BCTSI

EDUCHAIN: A SECURE BLOCKCHAIN APPROACH TO CERTIFICATION SYSTEMS

Education is one of the most important pillars of future planning Education plays a vital role in the production of knowledge and its distribution. Learning based on the online is accepted and adopted by many educational firms throughout the world during this COVID-19 pandemic to deliver a quality of education, training, certification, and even awarding higher degrees. The term quality education involves learners, environments for learning, content, process, and outcomes. To increase the quality of education, teachers need to apply different instructional methods such as online-based learning, e-learning, etc. and should carry out proper student evaluation and assessment processes. It is also important that traditional lesson styles should be reformed to safe and online-based especially during this COVID-19 pandemic. Currently educational institutions are awarding academic certificates that the students have obtained However, its student's responsibility to prove its authenticity. Most of the students' needs to depend on awarding educational institutes are third party verification companies to prove their certification is authentic.

Blockchain is a new technology and a revolution that has imposed itself on various areas of life. The application of blockchain in education is still in its infancy. The blockchain-based technology product EDUCHAIN proves that blockchain-based technology could undercut the educational institutions' central role as certification verifying agents and provide students with more flexible opportunities. Blockchain and education technologies can work in tandem to improve: - The transparency and record-keeping e.g. certification process and minimize forged degrees. - Traceability and file storage of academic programs and any committed changes across different participants. - Resistant and defense procedures to cyberattack and cybercrime. - Rewards mechanism via blockchain nodes. - Lower cost of operations through smart contract and process automation

Need Of EDUCHAIN Product to Market

to address the current challenges faced with misuse of issuing educational credentials by many parties involved in the educational sector. **EDUCHAIN** is a reliable, trusted, and verifiable blockchain-backed repository for academic course work and student/faculty certification. **EDUCHAIN** allows auditors, cultural bureaus, educational authorities, accreditors, and validators to participate and a trustless fashion. The model utilizes the structure of DLT and smart contracts in ELC architecture to allow secure delivery and storage of any educational program/curricular and enables efficient and trusted verification and authorization process

Phone:+1 2177412411
Fax : +1 2243861099
www.bctsi.com
info@bctsi.com
617 W Parkside Dr
Palatine IL 60067

EDUCHAIN utilizes Ethereum smart contracts and leverages the benefits of IPFS, to store the certificates in a decentralized file system. Up to our best knowledge, there is no other work proposing similar solution. We intend to ensure confidentiality and validity of the certificates. So, to provide confidentiality, the certificates will be encrypted with AES algorithm before creating the transaction. On the other hand, validity is ensured by signing the transaction with the private key of issuer university. Furthermore, the system offers a user-friendly interface, making it easier to access certificates, without the need to create a private wallet, or without even knowing what blockchain technology is. Users will be able to access their certificate from different devices, having an internet connection, this includes old smartphones, personal computers, laptops, or even smart TVs, and with the introduction of a QR scanner, users will not even need to remember anything. With the use of optional features such as IPFS, PWA, Cameras and more, the user can access even more information, such as a physically scanned document, Photo ID, with less effort.

EDUCAHIN ATTRIBUTES OF THE BLOCKCHAIN TECHNOLOGY A blockchain is a distributed digital record of transactions. The terminology comes from its setup structure where blocks are connected to each other chronologically. These blocks are linked together with each other, and in accordance with the implemented consensus protocol, in a single list which is called a chain. The current implementation of blockchains is now seeing in recording cryptocurrencies transactions. The notion of decentralization is a core component; hence, any involved transactions cannot be altered without the alteration of all concurrent blocks. The key characteristics of the blockchain include decentralization, persistence, anonymity, and audibility.

BLOCKCHAIN TECHNOLOGIES

The idea of the blockchain was initially introduced by Satoshi Nakamoto and later in 2009 implemented the 1st digital cryptocurrency known as Bitcoin. The structure of the Blockchain is shown in Figure 1. Blockchain then has become a core component of bitcoin's operation and the de facto inspirational to other projects. It provides a secure exchange and traceability in the event of any failures in case of a security breach. In a shift of paradigm, the blockchain removes centralized authority which is present in-between multiple entities that are processing the financial and other transactions on data using a public ledger which is incorruptible, immutable, and decentralized in nature.

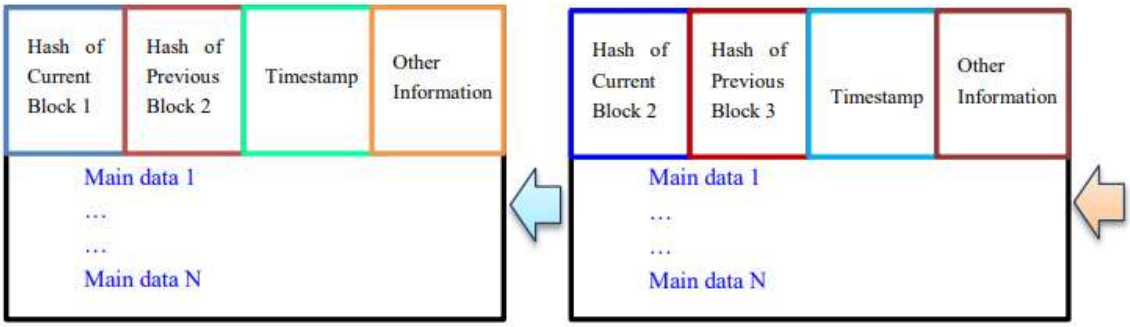


Figure 1: Structure of Blockchain

EDUCHAIN application: This application should be able to connect to a series of peer-topeer networks such as Ethereum, IPFS and other systems depending on future technology adoption; It should be user friendly and should be supported by a wide range of devices as they will be used by users to easily access the data; On the other hand this application will be used by institutions to assist in the development of the project and in the deployment of certificates; Gather informative and usage data. We have chosen Ethereum Blockchain to implement our solution, because this technology offers automation, ease of access and flexibility and deploying Smart Contracts will offer transparency, immutability, and security. The Smart contacts will be deployed in Remix Solidity Word processor/Ide

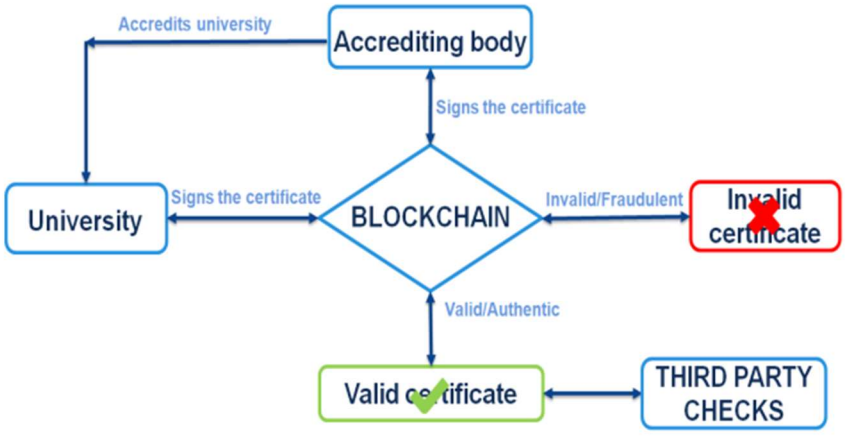


Figure 1. Architecture of issuing certificates

Issuing Certificates Interface. Once a student graduates or a certificate needs to be deployed, certain personnel is appointed to fill the needed forms and information. Once verified and approved, this information is signed by appropriate authorities, and if this university is accredited, the certificate is immediately added as APPROVED into the system, otherwise it is marked as PENDING. The encryption key is communicated to the student. This interface will be a typical web interface on a local network within the university.

Accrediting University Interface. An accreditation body takes care of adding universities into the approved list. The common attribute all universities have is a name or unique identifier. The accreditor can verify a university by specifying its public address and additional information such as name. The university is marked as VALID and it can issue valid certificates. In case this university is found to be fraudulent, it can be marked as Invalid and it can no longer issue certificates. The verification process is easy. User needs to put the certificate ID into the appropriate field, and the requested information will be shown. In upcoming work, other fields specific to the university can be implemented, but these will depend on further decisions of system implementation.

To verify the entered ID, in the client side can be used validator fields in front-end and backend. Furthermore, rate limiters, multiple layer protections and other techniques can be implemented to protect the system from DDoS attack.

On the other hand, the service rates are balanced between two main stakeholders of that service, graduated students, and employers. For students, the validity check of certificates is done at low cost and for employers, the system offers an easy authenticity and validity check of the certificate from trustable sources

Development Environment

A development environment needs to be flexible, temporary, and easy to set up. Due to the nature of the project, a variety of different technologies will be used to have a usable product such as: blockchain, smart contracts, IPFS (Interplanetary File System), PWA (Progressive Web Applications).

To develop the verifier application, following technologies need to be used:

• **Web3JS,** which enables the client to communicate with the blockchain network and enable us to deploy, view, add, modify, and validate contracts on the blockchain.
• **IPFS (InterPlanetary File System)** to have more data, private logs, and efficiency. It helps to make the web faster and safer. IPFS, as a peer-to-peer hypermedia protocol, is used for storing and sharing data in a distributed file system In our project it is planned to be used in storing photos, ID, avatars, or another biometric ID.

• **PWA (Progressive Web Application)** to use its benefits such as: Cross-platform, easy to use, enhances security. PWA is a type of application build and delivered using web technologies such as HTML, CSS, JavaScript and is intended to work on many platforms and devices such as mobiles and desktops. In this case, it will be used to make the verification process more accessible and available, without special hardware and additional software. PWA is delivered through common web protocols, but has reasonably more potential than a website, thus it needs to fulfil some requirements before being called a PWA, some of which are: use https, ability to be installable, respond offline, be fast and reliable.
• **OpenShift Online**, which allows us to deploy containers form pre-build images or from other sources without having to worry about server infrastructure and without introducing extra costs.
• **Docker**. Docker is the defacto standard to build and share containerized applications from desktop to the cloud Docker allows us to have images that run on multiple containers and can be easily redeployed and scaled. We are using a 'dockerfile' hosted on github that contains all the commands a user could call on the command line to assemble an image and build feature that hub.docker.com comes with.

The deployed smart contracts

The smart contracts are self-executing programs that utilize blockchain technology to digitally enforce verifying or negotiation of a contract. Therefore, they offer credibility between contracting parties, without involving third parties.

Thus, in our proposed implementation, we have deployed two smart contracts, called 'Certificate Issuer' and 'Permissions and role manager', and have implemented different functions inside them, to make the application works properly.

'Certificate Issuer' smart contract is responsible for adding and retrieving certificates on the network. Functions that are included in this certificate are:

• **addCert()** – responsible for adding new certificates to the system
• **getCert ()** – responsible for retrieving certificates using the student ID

'Permission and role manager' smart contract is responsible for managing accounts on the network. It defines each action an entity can perform. Functions that are included in this certificate are:
• **setMaster()** function – sets an account which will have full permissions to the system
• **getAcds()** function – returns a list of accreditor accounts
• **getAcd()** function – returns specific details about an accreditor
• **addAcd()** function – adds new accreditor to the system
• **switchAcd ()** function – switches the state of the accreditor to valid or invalid.
• **addUni()** function – adds a new university to the network
• **switchUni()** function – defines whether the university can add new approved certificates

Algorithm 1 and algorithm 2 presents the pseudocodes of 'Verifying student' and 'Sending signed transaction' respectively.

Algorithm 1: Pseudocode of verifying student (Client Side)

```
1:  function getCert (id) {
2:          ConnectToContract(network, address);
3:          return certificateData;
4:          }
5:
6:  function decryptData(data, encryptionKey) {
7:          decrypted = AES.decrypt(data, encryptionKey);
8:          if(decryptFailed) {
9:                alert("Error, failed decrypting");
10:         }
11: return decrypted;
12: }
```

Algorithm 2: Pseudocode of verifying student (Client Side)

```
1:  function sendTransaction(StudentID, encryptedData, state, contractAddress){
2:        privateKey = promptUser("Enter Private Key");
3:        certEncodedABI = contract.methods.addCert(StudentID, encrypted Data,
    state).encodeABI( );
4:     transaction = {
5:                    from: universityAddress,
6:                    Nonce: nonce + 1,
7:                    to: contractAddress,
8:                    gas: 3000000,
9:                    Data: certEncodedABI
10:                   };
11: web3.eth.accounts.signTransaction(transaction,
    privateKey).then(signedTransaction => { sent Transaction =
12: web3.eth.sendSignedTransaction(signedTransaction.rawTransaction);
13:
14: sentTransaction.on('confirmation', (confirmationNumber, receipt) => {
15:                   log(confirmationNumber, receipt)
16:                   })
17:
18:                   sentTransaction.on('receipt', receipt => {
19:                         log(receipt)
20:                   })
21:
22:                   sentTransaction.on(error, error => {
23:                         log(error)
24:                         })
25:               }
26: }
```

Phone:+1 2177412411
Fax : +1 2243861099
www.bctsi.com
info@bctsi.com
617 W Parkside Dr
Palatine IL 60067

University Application

The university interface is responsible for creating, signing, and issuing certificates to the network. Apart from those discussed, there are other functionalities involved in this interface, such as taking care of safely encrypting and sending the data, distributing the passphrase and certificate details, as well as offering a seamless experience for the user.

An important issue to concern is encrypting data, because of (1) sensitive information is transported to the blockchain and (2) how the transactions are made. There are multiple ways of encrypting data: (1) The first one is to use 'no encryption' at all and considering that we will be dealing with sensitive data (this is immediately ruled out); (2) The next choice is to handle encryption in the smart contract, but the issue here is that the data is transferred unencrypted, thus it remains visible in the transaction history; (3) The next best solution is to encrypt data before sending it to the blockchain. At this stage, AES encryption is being used, where processed and formatted data is fed to the function handling encryption. After this stage, the transaction is built and ready to be signed and sent to the network. Only general information of student is visible, and other sensitive information has been hidden. The only way to decrypt this sensitive information is to use the key, which was encrypted with.

By using an encryption key, we can offer data integrity into the network. Keeping the system rather simple to use, we consider several ways of sharing this encryption key with the student. Firstly, the simplest way is through a printed certificate format, where a QR code with the encryption key is embedded with it. Secondly, considering that universities maintain contact with their students by establishing a connection through phone number and email, this information can be used for the key distribution. Upon certificate deployment, a SMS or an e-mail is sent to the student notifying them about the status of their certificate and the private key.

If the encryption key is lost, data is also permanently lost. This means that the information of the certificate is stored on the blockchain, but it is meaningless, because it cannot be decrypted. However, the certificate deployments are periodically saved into local computers of the university. Thus, with a student request, an extract copy of the certificate can be redeployed, or a new version if needed. The only downside of this method is that the student must manually request a redeployment or encryption key of the certificate.

University Application Interface

So far, a basic interface has been created, as it is shown in Fig. 2, where the data is formatted and encrypted. Later a transaction is built, and after signing it with the private key, it is sent to the network. There is a config file called config.js responsible for the address used to sign the transaction, although a setting interface is planned to be developed, which will include other configuration options

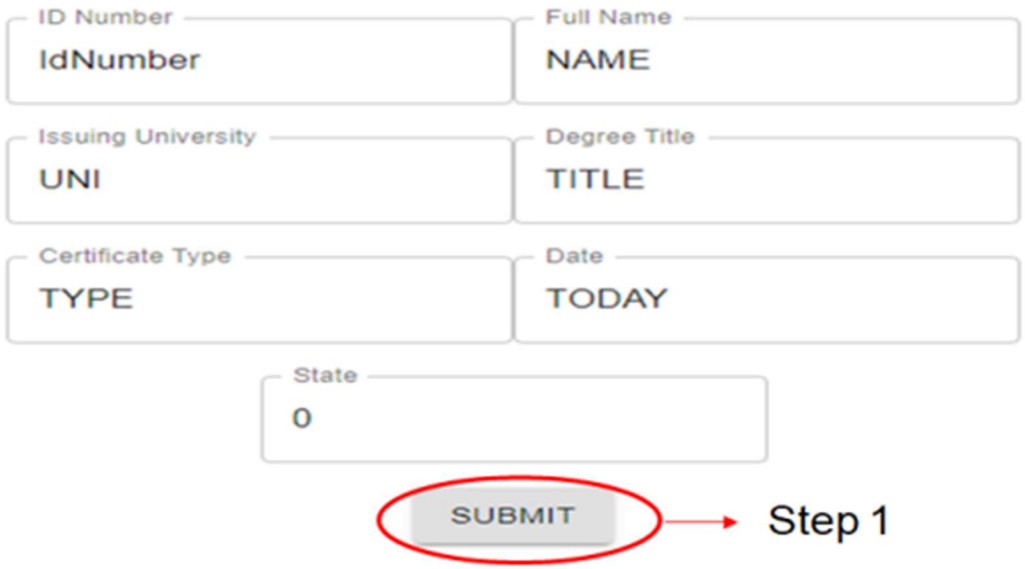


Figure 2. University Interface

Figure 3. Entering Encryption key and signing the transaction

Phone:+1 2177412411
Fax : +1 2243861099
www.bctsi.com
info@bctsi.com
617 W Parkside Dr
Palatine IL 60067

After submitting the form, users are required to enter their unique encryption key. This key is used to encrypt the data, allowing only authorized person to access the academic records. The encryption key will be different for each certificate. The next step requires to sign the transaction using a private key of the issuer university, which should be kept secret as it is presented in Fig. 3. It is university's responsibility to add certificates to the blockchain, validate/invalidate them if needed, to store encryption keys and logs in a private server as well.

Verification Application Implementation

The verification application is supposed to verify whether the issued certificate is valid or not. On top of that it should show a set of basic student details, such as Student ID, Date of issuance and Certificate status. These details are supposed to show as little information as possible, while maintaining the privacy of student. In case additional information is needed, it should be up to the student to allow access to this kind of information. This means that personal information should be somehow hidden from others and at the same time publicly accessible when needed. Such information might be: Full Student Name, University, degree, certificate type, and graduation date. The student is the one who allows access to this kind of information by providing the passphrase, which is the encryption key, that university issued to him/her by email, when the certificate was created. This key must be kept secret and in case the student loses it, the university has the responsibility to generate another one. Furthermore, as an open-source Ethereum application, it makes the process of verification more transparent and trustworthy. Every employer/verifier should download the application from trusted sources and use it only on trusted networks. To conclude, this application would be intuitive, easy to use and transparent.

The Verifier Application

This version of the application depends only by one contract, which is the 'storing certificates' contract. This contract is deployed on our test environment and after that we can start developing the verification application. To make development easier, an example certificate is deployed by default with the following details as are shown in Fig.:

The private data has been encrypted using AES and crypto-js library and can be decrypted using the same. But to decrypt, we need a decryption key, which will act as a kind of password. For this set of data, it is “SomerandomPassPhrase” and the data decrypted: FullName, University, Degre, CertType, Date.

```
<TextField
        type='text'
        label='Certificate Serial Number'
        name='certID'
        variant='outlined'
        onChange={(e) => setCertId(e.target.value)}
/>
```

First, a connection with the contract should be established. This is done by creating a new component called ‘ConnectC’ which makes this possible. This component handles the configuration including smart contract address and network location, as well as connecting to the network itself.

The only module being used up to this point is Web3, which uses HTTP, ICP/WebSocket to allow us interacting with a local/remote Ethereum node.

Following, we need to make use of this connection. We import this component into the main App component and use it in conjunction with a form element. This way we can get all the data given the certificate serial number. We make use of React State features to store and update this information.

This piece of code takes the entered value (in our case the certificate serial number) and passes it to certID which stored the current certificate ID (also referred as serial number).
This function takes the submitted serial number to the ConnectC component and sees the returned certificate data to cert which is used to show unencrypted data.

```
function handleSubmit(event){
        event.preventDefault();
        ConnectC(certId).then((r) => {setCert(r)});
        setDecData([]);
}
```

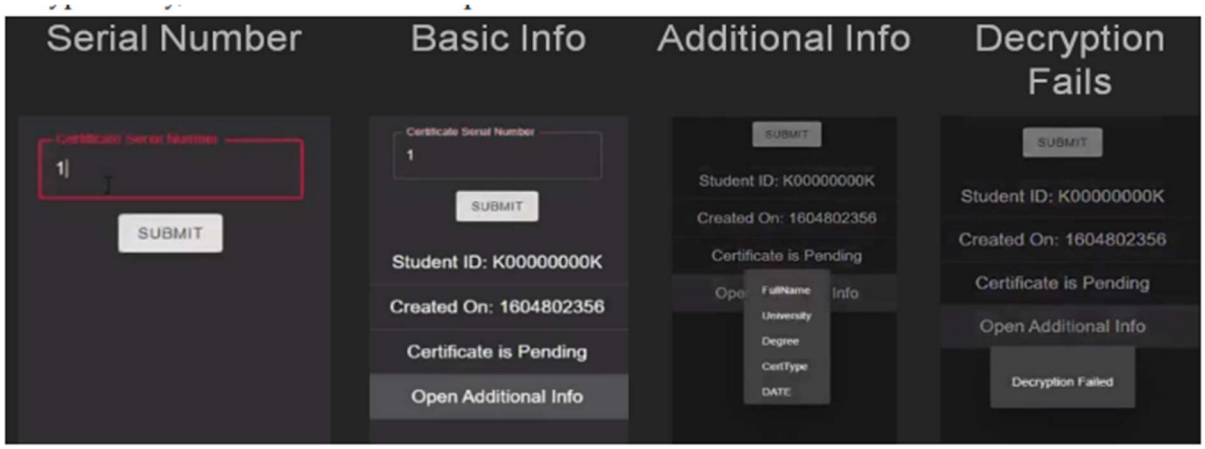
The Crypto-js module is used to encrypt data with AES encryption algorithm. The same module and encryption will be used to decrypt the data. Specifically, once the user requests additional data, he is prompted to enter a passphrase (decryption key). The following line takes care of decrypting the data:

```
AES.decrypt(encData, pass).toString(Utf8)
```

Furthermore, this piece of code takes the entered value (in our case the certificate serial number) and pass it. If the decryption is successful, the data is displayed on screen. Additional modules and libraries such as Material-Ui have been used to make the application more visually appealing, although at this point not much has been done regarding UI/UX.

Verifier Interface

The verifier interface will be used to verify whether the certificate is valid or not. It should be easy to access on most type of devices. Once the verification application is open, the interface is as in Fig below , presented below. After the form has been submitted with the certificate serial number, a basic information regarding the contract is displayed, such as student ID, date of certificate's creation and status of the certificate. For further details, the student can use the last button 'Open Additional Info', which requires decryption key, to enable the user view additional information related to the degree, university that has issued that certificate, date of creation and so on. If the student enters a wrong decryption key, certificate verification process will fail.



At this point in time, this application performs basic functionality by manually entering serial ID. A QR code scan feature is planned to be implemented, thus making the process easier than it currently is. Some extra information will be added and removed from 'Basic information' and 'Additional' windows, as well as adding a window for IPFS photo ID. Additionally, this application must become more trustworthy by implementing several warnings and following certain protocols to raise awareness about phishing attempts and clones.

Testing and evaluations

Due to the complexity and the type of data being handled by the project, a significant testing time will be required. This is because data being stored into the blockchain is immutable, and once those data are in the network, they cannot be removed. Furthermore, we need to make sure there are no missing spaces, security holes or loops in function execution. At the same time, this application needs to be publicly available, and transactions easily verifiable. Other things that will be tested are the costs of transaction, certificate deployments and action costs. For this reason, and more, a set of requirements must be met to publicly release the application. Some of these requirements will depend on the approaches and optional technologies chosen by the application developer.

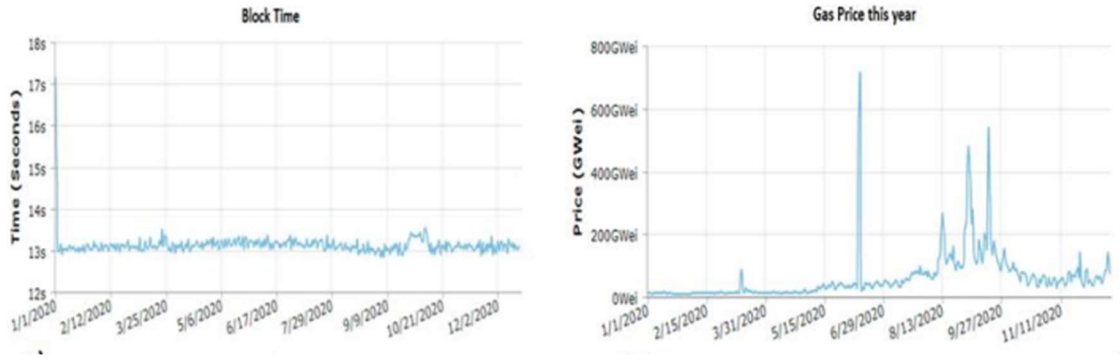
Early test versions regarding security and core functions will be deployed on a private blockchain network. This enables us to quickly revert changes, monitor costs and raw data transfers. Moreover, every data transacted in this stage will be completely private and core functionality will be prioritized. This means the UI/UX might be far from the final product, and only a small group of people will have access to this early version.

Next stage consists of deploying the product to a public test network. At this stage core functionality should not have any issues or bugs. The focus of this stage is to bring the interfaces to an intuitive state, where users will have no problems realizing how to use the application. Additionally, several error trackers, statistical and feedback handlers will be made, allowing us to fine tune the application, ideally making it error free and easily upgradable, and prioritizing the enhancement of current features. Means of gathering information during testing:

1.A closed feedback system consisting of tester-developer communication.

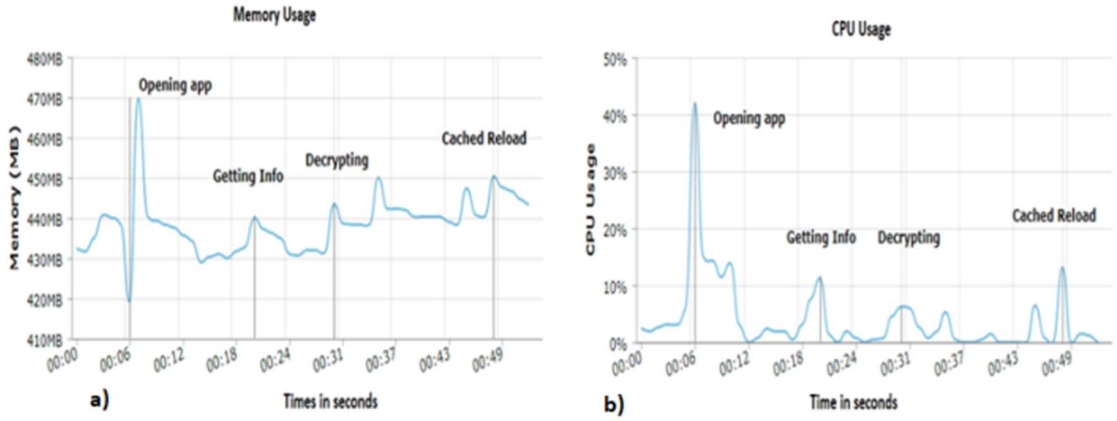
2.Several costs, transactions, data, usage information, which are monitored by using appropriate software and means. Such software will depend on the developer of the application.

The first graph in Fig., shows the time taken for a block to be included into the blockchain network. This is only for the past year and depends on global factors. The second graph shows the gas price in gwei. Gas price is another factor we cannot affect. It depends on global factors and the whole network status.



a) 'Block Time' and b) 'Gas Price' of Ethereum platform for year 2020 (Data Source from EtherScan.io)

Figures below will present CPU, Memory and Network usage, using SysGaugeMonitor. Those tests has been performed on a 3.4GHz Core on Windows 10. The memory usage, from the time (in seconds) the application is opened, until it is closed, is presented in Fig.. As seen, the memory usage at the end starts declining. Should be noted this benchmark also includes a chromium browser which is required to run a PWA, and will greatly differ from other devices, and future versions



a) Memory Usage and b) CPU usage of Application

CPU usage is shown in Fig..b. Values may be different on other devices as well, however as this graph shows, the application can run smoothly on low-end devices, given it fits the minimum requirements of the final product.

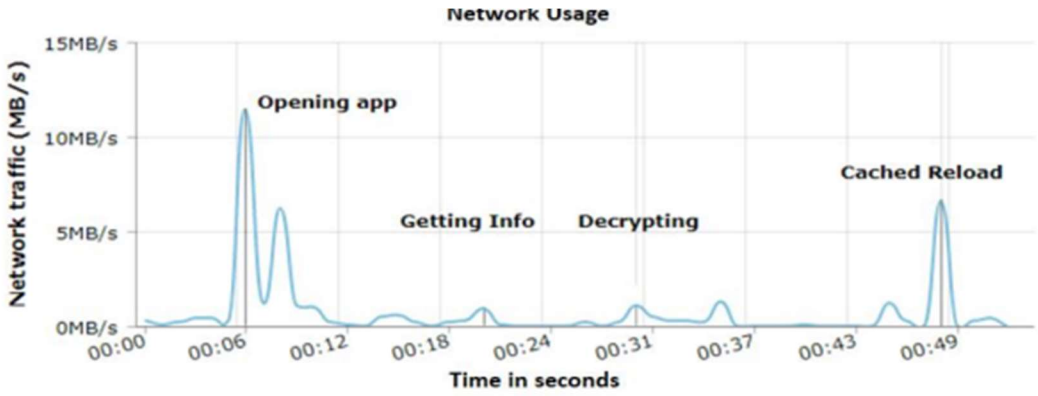


Figure :Verification Application Interface

The graph in Fig. represents the network usage this application utilizes. Due to the small size of the application, this is relatively low and barely measurable. A real difference can be seen on very slow devices or network connections.

The above graph in Fig.shows the cost in gas in order to perform an operation. These metrics include a single execution. Sequential and parallel executions will have different outcomes depending on their combination. Converting this to fiat currency depends on gas price, which is affected by global factors. The etherscan APIs tool can specifically be used to make the convertation to fiat currency.

Fig. shows the ability to parallelize requests and add certificates in bulk. This part of the graphic highly depends on the first two graphics, as blocktime and gas price are the main factors that determine the time for the transaction (certificate addition) to be confirmed. We have taken the best (highest and fastest) price at the time of the benchmark. (according to Etherscan gastracker API).

Adding a small number of certificates requires almost no parallelization, however the more we add, the heavier the parallelization becomes. In this example we are adding 6 certificates at a time through 10 accounts simultaneously, which adds up to 60 certificates at once. This will require finer tuning to get right. We specify a block time on our test network close to mainnet with fast gas price preset. Then we issue certificates on this network. This number is expected to improve, however it will be up to the university to decide (through a config panel). Should be noted that the more data we are adding, the more expensive the process becomes, and should be careful this process does not exceed the account balance limit.

Future Development

This section presents the future development of the proposed implementation regarding verifier, university, and accreditor applications as follow.

Verifier Application: On the verifier application, the user currently needs to manually enter a serial number and encryption key. Future includes adding a QR code scanner directly into the application, and the student can present a QR code to the verifier, thus the verification process will be instant without having to manually enter a serial number or passcode if preferred. Another plan is to add features to increase application usage awareness, so that everyone gets to know how to use it properly and not get tricked by off-market fake applications or clones.

University Application: The next most important feature to be added is a configuration panel, where each university can tweak the application settings according to their way of operating, such as the way the encryption keys are distributed, custom gateways, a history of deployed certificates with their respective state (pending, confirmed, declined), transaction hash and other important details. On top of that, depending on the parallelization technique used, the university might change the maximum number of certificates added at once to fit their requirements. Additionally, it is planned an import/export file, thus the certificates can be added in bulk. Furthermore, these certificates can be generated and signed on different computers. An example would be that a certificate is generated on personal computers, which may belong to authorized personnel, and can be later signed and deployed on the computers belonging to the university. The third feature planned is the way to distribute the encryption key, this will depend on how the universities establish the connection to the students, however an email distribution should be the default.

Accreditor Application: Updates on this application are similar the university application, they will include a configuration panel, import/export feature, and a distribution method. A unique feature is the ability to add multiple accounts per university, so each university can use multiple accounts to add certificates. An important feature would be the ability to invalidate or suspend universities, in case of an ongoing investigation or other reasons