

Decentralized DigiCerts user manual

1. Software Installation

1.1 Prerequisite softwares

Decentralized DigiCerts is a server application. Hence NodeJs has to be installed in the machine on which the server is running. Instructions to install NodeJs are available at <https://nodejs.org/en/download/>

1.2 Installing Decentralized DigiCerts

Once the NodeJs is installed, following are the instructions to install the Decentralized DigiCerts Software*.

1. Unzip the **blockchain_based_cert .zip** file
2. Enter the 'blockchain_based_cert' directory. **cd {PATH TO 'blockchain_based_cert' directory}/server_app**
3. Do **npm install** to install the software and its dependencies.

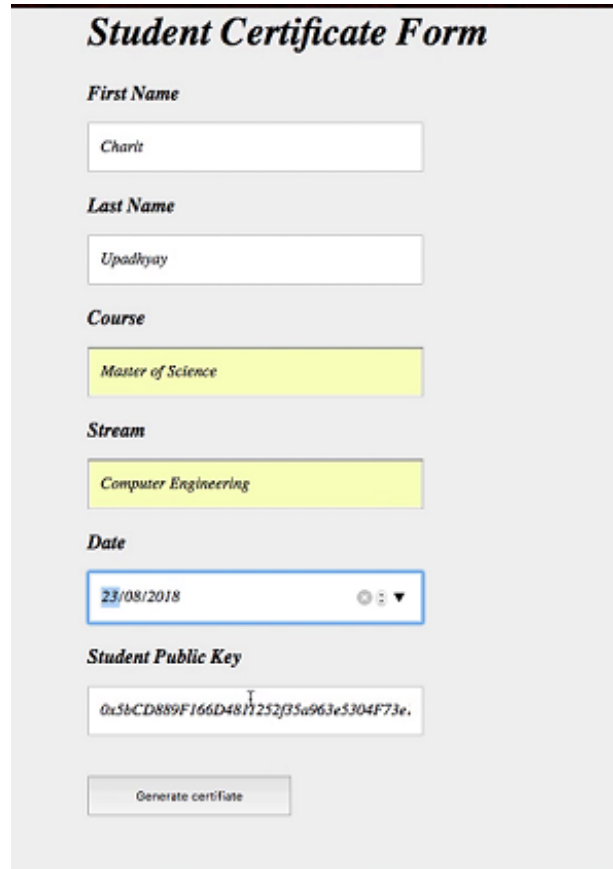
2. Software Usage

2.1 Starting the server software

1. Enter the 'blockchain_based_cert' directory. **cd {PATH TO 'blockchain_based_cert' directory}/server_app**
2. Do **npm start** to start the server app which will now listen on port 3000 for handling certificate issue requests.

2.2. Issuing the certificate


Certificates can be issued using web based front-end. In order to issue the certificate, use the web browser to access the URL of the digicerts server. In the browsers address bar, use **http://{IP of digicerts server}:3000**. For example, if the server is local, the command <http://localhost:3000> can be used. A HTML form as shown below will be rendered as a response by the server,



The image shows a web form titled "Student Certificate Form". It contains several input fields: "First Name" with the value "Charit", "Last Name" with the value "Upadhyay", "Course" with the value "Master of Science", and "Stream" with the value "Computer Engineering". There is a "Date" field showing "23/08/2018" and a "Student Public Key" field showing a long hexadecimal string. At the bottom, there is a "Generate certificate" button.

Figure 1: UI to generate certificate

where the issuing institution can key-in the required details and click on 'Generate Certificate' button to generate a certificate with embedded verify functionality as shown below.



The image shows a digital certificate from San José State University. The text on the certificate reads: "San José State University have conferred upon Charit Upadhyay the degree of Master of Science in Computer Engineering with all the rights and privileges pertaining thereto Given by the Trustees of The California State University at San José this twentieth day of December, two thousand ten." Below the text are two signatures and two circular seals. At the bottom, there is a "Contract Address:" field with the value "0x5bCD889F166D4811252J35a963e5304F73e10b66", a "Student public key:" field with the value "0xf50614e98dAeEBaF0b522d0b8e4a85Fe3F8033A7", and a "Verify Certificate" button.

Figure 2: Certificate with embedded verify functionality

2.3 Verifying the certificate

The certificate generated using 'Decentralized digicerts' system can be verified by clicking the 'Verify certificate' button. The verification involves computing the hash of the certificate, and comparing it with the blockchain hash. If they match the certificate is considered to be valid.

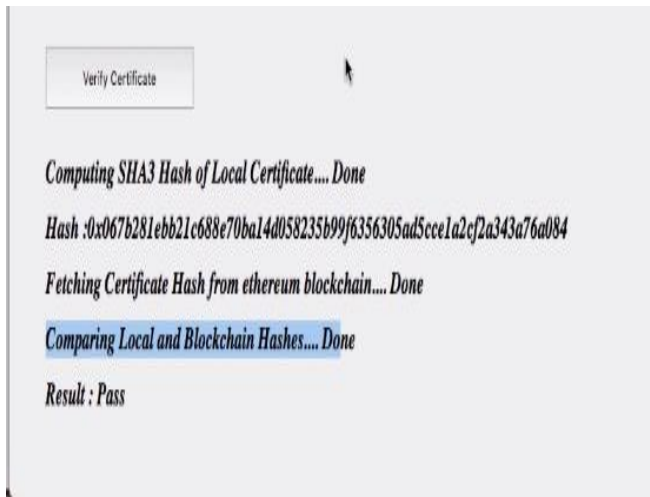


Figure A3: Certificate verify demonstrated.

2.4 Note on ethereum accounts

In order for 'Decentralized DigiCerts' to work following are the Ethereum accounts and addresses that are needed.

1. **Smart contract address:** Address of the contract owned by the issuing institution to maintain the record of credential owners and their certificate hashes. Currently we provide a pre-deployed contract for use by the issuing institution. The contract address is **0xf50614e98dAeEBaF0b522d0b8e4a85Fe3F8033A7**
2. **Issuer public and private key:** Issuer public key is the public key of the issuing institution and private key is used to sign the certificate issue transactions. Currently the issuer is given a pre-created account of public-private key. The issuers public key is **0x561da499e78486727bfeec2992c121c16724575f**
3. **Credential receiver public key:** Public key of the credential receiver, and can be keyed in the certificate creation form

As of now all the deployments and transactions are done in a test ethereum network called ropsten. The transactions can be viewed through the URL:

<https://ropsten.etherscan.io/address/0x561da499e78486727bfeec2992c121c16724575f>