Issuing and Verifying University Certificates on Blockchain

Dhiren Patel, Balakarthikeyan Rajan, Yogesh Mangnaik, Jatin Jain, Vasu Mistry, and Pearl Patel

Abstract In this paper, we propose a decentralized approach toward issuance of institute degree and their verification system using blockchain technology. Keeping the encrypted record on a public blockchain, using Oracles, we provide a framework for quick verification of the degrees. The system introduces the concept of generating a single root hash for all the degree-encrypted hashes to avoid excessive on-chain data storage. The system also explains the different smart contract workflow required to implement the idea along with the use of ERC223 token to monetize the Oracle service.

Keywords Blockchain · Merkle Tree · Smart contract · Verification

1 Introduction

Traditional security has been ensured through access and control with an institution in the center which may exploit and misuse trust of clients and users. Blockchain is a technology which decentralizes the access and control mechanism thus avoiding central points of trust. Blockchain is able to provide a tamper evident system to ensure data security.

In this paper, we present a system of issuing and verifying degree certificates by university/institute using blockchain and show by using Oracles, a way to decrypt the encrypted certificates on the blockchain thus allowing a method to monetize the entire system. Use of decentralized blockchain and smart contract will improve the

D. Patel · B. Rajan (✉) · Y. Mangnaik · J. Jain
Veermata Jijabai Technological Institute, Mumbai 400019, India
e-mail: bkybala9@gmail.com

V. Mistry
Independent Researcher, Mumbai, India
e-mail: vasu5235@gmail.com

P. Patel
Vidyalankar Institute of Technology, Mumbai 400307, India

D. Patel et al. (eds.), *IC-BCT 2019*, Blockchain Technologies,
https://doi.org/10.1007/978-981-15-4542-9_8

entire process of issuance and verification of degree certificates both on performance (time) and security (tamper proof).

Rest of this paper is organized as follows: Sect. 2 discusses basic improvements existing in certificate issuance system. In Sect. 3, we present a workflow for such system on blockchain. Section 4 discusses verification system for issued certificate. Section 5 discusses implementation architecture, with conclusions and references at the end.

2 Motivation and Background

2.1 Existing Degree Certificate Issuing and Verification Procedure

The existing system for certification issuance for the graduates and its validation is a long and cumbersome process. This paper was motivated by analyzing the current scenario in typical university/institution and is an attempt to apply blockchain technology for improving the current workflow. Most universities/institutes offer hard copies of degree certificates to their graduates. This makes verification during background checks cumbersome and typically requires the institute to check the copies and send back a signed copy attesting that these certificates are valid. A typical workflow is depicted in Fig. 1. Institutes also generally charge some money to process such verification requests and on an average, this process might take one to three weeks. To alleviate this problem, we propose a decentralized solution using blockchain technology to allow verification of such certificate data. This also means that now student certificates are uploaded onto the blockchain network. This gives the added benefit that students can now supply their certificates or educational record for verification without the use of any hard copy. Such verification can be done seamlessly via smart contracts which will also charge a small fee for verification. The smart contract provides a secure way to verify whether the given submitted certificate is genuine or not and collects some fees in the form of tokens as payment for this service.

2.2 Blockchain

Blockchains are distributed digital ledgers of cryptographically signed transactions that are grouped into blocks. Each block is linked to the previous one after validation and consensus of all participating nodes. As new blocks are added, older blocks become more difficult to modify. New blocks are replicated across all copies of the ledger in the network, and any conflicts are resolved automatically using established rules [1].

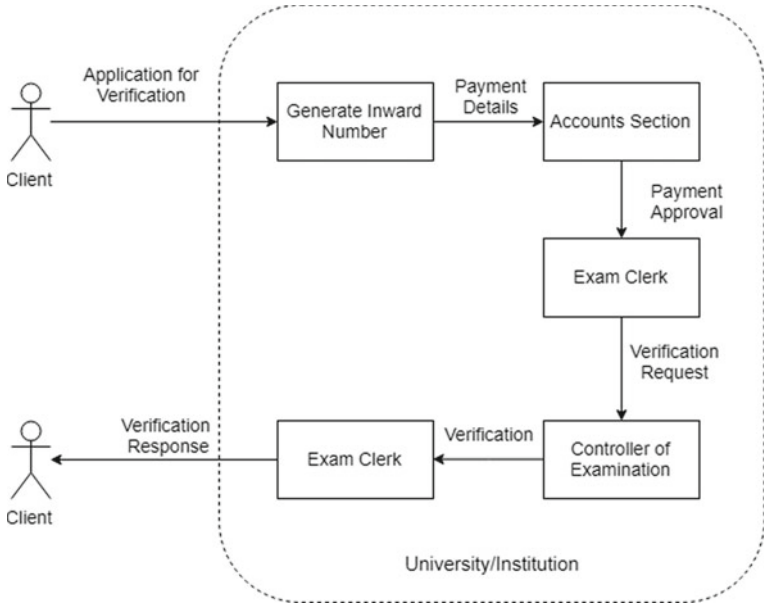

Fig. 1 Typical workflow for degree certificate verification in university/institute

At their most basic level, blockchain enables a community of users to record transactions in a ledger that is public to that community, such that no transaction can be changed once published. A block is an individual unit of a blockchain, composed of a collection of transactions and a block header. A block header keeps a collection of metadata about the block that contains a hash-value of its parent in the blockchain, and a hash of the aforementioned metadata and the data of the block itself [2].

A public blockchain network is completely open and anyone can join and participate in the network. The network typically has an incentivizing mechanism to encourage more participants to join the network. Ethereum is a turing complete programmable public blockchain platform [3].

2.3 Ethereum Public Blockchain

Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology. Ethereum is a programmable blockchain. Rather than giving users a set of predefined operations (e.g., bitcoin transactions), Ethereum allows users to create their own operations of any complexity they wish. In this way, It serves the purpose for many different types of decentralized blockchain applications, including but not limited to cryptocurrencies. Ethereum Virtual Machine ("EVM") can execute code of arbitrary algorithmic

complexity. In computer science terms, Ethereum is "Turing complete." Developers can create applications that run on the EVM using friendly programming languages modelled on existing languages like JavaScript and Python [4].

A contract is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. Contract accounts are able to pass messages between themselves as well as doing practically turing complete computation. Contracts live on the blockchain in an Ethereum-specific binary format called Ethereum Virtual Machine (EVM) bytecode, and contracts are typically written in some high-level language such as Solidity and then compiled into bytecode to be uploaded on the blockchain.

2.4 Merkle Trees and Merkle Proofs

A tree constructed by hashing paired data (the leaves), then pairing and hashing the results until a single hash remains, the Merkle Root. Merkle Tree is a tree in which every leaf node is labelled with the hash of a data block and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes. Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains. Demonstrating that a leaf node is a part of a given binary hash tree requires computing a number of hashes proportional to the logarithm of the number of leaf nodes of the tree (Fig. 2).

Merkle Proofs are established by hashing a hash's corresponding hash together and climbing up the tree until you obtain the root hash which is or can be publicly

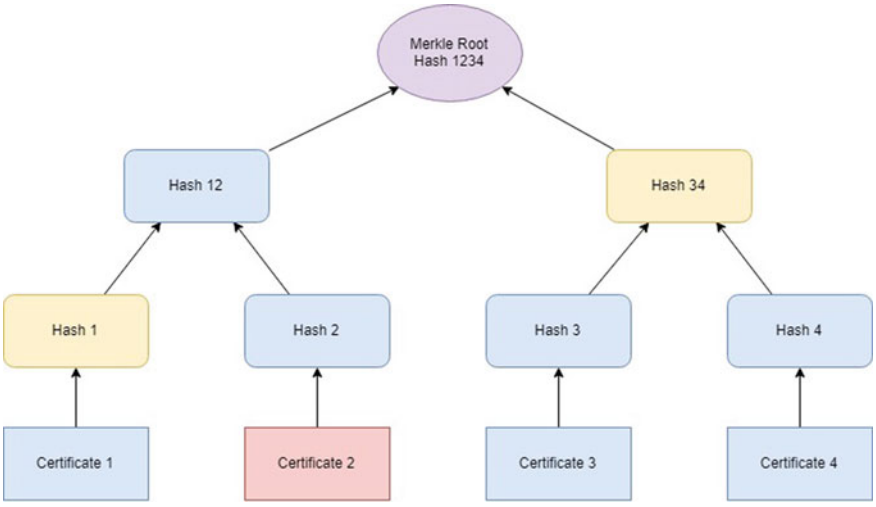

Fig. 2 Merkle Tree

known. Given that one-way hashes are intended to be collision free and deterministic, no two plaintext hashes are the same [5].

Assuming such a hash-function which we shall call as H, then

$$\begin{aligned} \text{Hash 1} &= \text{H (Certificate 1)} \\ \text{Hash 2} &= \text{H (Certificate 2)} \\ \text{Then Hash 12} &= \text{H (Hash 1 | Hash 2)} \\ \Rightarrow \text{Hash12} &= \text{H (H (Certificate1) | H (Certificate 2))} \end{aligned}$$

Merkle Proofs are used to decide upon the following factors:

- If the data belongs to the Merkle Tree.
- To concisely prove the validity of data being part of a data set without storing the whole data set.
- To ensure the validity of a certain data set being inclusive in a larger data set without revealing either the complete data set or its subset.

2.5 Oracle

An oracle is an agent that finds and verifies real-world occurrences and submits this information to the blockchain to be used by smart contracts.

Oraclize service allows smart contracts to connect and obtain results from different services like IPFS, WolframAlpha.

- oraclize_query("IPFS", "<file-hash>") will retrieve the contents stored on the IPFS network
- oraclize_query("WolframAlpha", "flip a coin") will return either "heads" or "tails" to the smart contract [6].

Oracle allows smart contracts to interact with the outside world using APIs.

Oracalize.it [6] is a service and library for Solidity which provides Oracle services. Oracles attest for the proof of incoming information from outside a blockchain. The oracle can be thought of as a gateway which provides secure communication between a blockchain and the rest of the Internet.

Today, most universities/institutes issue paper certificates to students who have passed their offered courses. Issuing paper certificates involves a properly designed process to ensure certificates are not temperable and are not illegally duplicated. Also one has to ensure an easy way to identify and attest to the validity of a certificate. That is, we need to authenticate and validate certificates. Many digital courses allow online validation of issued certificates to check their genuineness. The cost for this in terms of infrastructure and facilitation is passed onto students. Companies also spend time and money themselves or third-party agencies to verify authenticity of students joining them and the validity of their degrees. The system proposed in this

paper aims to address the problems by making these certificates available on a public blockchain and writing appropriate smart contract codes to publish and verify these certificates. The verification process can be built in such a way that a small fee would be needed to be paid by the requester to assess the genuineness of the certificate. The certificate now can be just given as a small string to the holder (a hashed certificate) rather than a pdf document. This would make the verification process transparent, trustable, and faster.

One such system has already been implemented by the Massachusetts Institute of Technology [8]. The existing system allows verification code to be run independently by any party and provides a sample code and system to be based on. In this paper, we explore the creation of such a system where verification process and certificate issuance are delegated to a smart contract on the Ethereum network and an Oracle service. This allows institutes greater control on the process and can design their own verification algorithms, and also enables institutes to charge fees as in the former system for verification requests. We also try to achieve this with minimal input from the side of the requester and the student. Since the system using encrypted hashes to construct the Merkle Tree, an Oracle service is used to get the encrypted hash. Thus, this system now ensures that there is a fee charged to the requester for the verification by the use of encrypted hashes and decrypting them using oracles

3 Workflow for Degree Certificate Issuance System on Blockchain

3.1 Digitizing the Certificates

Certificates need to be stored in a convenient digital format. In this case, we shall create a simple JSON schema to represent the certificate.

The following is a representative schema for the certificate

```
.certificate {
full_name: String,
institute: String,
cgpa: Double,
student_id: String,
batch: Int
hash: SHA256
proof: List of SHA256
}
```

The hash field here is populated as a SHA256 of full_name, institute, cgpa, student_id, and batch only. Let us call a subset of the above fields as a minimal certificate. The following minimal certificate is then hashed to get hash H. The H is re-hashed with the private key of the institute to get an encrypted hash E. This procedure shall

now be repeated for every certificate making up a batch. For example, all certificates for the Batch of 2017 shall be generated and for each certificate, the hash E shall be computed as described above. After this, a Merkle Tree is generated for the entire batch and its Merkle Root M is found out in the next steps. The proof field includes the Merkle Path of these encrypted hashes which will be used as a check to see whether the certificate has been included by the Merkle Root or not.

3.2 Generating Certificate Blocks

Once the Encrypted Hashes E for every certificate is known, each of them can be directly put on to the blockchain. But this will involve a lot of excess computation and costs, each transaction costs gas fees and thus, one/the institute would end up spending a lot of money due to the fact that there are many certificates to be published. To overcome this, we use all of these encrypted hashes to create the Merkle Tree. The root of this tree called M, is known as Merkle Root and it is representative of the elements which made up this root. Specifically, Merkle Paths allow us to create a Merkle Proof where in knowing the Merkle Root, the corresponding Merkle Path to a leaf and the data of the leaf, we can verify whether the leaf was part of this root or not. Thus, we can securely say that the hash (i.e., whether the certifcate) was part of the root or not. Thus, now our transaction is reduced to only one, which is a single transaction containing the Merkle Root can be published. We shall now add this Merkle Path to the proof field of our digital certificate. Once every certificate has its proof field added, we can email this to each candidate. This ends the process from the side of the institute.

3.3 Publishing the Certificates on the Blockchain

The Merkle Root generated will now be added to the blockchain. To enable this we call a function on our smart contract with this Merkle Root as the data. The contract will accept this transaction and fire an event; tagging this transaction if and only if the caller is the contract deployer which in this case will be the university/institute. This published transaction will be mined by one of the miners on the ethereum network and included in some block that will be added to the blockchain. As the time goes on, more blocks will be mined and included in the blockchain. As more blocks are appended to the block containing the published transaction, the harder it is to change it. There is a certain number of blocks after which the transaction is considered to be immutable. Every block appended is considered to be a verification for the previous blocks.

4 Verification of Issued Certificates

We shall provide a simple smart contract function which shall return a Boolean value true or false depending on the status of the verification. The verification process follows the following steps. Before submitting the hash for verification to the smart contract, we run a small client-side check to ensure that the hash field in the certificate matches the hash of the certificate. The contract will receive only the payment tokens, certificate hash field, and the Merkle Proof field.

1. The smart contract first finds the transaction associated with this batch, since every batch had a different Merkle Root.
2. The smart contract verifies the Merkle Root transaction's issuer's identity. It checks if the transaction containing the Merkle Root was signed by the private key of the institute/university.
3. The contract requests the Oracle service for the encrypted hash for the certificate hash given as input.
4. Next, the contract builds the Merkle Root using the path info and checks whether this generated Merkle Root is same as the one in the transaction. This finally ensures that the certificate was part of the Merkle Root (Figs. 3, 4, 5 and 6).

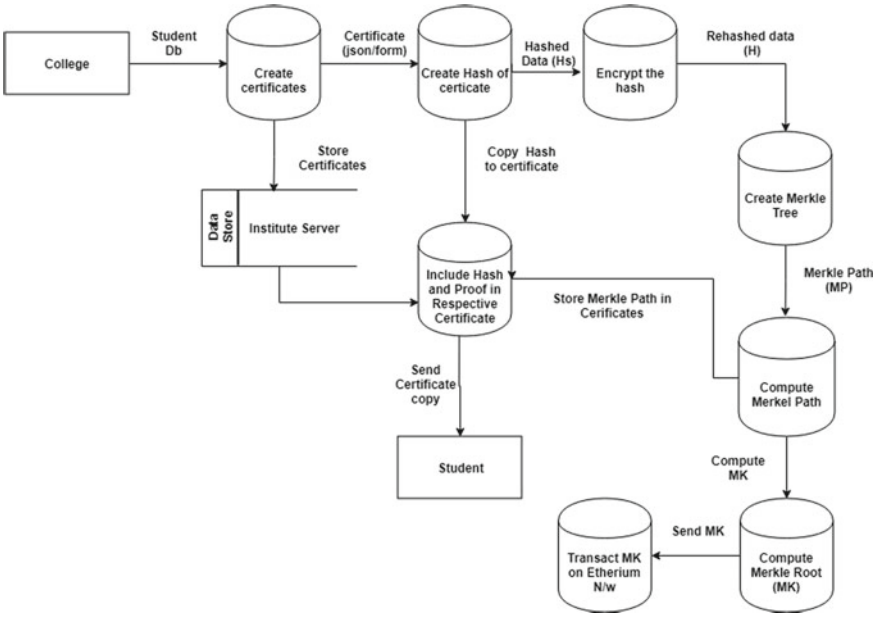

Fig. 3 Data flow diagram showing issuance of the certificate by the institute/university

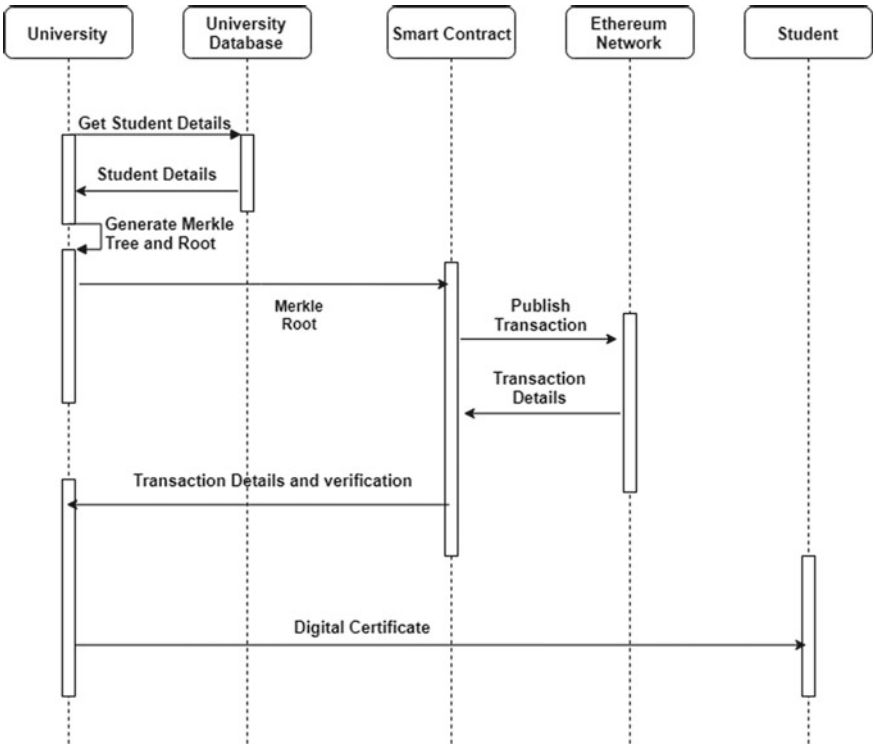

Fig. 4 Sequence diagram showing issuance of the certificate by the institute/university

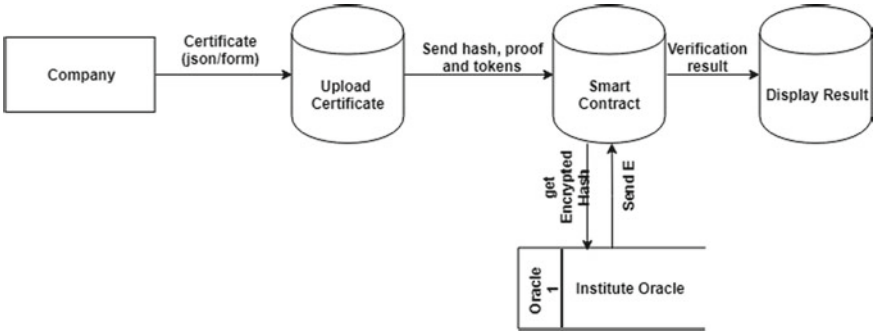

Fig. 5 Data flow diagram showing verification of the certificate by company

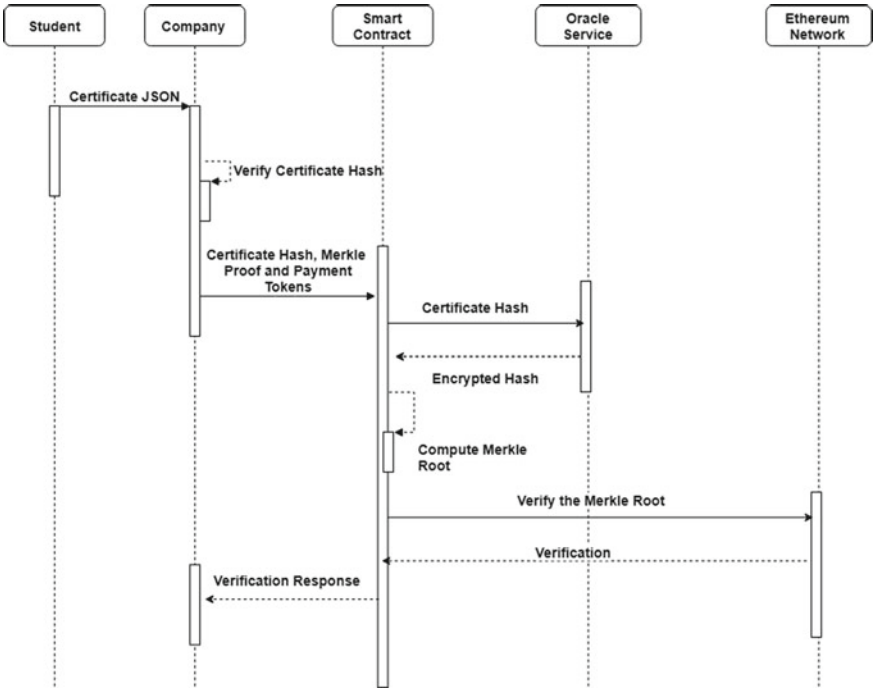

Fig. 6 Sequence diagram showing verification of the certificate by company

5 Implementation Architecture

5.1 Smart Contracts

The deployment will include smart contracts to facilitate publication of the certificate to the blockchain and also the verification of published certificates. The ***publish*** function would take the signed transaction with the Merkle Root and record it publically. This function can be only called by the contract deployer that is the institute/university can only publish the certificates. A mapping will be recorded between each batch and the transaction containing its Merkle Root allowing easy access to that year's certificates.

The second function ***verify***, shall be accessible to any verifier and accepts as fees a set of predefined tokens to carry out the verification. It shall return only a Boolean Yes/No and an optional message detailing failure if needed. This function shall accept the given certificate which shall contain the hash and the Merkle Proof to validate the authenticity of the certificate. It performs tasks as outlined in the verification steps. The tokens for the same can be pulled out of the client (verification request submitter) Metamask [9] wallet.

5.2 Server-Side Deployment

This is the only code which shall be maintained on the server by the institute. This code contains the service which is responsible for loading the form which the end-user will utilize for the verification process. It can also contain additional code to collect statistics and other data.

5.3 Payment Gateway

To monetize the process of verification of certificates, tokens are accepted by the smart contract before disclosing the results of such validation as payment. To generate these tokens, we shall deploy using the institute address ERC223 tokens [8]. A payment gateway shall accept money (real currency in INR) and instruct our deployment of the ERC223 tokens smart contract to issue equivalent tokens to the recipient address. This shall be a transaction called by the institute to the recipient. The tokens shall be reflected in the recipients Metamask wallet [7].

5.4 Oracle Service

We would need to implement an Oracle service which listens for an event and returns the corresponding encrypted hash for the given Hash. This service can be implemented using Oracalize.it [6]. The Oracle will prevent the requester from directly knowing the encrypted hash (Fig. 7).

The architecture is implemented in a way so as to allow the university/institute to maintain control over the process of verification and monetize it in a way it deems to be fit. It is also made in a way to store least amount of information at the institute's end. It also minimizes processing at the servers maintained by the institute. Thus, the architecture ensures that minimal trust is put on the issuing university/institute and allow for a decentralized verification of certificates.

6 Conclusions

With the rising increase of educational certificate fraud and misuse, it becomes imperative to design an easy to use trustless, decentralized validation system to verify authenticity of certificates and to make student digital certificates tamper proof in nature. We have looked at an interesting use case and implemented a solution on the Ethereum public blockchain to ensure tamper-proof certificate issuance and to verify their authenticity. This also reduces time and efforts spent by the institute in

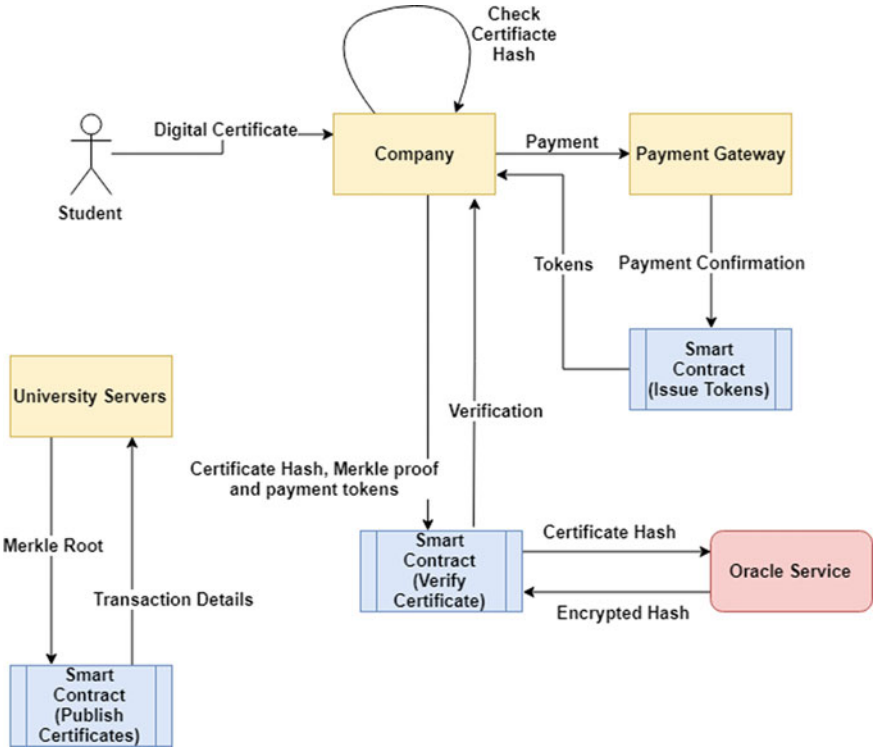

Fig. 7 Implementation architecture diagram for certificate issuance and verification system

verification of their certificates while still allowing them to monetize the system. The system also ensures that minimal trust is needed on the institute/university for verification of the issued certificates.

References

1. Yaga D, Mell P, Roby N, Scarfone K (2018) Blockchain technology overview. Draft NISTIR 8202, NIST, US
2. Wurster S et al (2017) Specification on Blockchain Technology. ISO/TC 307, Tokyo
3. Ethereum Project. https://www.ethereum.org/
4. Becker G (2008) Merkle signature schemes, Merkle trees and their cryptanalysis. Ruhr-University Bochum, Technical Report
5. Oraclize-blockchain oracle service, enabling data-rich smart contracts. http://www.oraclize.it/
6. Digital Certificates Project. http://certificates.media.mit.edu/
7. MetaMask Ethereum Browser Extension. https://metamask.io/

8. Ethereum: ERC223 token standard—Issue#223—ethereum/EIPs. https://github.com/ethereum/EIPs/issues/223
9. Cheng JC, Lee NY, Chi C, Chen YH (2018) Blockchain and smart contract for digital certificate. 2018 IEEE International Conference on Applied System Invention (ICASI). https://doi.org/10.1109/icasi.2018.8394455