status by the administrator
28    function updateStatus(address student, bool status) public { 27186 gas
29        require(msg.sender == owner, "Only the owner (administrator) can change verification status.");
30        credentials[student].isVerified = status;
31    }
32 }
33
```

Exp6: SmartContract

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 import "./StudentCredentials.sol"; // Import the StudentCredentials contract
5
6 contract CredentialVerifier {
7     address public owner;
8     address public administrator; // Address with the authority to change isVerified
9     StudentCredentials public studentCredentialsContract; // Reference to the StudentCredentials contract
10
11     constructor(address _studentCredentialsAddress) {  infinite gas 360600 gas
12         owner = msg.sender; // Contract creator is the owner
13         administrator = msg.sender; // Set the initial administrator to the contract creator
14         studentCredentialsContract = StudentCredentials(_studentCredentialsAddress);
15     }
16
17     // Struct to represent academic credentials
18     struct Credential {
19         string institution;
20         string degree;
21         uint256 year;
22         bool isVerified;
23     }
24
25     // Function to verify academic credentials
26     function verifyCredential(address student) public view returns (string memory) {  infinite gas
27         string memory institution;
28         string memory degree;
29         uint256 year;
30         bool isVerified;
31
32         (institution, degree, year, isVerified) = studentCredentialsContract.credentials(student);
33
34         // Check if credentials exist and are verified
35         if (bytes(institution).length > 0 && isVerified) {
36             return "The details entered are correct" ;
37         } else {
38             return "The details entered are incorrect, please update";
39         }
40     }
41 }
42
```

Exp 7: Integration with metamask

The screenshot displays the Remix IDE interface with the following components:

- Browser Tabs:** Includes 'Remix - Ethereum IDE', '(3) WhatsApp', 'Sepolia Faucet', 'BC DLOC Lab6 - Google', 'Blockchain ganache - G...', and 'Untitled docu...'. The active URL is `remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js`.
- Left Panel (DEPLOY & RUN TRANSACTIONS):**
 - ENVIRONMENT:** Set to 'Injected Provider - MetaMask' on the 'Sepolia (11155111) network'.
 - ACCOUNT:** Address '0x729...71192' with a balance of '0.5 ether'.
 - GAS LIMIT:** Set to '3000000'.
 - VALUE:** Set to '0' Wei.
 - CONTRACT:** Selected as 'StudentCredentials - contracts/artif'.
 - Buttons:** 'Deploy' (orange), 'Publish to IPFS' (checkbox), and 'At Address' (blue).
 - Footer:** 'Transactions recorded 1' and 'Deployed Contracts'.
- Code Editor:** Displays the `CredentialVerifier.sol` file with the following Solidity code:

```
1 pragma solidity ^0.8.0;
2
3
4 import "../StudentCredentials.sol"; // Import the StudentCredentials contract
5
6 contract CredentialVerifier {
7     address public owner;
8     address public administrator; // Address with the authority to change isVerified
9     StudentCredentials public studentCredentialsContract; // Reference to the StudentCredentials contract
10
11     constructor(address _studentCredentialsAddress) {
12         owner = msg.sender; // Contract creator is the owner
13         administrator = msg.sender; // Set the initial administrator to the contract creator
14         studentCredentialsContract = StudentCredentials(_studentCredentialsAddress);
15     }
16
17     // Struct to represent academic credentials
18     struct Credential {
19         string institution;
20         string degree;
21         uint256 year;
22         bool isVerified;
23     }
24
25     // Function to verify academic credentials
26     function verifyCredential(address student) public view returns (bool) {
27         string memory institution;
28         string memory degree;
29         uint256 year;
30         bool isVerified;
31     }
```
- MetaMask Notification Overlay:**
 - Header:** 'MetaMask Notification' with 'Sepolia test network' selected.
 - Account:** 'Account 2' with a 'New contract' button.
 - URL:** `https://remix.ethereum.org` with a 'CONTRACT DEPLOYMENT' button.
 - DETAILS:**
 - Gas (estimated):** 0.00182384 SepoliaETH.
 - Very likely in < 15 seconds.**
 - Max fee:** 0.00182388 SepoliaETH.
 - Total:** 0.00182384 SepoliaETH.
 - Amount + gas fee:** Max amount: 0.00182388 SepoliaETH.
 - Buttons:** 'Reject' (white) and 'Confirm' (blue).

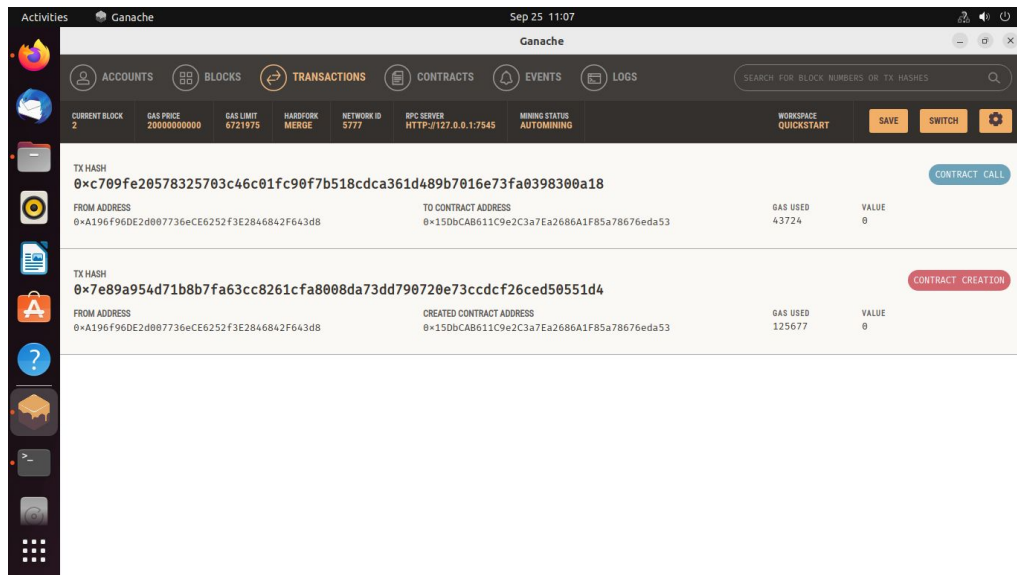
Exp8: Integration with Ganache

The screenshot displays the Remix IDE interface with the following components:

- Top Bar:** Shows the URL `remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js`.
- Left Panel (Deploy & Run Transactions):**
 - ENVIRONMENT:** Injected Provider - MetaMask.
 - ACCOUNT:** 0xd60...D500A (100 ether).
 - GAS LIMIT:** 3000000.
 - VALUE:** 0 Wei.
 - CONTRACT:** StudentCredentials - StudentCredier.
 - Buttons:** Deploy, Publish to IPFS, At Address, Load contract from Address.
 - Transactions recorded:** 1.
 - Deployed Contracts:** (Empty list).
- Center Panel (Code Editor):** Displays the `StudentCredentials.sol` contract code.

```
4 contract StudentCredentials {
5     address public owner;
6
7     // Struct to represent academic credentials
8     struct Credential {
9         string institution;
10        string degree;
11        uint256 year;
12        bool isVerified;
13    }
14
15    // Mapping to store credentials for each student
16    mapping(address => Credential) public credentials;
17
18    constructor() {
19        owner = msg.sender; // Contract creator is the owner
20    }
21
22    // Function for students to add or update academic credentials
23    function addOrUpdateCredential(string memory institution, string memory degree,
24        Credential memory credential) public {
25        credentials[msg.sender] = Credential(institution, degree, year, false);
26    }
27
28    // Function to update the isVerified status by the administrator
29    function updateIsVerified(address student, bool status) public {
30        require(msg.sender == owner, "Only the owner (administrator) can change verification status.");
31        credentials[student].isVerified = status;
32    }
}
```
- Right Panel (Ganache):**
 - Account 2:** New contract.
 - URL:** `https://remix.ethereum.org`.
 - CONTRACT DEPLOYMENT:** Button.
 - DETAILS:** Tab selected.
 - Gas (estimated):** 0.00328196 ETH.
 - Total:** 0.00328196 ETH.
 - Buttons:** Reject, Confirm.

Exp8: Integration with Ganache



addOrUpdateC...

"VESIT", "CS", 2020

▼

updateStatus

address student, bool status

▼

credentials

0xAb8483F64d9C6d1EcF9b8•

▼

0: string: institution VESIT

1: string: degree CS

2: uint256: year 2020

3: bool: isVerified false

owner