INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Verification of Ph.D. Certificate Using QR Code on Blockchain Ethereum

Nur Khairunnisa Noorhizam [a], Zubaile Abdullah [a,*], Shahreen Kasim [a], Isredza Rahmi A Hamid [a], Mohd Anuar Mat Isa [b]

[a] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, 86400, Malaysia*
[b] *iExploTech Sdn. Bhd., Cyberview CoPlace 1, Cyberjaya, Selangor, 42100, Malaysia*
*Corresponding authors: *zubaile@uthm.edu.my*

Abstract **— One of the major challenges the university faces is to provide real-time verification of their student's degree certification upon request by other parties. Conventional verification systems are typically costly, time-consuming and bureaucratic against certificate credential misconduct. In addition, the forgery of graduation degree certificates has become more efficient due to easy-to-use scanning, editing, and printing technologies. Therefore, this research proposes verifying Ph.D. certificates using QR codes on the Ethereum blockchain to address certificate verification challenges. Blockchain technology ensures tamper-proof and decentralized management of degree certificates as the certificates stored on the blockchain are replicated across the network. The issuance of certificates requires the use of the issuer's private key, thus preventing forgery. The system was developed using Solidity for the smart contract, PHP, HTML/CSS for the web-based implementation, and MetaMask for blockchain integration. User testing confirmed the successful implementation and functionality of the system. Users can add, update, and delete certificates, generate and scan QR codes, and receive instant verification feedback. The verification system effectively meets all requirements, providing a robust solution for validating Ph.D. certificates. Future research may focus on scalability and adoption, privacy and data protection, user experience, and integration with existing systems. Other researchers can optimize the verification system for widespread adoption and utilization by exploring these areas. This research contributes to securing and efficiently verifying academic certificates using QR codes on the Ethereum blockchain. Ultimately, this work advances the field of certificate verification and promotes trust in academic credentials.**

Keywords— **Blockchain; smart contract; Ethereum; decentralized application; certificate verification.**

Manuscript received 18 Oct. 2022; revised 14 Apr. 2023; accepted 22 Jun. 2023. Date of publication 10 Sep. 2023.

I. INTRODUCTION

Nowadays, every work requires a certificate; sometimes, the most required certificate for a certain work field is a diploma certificate, and other times it is a degree certificate. As a result, many people chose to fake one if they did nothave one [1]–[3]. Furthermore, there are several services accessible online for faking a certificate. However, this practice denies the opportunity of those who worked hard for their Ph.D. certificate.

Also, an education certificate credential is vital in today's increasingly competitive job market. Universities issue certificates to their students after fulfilling graduation requirements. However, the forgery of graduation certificates has become more efficient nowadays due to easy-to-use scanning, editing, and printing technologies. Therefore, certificate verification is an increasingly important and mundane task. Existing papers on certificate verification methods include a watermarking technique [4]–[6], 2D barcode method [7]–[9], QR code method [10], and web services [11] cross-check with serial number. These methods, however, have one or more issues, such as counterfeit certificates, single point of failure, high cost of the verification process, and time-consuming [12].

Recently, blockchain technology has been applied to solve certificate verification issues [7], [12], [13]. Blockchain technology allows universities and graduates to manage degree certificates over the distributed network using a blockchain-tamper-proof and decentralized structure. Certificates issued using blockchain cannot be tampered with because this certificate is stored in a block replicated across a blockchain network. In addition, when a certificate is issued to a blockchain, the issuer's private key is required; thus, the forgery of certificates is prevented. Due to the decentralized

property of the blockchain, it can also prevent a single point. Of failure [14]–[17]. Hence this research proposes the Verification of Ph.D. Certificate using QR Code on Blockchain Ethereum.

Some issues have been detected in the existing method of certificate verification. Firstly, there is always a time delay in the manual certificate verification method [18]–[20]. Sometimes, it took days, even weeks, considering the list of people who wanted to verify their certificates. That isprobably one of the reasons why some employers did not bother to verify their future employees' certificates. Thus, this problem can be solved by using blockchain Ethereum technology [21], [22]. On blockchain Ethereum, there is a program that is a smart contract. The user will interact with it by submitting the certificate and executing a function defined on the smart contract. Hence, therefore the process will be efficient and not time-consuming.

Next, the issue is when the user is required to upload the certificate [21], [23]–[25] instead of inputting the information about the user's certificate, such as the name of the institution, course name, and year of graduation. If there is a typo or incorrect information that is detected by the system, the user cannot edit the information. This makes the security of altering the information is weak. Hence, the proposed system will require the user to input every piece of information instead of uploading their certificate to avoid future problems.

Finally, their ways of keeping the certificate's record are ineffective [26]. Some systems did not encrypt or hash the information after entering the record. Furthermore, the data is not permanent. It indicates that third parties can modify the information in the record. However, with the suggested technology, which is Blockchain Ethereum, the data stored on the blockchain is immutable and safe. This is because all information stored on the blockchain cannot be altered, is irreversible, and hashed.

In Blockchain-Based Certificate Verification using Ethereum and Smart Contract study, the decentralized certificate verification application on the Ethereum Blockchain [18] is developed. It is being chosen since this blockchain technology is immutable, traceable, tamper-proof, encrypted, and more secure. The study stated that problems such as fake certificates and double spending could be eliminated using blockchain technology. The study used smart contracts to interact with the blockchain and ipfs to encrypt the hash value of each certificate that will be stored in the blockchain. According to the study, blockchain highlights existing issues relevant to educational institutes and finds appropriate blockchain features that could solve them.

In Certificate Validation using Blockchain [10] created an Android application to provide secure certificate verification. Graduation certificates and transcripts now contain information that can be easily tampered with illegally and made available to third parties, according to the study. As a result, an efficient mechanism that can ensure the information in such certificates is original, that is, the document originated from a reliable and authorized source and is not forged, is required. Aside from that, the study stated that using blockchain will provide more security and accurate and reliable information. The application that been developed operates in offline mode, allowing the certificate to be validated quickly.

In Proposing a Reliable Method of Securing and Verifying the Credentials of Graduates through Blockchain study, Reddy et al. [20] used blockchain technology to create a platform for storing and verifying student credentials in this study. When a certificate is added to a block, it will return a unique certificate ID, including the student Aadhar card number which serves as a primary key. Since the certificate ID is unique, both the student and the company will use to verify the authenticity of the certificate and whether the certificate provided by the student is genuine or not. Aside from that, each student's Aadhar card number will make it easier for the verifier to verify each student's certificate individually. Unfortunately, every time a certificate is added, the certificate authority must pay an Ethereum gas value. As a result, the miners will be able to contribute to the addition of blocks to the blockchain.

Xie et al. [27] proposed a scheme for developing a decentralized certificate system based on blockchain technology and smart contracts. The blockchain certificate system is being developed to provide blockchain certificate services for college students' innovation and entrepreneurship competition. The authors stated in this study that some of the certificate's functions are management, issuing, verification, and revocation, which are realized through smart contracts.

In A Blockchain-Based Verification System for Academic Certificates study, Gaikwad et al. [28] combined Blockchain and web-app to create a high-end product with a secure, fast, and reliable network, a powerful OCR, and an interactive interface. Once committed, the network on which the data is stored will keep the data intact. The verifier can begin the verification process by uploading the document, after which the OCR will work on it and extract the text data. Following that, the data will be hashed and stored in the blockchain. If any hash matches the hash of the data, the certificate is real; otherwise, the certificate is a forgery.

The novelty of the proposed system lies in several key aspects:

1) Adoption of Blockchain Technology: The research proposes leveraging blockchain technology, specifically the Ethereum blockchain, to manage and verify degree certificates. By utilizing a distributed and tamper-proof blockchain network, the system aims to enhance the security and immutability of certificates, making forgery more difficult.

2) Integration of Smart Contracts: The proposed system utilizes smart contracts, which are self-executing contracts with predefined rules and conditions. Smart contracts enable the automation of the verification process and ensure smooth operations without the need for third-party intervention.

3) QR Code Tagging: Each certificate is tagged with a QR code, enabling third parties, such as employers or other universities, to scan and verify the authenticity of the certificates in real-time. This provides a convenient and efficient method for verification, reducing the reliance on manual processes and paperwork.

4) Immutable Certificate Records: The research emphasizes the immutability of information stored on the Ethereum blockchain. Once a certificate is added to the blockchain, it cannot be altered or modified, ensuring the

integrity and reliability of the certificate records. This feature enhances the security of the system and reduces the risk of unauthorized modifications.

II. Materials and Method

This section will discuss the development of the web-based system and its component.

A. Blockchain Ethereum

Ethereum is a distributed public blockchain network that focuses on running the programming code for any decentralized application [29], [30]. Simply said, it is a global platform for exchanging information that cannot be altered or modified. The blockchain, which functions as a public database, is continually updated, and shared across a network of computers. The data will be stored in "blocks." As a result, because many people will utilize the system, all the information will be saved together in the blocks with the help of the "chain." The data in the block cannot be changed in any manner.

Moreover, this project uses HTML/CSS, PHP, and JavaScript computer languages to create a web-based system. Once created, it will use Meta Mask to communicate with the Ethereum blockchain. These three pieces of software are linked with the aid of Meta Mask, which also has a smart contract deployed in it, to effectively develop a Ph.D. certificate verification on the Ethereum blockchain.

All transactions in this project are immutable [31]–[33] since the Ethereum blockchain is being used. It signifies that information cannot be changed after it has been added or altered. As a result, hacking is practically difficult. In addition, the procedure is considerably quicker than human verification because of blockchain transaction automation. This project primarily focuses on one of the issue statements in question.

Aside from that, the Ethereum blockchain has three times as many nodes for transaction verification as Bitcoin and all transactions are cryptographically secure. The blockchain of Ethereum has been running for more than three years, and it has since been shown to be a reliable platform. It implies that the user may trust the blockchain to store all their data. In this case, the user-submitted certificate information will be securely stored in the Ethereum blockchain.

B. QR Code

A QR Code will be produced each time a certificate is uploaded to the Ethereum blockchain. The QR Code's variable, qrImage, is kept in the database and is accessible to users via the "generate QR Code" page when they need to download the QR Code. The user's download folder will then get the QR Code in PNG format. The source code for the QR Code, which contains the user's certificate number, is shown in Figure 1. It is written in PHP programming and is automatically generated whenever a user submits a certificate.

```php
function qrcreate($certNo)
{
    include('phpqrcode/qrlib.php');
    $filename = $certNo;
    $fileDir = 'QRCode/';
    $filetype = ''.$filename.'.png';
    $codeContents = $certNo;
    QRCode::png($codeContents, $fileDir.''.$filetype , QR_ECLEVEL_L, 5);

    return $filetype;
}
```

Fig. 1 A source code for generating QR Code once the certificate added

C. SHA-256

All the certificate's data, excluding the certificate number, will be hashed using SHA-256 when the user uploads or updates their certificate. The certificate number is not hashed since a QR code will be created using it later. The "verify certificate" page will then translate the created QR Code containing the certificate number. The isValidCertificate() function will then be used to verify whether the certificate is valid or a fake based on the certificate number. The source code for the hashing of the data when it is submitted or updated by the user is shown in Figures 2 and 3, respectively.

```php
//getting the post values
$certNo         = $_POST['certNo'];
$qrImage        = qrcreate($certNo);
$name           = hash('sha256',$_POST['name']);
$ic             = hash('sha256',$_POST['ic']);
$studentId      = hash('sha256',$_POST['studentId']);
$programme      = hash('sha256',$_POST['programme']);
$convoDate      = hash('sha256',$_POST['convoDate']);
$semesterFinish = hash('sha256',$_POST['semesterFinish']);
```

Fig. 2 A source code to hash the plaintext after the user submits a certificate

```php
//Getting Post Values
$certNo         = $_POST['certNo'];
$name           = hash('sha256',$_POST['name']);
$ic             = hash('sha256',$_POST['ic']);
$studentId      = hash('sha256',$_POST['studentId']);
$programme      = hash('sha256',$_POST['programme']);
$convoDate      = hash('sha256',$_POST['convoDate']);
$semesterFinish = hash('sha256',$_POST['semesterFinish']);
```

Fig. 3 A source code to hash the plaintext after the user updates a certificate

D. Add Certificate Interface

The interface for the add certificate page is shown in Figures 4 and 5. Users must enter their certificate information here before clicking the "add" button. The message "You are connected to Ethereum blockchain" means that the web-based system interacts with the Ethereum blockchain and is ready to be utilized for adding certificate.

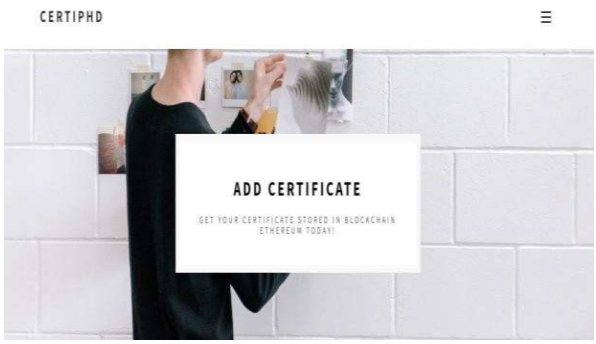

Fig. 4 The interface for add certificate

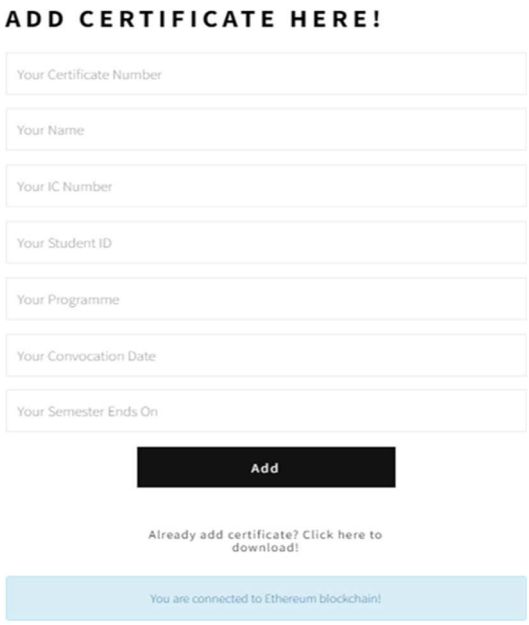

Fig. 5 The interface for add certificate (continue)

Moreover, the source code required to enter data into the Ethereum blockchain is seen in Figure 6. The user will get the message "Certificate already exists." if the certificate number is already in use. The user-provided certificate information will be saved in MySQL and the Ethereum blockchain after submission (using the source code in Figure 7). It will be saved in MySQL immediately after the user clicks the "submit" button. However, a transaction must be completed for the certificate to be put on the Ethereum blockchain. The procedure of adding a certificate is successful if the user clicked "Accept" and the transaction was successful. The pushContractInfo() function in Figure 8 is utilized because, as was previously indicated, a transaction is required for the additional certificate to be successful.

```
if(isset($_POST['submit']))
{
    //getting the post values
    $certNo      = $_POST['certNo'];
    $qrImage     = qrcreate($certNo);
    $name        = hash('sha256',$_POST['name']);
    $ic          = hash('sha256',$_POST['ic']);
    $studentId   = hash('sha256',$_POST['studentId']);
    $programme   = hash('sha256',$_POST['programme']);
    $convoDate   = hash('sha256',$_POST['convoDate']);
    $semesterFinish = hash('sha256',$_POST['semesterFinish']);
    $checkquery =mysqli_query($con, "select certNo from addcertificate where certNo='$certNo'");

    if(mysqli_num_rows($checkquery)>0)
    {
        echo "<script>alert('Certificate is already exist.');</script>";
    }
    else
    {
        // Query for data insertion
        $query=mysqli_query($con, "insert into addcertificate(certNo,name,ic,studentId,programme,convoDate,semesterFinish,qrImage)
        values('$certNo', '$name', '$ic', '$studentId', '$programme', '$convoDate', '$semesterFinish','$qrImage')");
        if ($query==true)
        {
            echo "<script type='text/javascript'>
            pushContractInfo('$certNo','$name','$ic','$studentId','$programme','$convoDate','$semesterFinish');
            alert('You have successfully inserted the data');
            </script>";
        }
        else
        {
            echo "<script>alert('Something Went Wrong. Please try again');</script>";
        }
    }
}
```

Fig. 6 The source code for data insertion

```
<div class="wow fadeInUp col-md-6 col-sm-12" data-wow-delay="1.6s">
    <h1>Add certificate here!</h1>
    <div class="contact-form">
        <form id="contact-form" method="POST">
            <input name="certNo" type="text" name="certNo" class="form-control" placeholder="Your Certificate Number" required>
            <input name="name" type="text" name="name" class="form-control" placeholder="Your Name" required>
            <input name="ic" type="text" name="ic" class="form-control" placeholder="Your IC Number" required>
            <input name="studentId" type="text" name="studentId" class="form-control" placeholder="Your Student ID" required>
            <input name="programme" type="text" name="programme" class="form-control" placeholder="Your Programme" required>
            <input name="convoDate" type="text" name="convoDate" class="form-control" placeholder="Your Convocation Date" required>
            <input name="semesterFinish" type="text" name="semesterFinish" class="form-control" placeholder="Your Semester Ends On" required>
            <center><div class="contact-submit">
                <input type="submit" class="form-control submit" name="submit" value="Add"><br>
            </div></center>
            <center><div class="contact-submit">
                <a href="displaycertificate.php"><h5>Already add certificate? Click here to download!</a><br><br>
            </div></center>
        <center><div class="alert alert-info" role="alert" id="status"></div></center>
        </form>
    </div>
</div>
```

Fig. 7 The source code for the form to get input from the user

```
async function pushContractInfo(a,b,c,d,e,f,g)
{
    const account = await getCurrentAccount();
    debugger;
    console.log(account);
    const coolNumber = await window.contract.methods.addCertificate(a,b,c,d,e,f,g).send({ from: account });
    debugger;
    console.log(coolNumber);
    updateStatus(`Contract Successfully Pushed to Blockchain`);
    debugger;
    console.log(a,b,c,d,e,f,g);
}
```

Fig. 8 The function to push the data into the blockchain

E. Generate QR Code Interface

Figures 9 show the interface for generating QR codes. User has to enter their certificate number and click on the "Generate QR Code" button. If the certificate number exists, the QR Code will appear in Figure 10 and ready to be downloaded by clicking the "Download QR Code" button.

Fig. 9 The interface for generate QR Code

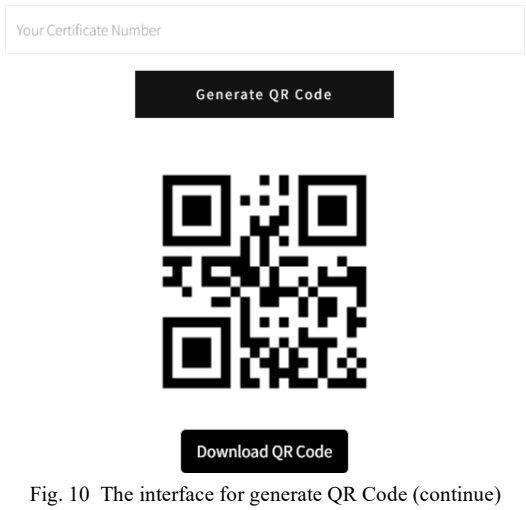

Fig. 10 The interface for generate QR Code (continue)

Other than that, the source code that was utilized after the user hit the "submit" button is seen in Figure 11. Here, the submit button represents the "Generate QR Code" button. After being clicked, the database will choose a QR picture, which will then show up underneath the button. The user will receive a "Certificate does not exist" notice if the supplied certificate number does not exist. After the user selected the "download" button, the source code is displayed in Figure 12. The downloaded QR Code will be kept in.png format in the Download folder.

```php
if(isset($_POST['submit']))
{
    //getting the post values
    $certNo         = $_POST['certNo'];
    $checkquery = mysqli_query($con, "select certNo from addcertificate where certNo='$certNo'");

    if(mysqli_num_rows($checkquery) > 0)
    {
        $query = mysqli_query($con,"SELECT qrImage FROM addcertificate WHERE certNo = '$certNo'");
        $qrImage = mysqli_fetch_assoc($query);
        $hasImage = true;
    }
    else
    {
        echo "<script>alert('Certificate does not exist.');</script>";
    }
}
```

Fig. 11 The source code after the user clicked "submit" button

```php
if(isset($_GET['download'])){

    $qrImage = $_GET['file'];
    $file_path = 'QRCode/'.$qrImage;
    $filename = $qrImage;
    header("Content-Type: application/octet-stream");
    header("Content-Transfer-Encoding: Binary");
    header("Content-disposition: attachment; filename=\"".$filename."\"");
    echo readfile($file_path);
}
```

Fig. 12 The source code after the user clicked "download" button

F. Update Certificate Interface

The user may easily update their certificate on this website if the certificate number remains the same. Because the certificate number serves as the PRIMARY KEY, it must be unique, and no two users can have the same certificate number. Other details can be modified and updated as long as the certificate number remains the same. Figures 13 and 14 shows the interface for the update certificate page. The message "You are connected to Ethereum blockchain" means that the web-based interacts with the Ethereum blockchain and is ready to be used for updating the certificate.

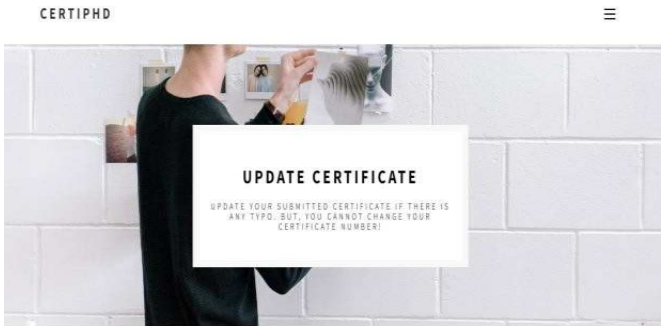

Fig. 13 The interface for update certificate

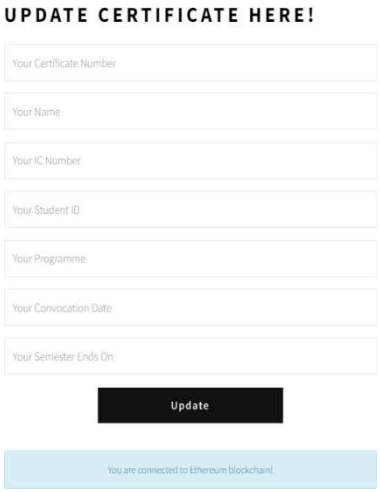

Fig. 14 The interface for update certificate (continue)

Figure 15 displays the source code executed following a user's click on the "submit" and the "update" button. The source code in this case resembles adding certificate source code somewhat. The pushContractInfo() method will push the revised certificate onto the blockchain in hashed form using SHA-256. It is for upgrading the certificate, though, and it is for updating the certificate. Other than that, if the certificate number is different from what was previously submitted, the user cannot amend the information on their certificate.

Aside from that, the JavaScript source code that is used to send the amended certificate data into the blockchain is seen in Figure 16. As previously noted, SHA-256 will be used to hash all certificate information other than the certificate number.

```php
if(isset($_POST['submit']))
{
    //Getting Post Values
    $certNo          = $_POST['certNo'];
    $name            = hash('sha256',$_POST['name']);
    $ic              = hash('sha256',$_POST['ic']);
    $studentId       = hash('sha256',$_POST['studentId']);
    $programme       = hash('sha256',$_POST['programme']);
    $convoDate       = hash('sha256',$_POST['convoDate']);
    $semesterFinish = hash('sha256',$_POST['semesterFinish']);

    $select = mysqli_query($con,"SELECT id FROM addCertificate WHERE certNo='$certNo' ");
    $row = mysqli_num_rows($select);
    if($row > 0){
        //Query for data updation
        $query =  "update addCertificate set name='$name', ic='$ic', studentId='$studentId', programme='$programme',
        convoDate='$convoDate', semesterFinish='$semesterFinish' where certNo = '$certNo'";
        $result = mysqli_query($con,$query);

        if ($result) {
            echo "<script type='text/javascript'>pushContractInfo('$certNo','$name','$ic','$studentId','$programme','$convoDate',
            '$semesterFinish');</script>";
        }
    }
    else {
        echo "<script>alert('Certificate cannot be updated. Please try again');</script>";
    }

    $con->close();
}
```

Fig. 15 The source code after the user clicked "submit" button

```javascript
async function pushContractInfo(a,b,c,d,e,f,g)
{
    const account = await getCurrentAccount();
    const coolNumber = await window.contract.methods.updateCertificate(a,b,c,d,e,f,g).send({ from: account });
    updateStatus(`Contract Successfully Pushed to Blockchain`);
}
```

Fig. 16 The source code to push the updated certificate details into the blockchain

G. Delete Certificate Interface

The interface for deleting a certificate is shown in Figures 17 and 18. The user must provide their certificate number here to erase their certificate from the blockchain. After that, the user may validate the generated QR code that they downloaded when they added their certificate in the added

certificate interface to see if the certificate number is still valid on the blockchain.

Fig. 17 The interface for delete certificate

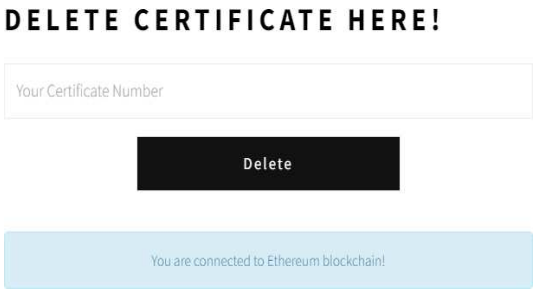

Fig. 18 The interface for delete certificate

The action is taken when a user clicks the "delete" button as seen in Figure 19. The certificate number must be removed from both the MySQL and Ethereum blockchain databases since they are both used by the web-based system for this project. While the pushContractInfo() function is used to push the required certificate number onto the blockchain and get erased, the select function is used to remove the desired certificate number from the MySQL database. The pushContractInfo() function is seen in Figure 20 and is used to push the certificate number into the Ethereum blockchain and request that it be erased.

```
if(isset($_POST['delcertNo']))
{
$rcertNo=$_POST['certNo'];
$sql=mysqli_query($con,"delete from addCertificate where certNo='$rcertNo'");
echo "<script type='text/javascript'>pushContractInfo('$rcertNo');</script>";
}
```

Fig. 19 The source code for the function to select the certificate number from the Ethereum blockchain

```
async function pushContractInfo(a)
{
    const account = await getCurrentAccount();
    const coolNumber = await window.contract.methods.deleteCertificate(a).send({ from: account });
    updateStatus(`Contract Successfully Pushed to Blockchain`);
}
```

Fig. 20 The source code to push the certificate number into the Ethereum blockchain

H. Verify Certificate Interface

Figures 21 and 22 display the user interfaces for the QR Code Scanner page. The user must first enable their camera to scan their QR code. After that, the certificate number will be shown, and you may check it by clicking the Verify button. You are linked to the Ethereum blockchain! indicates that the web-based system is already interacting with the Ethereum blockchain and the certificate is prepared for verification.

Fig. 21 The interface for QR Code scanner

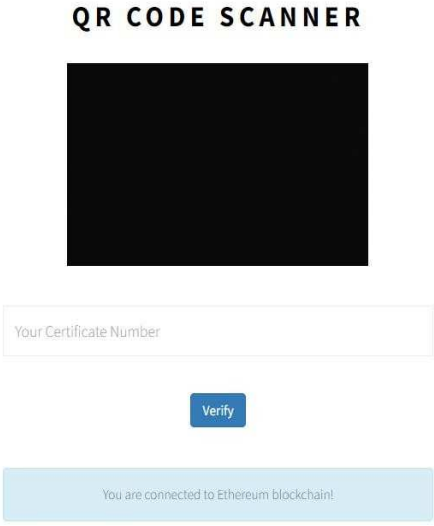

Fig. 22 The interface for QR Code scanner (continue)

The source code needed by the user to begin scanning the downloaded QR Code is displayed in Figure 23. A certificate number will appear once the QR code has been read. However, the certificate number is read-only coded, thus the user cannot change it. The certificate number will then be pushed into the blockchain to be verified, as seen in Figure 24, once the user clicks the "verify" button. In this PHP code, the pushContractInfo() function is used. The source code to put the certificate number into the blockchain is then shown in Figure 25. If the certificate number is valid, the message "Certificate is valid" will show; otherwise, the message "Certificate is fake" will appear.

```
let scanner = new Instascan.Scanner({video:document.getElementById('preview')});
Instascan.Camera.getCameras().then(function(cameras){
    if (cameras.length>0)
    {
        // 0 open the front camera
        // 1 open the back camera
        scanner.start(cameras[0])
    }
    else
    {
        alert("No Camera Found");
    }
}).catch(function(e)
{
    console.error(e);
});
// scan then qr code part
scanner.addListener('scan',function(e){

let string = e;
const qrText = string;

    document.getElementById("certNo").value = qrText.toString();
});
```

Fig. 23 The JavaScript for QR code scanner

```
if(isset($_POST['verify']))
  {
      $certNo = $_POST['certNo'];
      echo "<script type='text/javascript'>pushContractInfo('$certNo');</script>";

  }
```

Fig. 24 The source code to scan the QR code

```
async function pushContractInfo(a)
{
    const isValid = await window.contract.methods.isValidCertificate(a).call();
if(isValid)
{
  alert("Certificate is valid.");
} else
{
  alert("Certificate is fake.");
}
}
```

Fig. 25 The source code to push the certificate number into the Ethereum blockchain

I. Smart Contract: addCertificate function

This function (Figures 26 and 27) that acts as a program in the blockchain to add the certificate into the blockchain. Every variable is declared again with underscore (_) for each of them. The certificate can be added into the blockchain if this program is successfully executed.

```
function addCertificate(string memory _certNo, string memory _name, string memory _ic,
    string memory _studentId, string memory _programme, string memory _convoDate,
    string memory _semesterFinish) public onlyRegistrar returns (bool Status) {

    // Check CertNo existent, true if CertNo existed
    if(isValidCertificate(_certNo) == true) {
        return (false);  // may use revert()
    }

    mapCert[_certNo].name = _name;
    mapCert[_certNo].ic = _ic;
    mapCert[_certNo].studentId = _studentId;
    mapCert[_certNo].programme = _programme;
    mapCert[_certNo].convoDate = _convoDate;
    mapCert[_certNo].semesterFinish = _semesterFinish;
    mapCert[_certNo].next = "";  // next always empty until new added cert trigered. Latest added cert will update previous cert next

    if(totalMapCert == 0) {  // Will add the first cert into mapCert
        mapCert[_certNo].prev = "";  // The first cert added always prev empty
        tempFirstCertNo = _certNo;  // Pointer to the first added CertNo, use for forward traversal searching Cert
    } else {
        mapCert[_certNo].prev = tempLatestCertNo;  // Add previous, use for backward traversal searching Cert
        mapCert[tempLatestCertNo].next = _certNo;  // Update previous cert with next CertNo, use for forward traversal searching Cert

        //mapCert[_certNo].next = tempFirstCertNo; // LinkedList: Making Circle traversal searching Cert,
        //Not Recomended - overheat write operation
```

Fig. 26 addCertificate function on smart contract

```
function addCertificate(string memory _certNo, string memory _name, string memory _ic,
    string memory _studentId, string memory _programme, string memory _convoDate,
    string memory _semesterFinish) public onlyRegistrar returns (bool Status) {

    // Check CertNo existent, true if CertNo existed
    if(isValidCertificate(_certNo) == true) {
        return (false);  // may use revert()
    }

    mapCert[_certNo].name = _name;
    mapCert[_certNo].ic = _ic;
    mapCert[_certNo].studentId = _studentId;
    mapCert[_certNo].programme = _programme;
    mapCert[_certNo].convoDate = _convoDate;
    mapCert[_certNo].semesterFinish = _semesterFinish;
    mapCert[_certNo].next = "";  // next always empty until new added cert trigered. Latest added cert will update previous cert next

    if(totalMapCert == 0) {  // Will add the first cert into mapCert
        mapCert[_certNo].prev = "";  // The first cert added always prev empty
        tempFirstCertNo = _certNo;  // Pointer to the first added CertNo, use for forward traversal searching Cert
    } else {
        mapCert[_certNo].prev = tempLatestCertNo;  // Add previous, use for backward traversal searching Cert
        mapCert[tempLatestCertNo].next = _certNo;  // Update previous cert with next CertNo, use for forward traversal searching Cert

        //mapCert[_certNo].next = tempFirstCertNo; // LinkedList: Making Circle traversal searching Cert,
        //Not Recomended - overheat write operation
```

Fig. 27 addCertificate function on smart contract (continue)

J. Smart Contract: updateCertificate function

This function (Figure 28) works almost the same with addCertificate function. It will update the certificate if the certificate number is remained unchanged.

```
function updateCertificate(string memory _certNo, string memory _name, string memory _ic,
    string memory _studentId, string memory _programme, string memory _convoDate,
    string memory _semesterFinish) public onlyRegistrar returns (bool Status) {

    if(isValidCertificate(_certNo) == false) {
        return (false); // may use revert()
    }

    mapCert[_certNo].name = _name;
    mapCert[_certNo].ic = _ic;
    mapCert[_certNo].studentId = _studentId;
    mapCert[_certNo].programme = _programme;
    mapCert[_certNo].convoDate = _convoDate;
    mapCert[_certNo].semesterFinish = _semesterFinish;

    lastUpdate = block.timestamp;
    emit updatedCertificate(_certNo);  // Event Log

    return true;
}
```

Fig. 28 updateCertificate function on smart contract

K. Smart Contract: deleteCertificate function

This function (Figures 29 and 30) will delete the certificate when the user enters the certificate number. For this program, there is no need to put the whole certificate details only it is to be deleted then.

```
    if(isValidCertificate(_certNo) == false) {
        return (false);  // may use revert()
    }
    else {  // Many condition MUST be satified to avoid broken LinkedList of Cert
        if(strCompare(mapCert[_certNo].prev, "") != 0) {  // if previous CertNo exist, change it next pointer
                                                          //to the deleted cert next pointer
            mapCert[mapCert[_certNo].prev].next = mapCert[_certNo].next;
        }

        if(strCompare(mapCert[_certNo].next, "") != 0) {  // if next CertNo exist, change it prev pointer
                                                          //to the deleted cert prev pointer
            mapCert[mapCert[_certNo].next].prev = mapCert[_certNo].prev;
        }

        // If the deleted CertNo it is the first cert in list, update the second cert as the new first cert
        if(strCompare(tempFirstCertNo, _certNo) == 0 && strCompare(mapCert[_certNo].next, "") != 0) {
            tempFirstCertNo = mapCert[_certNo].next;
        }
```

Fig. 29 deleteCertificate function on smart contract

```
        // If the deleted CertNo it is the last cert in list, update the second last cert as the new last cert
        if(strCompare(tempLatestCertNo, _certNo) == 0 && strCompare(mapCert[_certNo].prev, "") != 0) {
            tempLatestCertNo = mapCert[_certNo].prev;
        }

        if(totalMapCert == 1) {  // If only one cert exist, deleted cert will cause First & Latest pointers to Empty
            tempFirstCertNo = "";
            tempLatestCertNo = "";
        }

        delete mapCert[_certNo];
        totalMapCert -= 1;  // deduct cert counter

        lastUpdate = block.timestamp;
        emit deletedCertificate(_certNo);  // Event Log

        return (true);
    }
}
```

Fig. 30 deleteCertificate function on smart contract (continue)

L. Smart Contract: isValidCertificate function

This function (Figure 31) is important because it will verify the originality of the certificate that is kept in the blockchain. If the certificate number that the user enters is exist in the blockchain, then it means that the certificate is valid. Else, it means the certificate is never stored in the blockchain thus it also means that the certificate is fake.

```
function isValidCertificate(string memory _certNo) public view returns (bool Status) {

    // if studentId is empty, then CertNo not exist
    if(strCompare(mapCert[_certNo].studentId , "") == 0)  // 0 is equal
        return (false);
    else
        return (true);
}
```

Fig. 31 isValidCertificate function on smart contract

III. Results and Discussion

A. User Testing

In conclusion, in user testing result, the verification system has successfully passed all the checklist statements, indicating the successful implementation and functionality of the various components. In Part A, the user can add a certificate, and Meta Mask prompts for transaction approval, ensuring secure interactions. The transaction is successful, and the certificate number is stored in the smart contract on Remix IDE Ethereum. Duplicate certificate numbers are detected, and the user receives a message indicating that the certificate already exists.

TABLE I
User Testing Result for Certified

Checklist Statement	Pass	Fail
Part A: Add Certificate		
The user can click the add button.	√	
Meta Mask asked permission to accept or reject the transaction.	√	
The transaction is successful.	√	
The certificate number exists in the smart contract on Remix IDE Ethereum.	√	
The user will receive a message "Certificate is already exist" if the user entered the same certificate number that is stored in the blockchain.	√	
Part B: Generate QR Code		
The user can enter their certificate number.	√	
The user can see the generated QR Code after the user clicked "Generate QR Code" button.	√	
The user can download the QR Code after the user clicked "Download" button.	√	
The user can see the generated QR Code after it is successfully downloaded.	√	
Part C: Update Certificate		
The user can update a certificate.	√	
No QR Code is generated right after the user clicked on the submit button.	√	
The user can update a certificate without changing the certificate number.	√	
Meta Mask asked permission to accept or reject the transaction.	√	
The transaction is successful.	√	
The hashed data that the user has updated is successfully updated in the smart contract on Remix IDE Ethereum.	√	
Part D: Delete Certificate		
The user can enter the certificate number in the form.	√	
The delete certificate page is refreshed when the user clicked on the submit button.	√	
Part E: Verify Certificate		
The user can scan the QR code using their camera.	√	
The certificate number is displayed on the certificate number form.	√	
The user can click on the Verify button.	√	
The message "Certificate is valid" or "Certificate is fake" appears right after the Verify button is clicked.	√	

In Part B, the user can generate a QR code by entering the certificate number, view it, and download it. Part C enables certificate updates without changing the certificate number, with Meta Mask requesting transaction approval and successful updates recorded in the smart contract. In Part D, users can delete certificates by entering the certificate number, refreshing the delete certificate page upon submission. Lastly, Part E allows users to verify certificates by scanning the QR code, displaying the certificate number, and clicking the Verify button. The system promptly responds with a message indicating whether the certificate is valid or fake. Overall, the verification system has effectively fulfilled all the requirements and offers a robust solution for validating Ph.D. certificates using QR codes on the Ethereum blockchain.

B. Scenario Test Plan

The scenario test plan results indicate a successful execution of all the specified tasks. In Part A, the certificate is successfully added to the database and blockchain, with the certificate (excluding the certificate number) being hashed using SHA256 before storage. The user is able to click the submit button without any issues. In Part B, the certificate is successfully updated in the database and blockchain, ensuring that the hashed data is also updated accordingly.

TABLE II
Scenario Test Plan Result for Certified

Checklist Statement	Pass	Fail
Part A: Add Certificate		
The certificate is successfully added into the database and blockchain.	√	
Certificate (except certificate number) is hashed by using SHA256 before it is stored into the database and blockchain.	√	
The user be able to click the submit button.	√	
Part B: Update Certificate		
The certificate is successfully updated into the database and blockchain.	√	
The hashed data is updated for both in the database and blockchain.	√	
The user can click the update button.	√	
Part C: Generate QR Code		
The generated QR Code appears after the user clicked "Generate QR Code" button.	√	
The user can download the generated QR Code.	√	
Part D: Delete Certificate		
The certificate is successfully deleted from the database and blockchain.	√	
The user can click the delete button.	√	
Part E: Verify Certificate		
The user can open their camera after they allow the camera permission.	√	
The user can interpret the certificate number that is contained in the QR code.	√	
The user can check the originality of the certificate and valid or fake message.	√	

The user can click the update button as expected. In Part C, the generated QR code appears as intended after the user clicks the "Generate QR Code" button, and the user is able to download the generated QR code without any complications. Part D demonstrates the successful deletion of the certificate from both the database and blockchain, and the user can click the delete button without any difficulties. Lastly, in Part E, the user can open their camera after granting camera permission, interpret the certificate number contained in the QR code accurately, and check the originality of the certificate along with the corresponding valid or fake message. Overall, the scenario test plan yielded positive outcomes, indicating that the system functions as intended, ensuring the successful addition, update, generation, deletion, and verification of Ph.D. certificates using QR codes on the Ethereum blockchain.

IV. CONCLUSION

To conclude the research, the verification of Ph.D. certificate using QR code on Ethereum blockchain is successfully developed by using Solidity language for its smart contract, PHP, HTML/CSS for the implementation of web-based system and its functionality, Meta Mask that acts a blockchain and Remix IDE software to deploy the smart contract. Certificate verification is a successful project and can be verified in less than one hour if the Meta Mask has the ether to make a transaction. Also, that certificate verification is fast, user-friendly, and can be sent by whoever needs to verify a Ph.D. certificate.

Consideration should be given to exploring the scalability of the system and its potential for adoption by other educational institutions beyond the scope of this research. Addressing scalability challenges will be vital to ensure that the system can efficiently handle a large volume of certificate verification requests while maintaining optimal performance. Also, the system should be designed to address privacy concerns and ensure robust data protection. Exploring encryption techniques and privacy-enhancing technologies will be crucial in safeguarding sensitive student data while preserving the transparency and verification capabilities of the blockchain system.

However, further refinement of the user experience aspects of the system is necessary. This entails focusing on interface design, usability considerations, and incorporating user feedback or conducting usability testing to ensure the system is intuitive, user-friendly, and accessible for all stakeholders involved in the certificate verification process. Besides, exploring the feasibility and challenges associated with integrating the proposed blockchain-based certificate verification system with existing infrastructure and systems used by universities, employers, or other relevant parties is essential. Addressing interoperability concerns and highlighting potential benefits of seamless integration will facilitate efficient collaboration and utilization of the system across various platforms and entities.

ACKNOWLEDGMENT

This research is fully funded and supported by Universiti Tun Hussein Onn Malaysia (UTHM) Tier 1 Grant Scheme (Vot: Q027).

REFERENCES

[1] N. Malsa, V. Vyas, J. Gautam, A. Ghosh, and R. N. Shaw, "CERTbchain: A Step by Step Approach Towards Building A Blockchain based Distributed Appliaction for Certificate Verification System," in *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, Dec. 2021, pp. 800–806. doi: 10.1109/ICCCA52192.2021.9666311.

[2] L. S. S, Mrs. P. N, and A. Shettar, "Block chain Based Framework for Document Verification," in *2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP)*, IEEE, Feb. 2022, pp. 1–5. doi: 10.1109/AISP53593.2022.9760651.

[3] A. A. Abuhashim, H. A. Shafei, and C. C. Tan, "Block-VC: A Blockchain-Based Global Vaccination Certification," in *2021 IEEE International Conference on Blockchain (Blockchain)*, IEEE, Dec. 2021, pp. 347–352. doi: 10.1109/Blockchain53845.2021.00055.

[4] H. Liang, H. Yin, and X. Niu, "A Robust Digital Watermarking Scheme and Its Application in Certificate Verification," in *2009 International Conference on Measuring Technology and Mechatronics Automation*, IEEE, 2009, pp. 410–413. doi: 10.1109/ICMTMA.2009.295.

[5] G. Nian, X. Tang, D. Wang, and H. Liu, "Print-scan resilient data hiding scheme applied in certificate verification," in *2010 3rd International Congress on Image and Signal Processing*, IEEE, Oct. 2010, pp. 1161–1165. doi: 10.1109/CISP.2010.5646731.

[6] Y.-L. Tang and Y.-T. Huang, "Print-and-Scan Resilient Watermarking for Authenticating Paper-Based Certificates," in *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, IEEE, Sep. 2010, pp. 357–361. doi: 10.1109/PCSPA.2010.93.

[7] E. Febriyanto, U. Rahardja, A. Faturahman, and N. Lutfiani, "Sistem Verifikasi Sertifikat Menggunakan Qrcode pada Central Event Information," *Techno.Com*, vol. 18, no. 1, pp. 50–63, Feb. 2019, doi: 10.33633/tc.v18i1.2078.

[8] H. A. M. Wahsheh and M. S. Al-Zahrani, "Secure and Usable QR Codes for Healthcare Systems: The Case of Covid-19 Pandemic," in *2021 12th International Conference on Information and Communication Systems (ICICS)*, IEEE, May 2021, pp. 324–329. doi: 10.1109/ICICS52457.2021.9464565.

[9] A. Singhal and R. S. Pavithr, "Degree Certificate Authentication using QR Code and Smartphone," *Int J Comput Appl*, vol. 120, no. 16, pp. 38–43, Jun. 2015, doi: 10.5120/21315-4303.

[10] M. K. Dash, G. Panda, A. Kumar, and S. Luthra, "Applications of blockchain in government education sector: a comprehensive review and future research potentials," *Journal of Global Operations and Strategic Sourcing*, vol. 15, no. 3. Emerald Group Holdings Ltd., pp. 449–472, Aug. 05, 2022. doi: 10.1108/JGOSS-09-2021-0076.

[11] I. Khan and A. Shahaab, "A Peer-To-Peer Publication Model on Blockchain," *Frontiers in Blockchain*, vol. 4, Feb. 2021, doi: 10.3389/fbloc.2021.615726.

[12] M. Sharples and J. Domingue, "The Blockchain and Kudos: A Distributed System for Educational Record, Reputation and Reward," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2016, pp. 490–496. doi: 10.1007/978-3-319-45153-4_48.

[13] 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, Apr. 2020, doi: 10.1007/s11042-019-08059-1.

[14] Y. C. Elloh Adja, B. Hammi, A. Serhrouchni, and S. Zeadally, "A blockchain-based certificate revocation management and status verification system," *Comput Secur*, vol. 104, p. 102209, May 2021, doi: 10.1016/j.cose.2021.102209.

[15] S. Pu and J. S. L. Lam, "The benefits of blockchain for digital certificates: A multiple case study analysis," *Technol Soc*, vol. 72, Feb. 2023, doi: 10.1016/j.techsoc.2022.102176.

[16] M. Alsadi, J. Arshad, J. Ali, A. Prince, and S. Shishank, "TruCert: Blockchain-based trustworthy product certification within autonomous automotive supply chains," *Computers and Electrical Engineering*, vol. 109, Aug. 2023, doi: 10.1016/j.compeleceng.2023.108738.

[17] M. Foy, D. Martyn, D. Daly, A. Byrne, C. Aguneche, and R. Brennan, "Blockchain-based governance models for COVID-19 digital health certificates: A legal, technical, ethical and security requirements analysis," *Procedia Comput Sci*, vol. 198, pp. 662–669, 2022, doi: 10.1016/j.procs.2021.12.303.

[18] A. Bathula, S. Muhuri, S. kr. Gupta, and S. Merugu, "Secure certificate sharing based on Blockchain framework for online education," *Multimed Tools Appl*, vol. 82, no. 11, pp. 16479–16500, May 2023, doi: 10.1007/s11042-022-14126-x.

[19] G. Sethia, S. Namratha, H. Srikanth, and S. C. Sreeja, "Academic Certificate Validation Using Blockchain Technology," in *2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies, TQCEBT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/TQCEBT54229.2022.10041550.

[20] T. Rama Reddy, P. V. G. D. Prasad Reddy, R. Srinivas, C. V. Raghavendran, R. V. S. Lalitha, and B. Annapurna, "Proposing a reliable method of securing and verifying the credentials of graduates through blockchain," *EURASIP J Inf Secur*, vol. 2021, no. 1, Dec. 2021, doi: 10.1186/s13635-021-00122-5.

[21] H. Zhang and F. Zhao, "Cross-domain identity authentication scheme based on blockchain and PKI system," *High-Confidence Computing*, Mar. 2023, doi: 10.1016/j.hcc.2022.100096.

[22] N. Nousias, G. Tsakalidis, G. Michoulis, S. Petridou, and K. Vergidis, "A process-aware approach for blockchain-based verification of academic qualifications," *Simul Model Pract Theory*, vol. 121, p. 102642, Dec. 2022, doi: 10.1016/j.simpat.2022.102642.

[23] H. Chen, X. Luo, L. Shi, Y. Cao, and Y. Zhang, "Security challenges and defense approaches for blockchain-based services from a full-stack architecture perspective," *Blockchain: Research and Applications*, p. 100135, Mar. 2023, doi: 10.1016/j.bcra.2023.100135.

[24] S. S. Nabil, M. S. Alam Pran, A. A. Al Haque, N. R. Chakraborty, M. J. M. Chowdhury, and M. S. Ferdous, "Blockchain-based COVID vaccination registration and monitoring," *Blockchain: Research and Applications*, vol. 3, no. 4, Dec. 2022, doi: 10.1016/j.bcra.2022.100092.

[25] M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar, "CertLedger: A new PKI model with Certificate Transparency based on blockchain," *Comput Secur*, vol. 85, pp. 333–352, Aug. 2019, doi: 10.1016/j.cose.2019.05.013.

[26] A. Ismail, M. Toohey, Y. C. Lee, Z. Dong, and A. Y. Zomaya, "Cost and Performance Analysis on Decentralized File Systems for Blockchain-Based Applications: State-of-the-Art Report," in *2022 IEEE International Conference on Blockchain (Blockchain)*, IEEE, Aug. 2022, pp. 230–237. doi: 10.1109/Blockchain55522.2022.00039.

[27] R. Xie *et al.*, "Ethereum-Blockchain-Based Technology of Decentralized Smart Contract Certificate System," *IEEE Internet of Things Magazine*, vol. 3, no. 2, pp. 44–50, Jun. 2020, doi: 10.1109/iotm.0001.1900094.

[28] A. Urquhart, "Under the hood of the Ethereum blockchain," *Financ Res Lett*, vol. 47, Jun. 2022, doi: 10.1016/j.frl.2021.102628.

[29] H.-J. Yoon, "Blockchain Technology and Healthcare," *Healthc Inform Res*, vol. 25, no. 2, p. 59, Apr. 2019, doi: 10.4258/hir.2019.25.2.59.

[30] A. Dika and M. Nowostawski, "Security Vulnerabilities in Ethereum Smart Contracts," in 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, Jul. 2018, pp. 955–962. doi: 10.1109/Cybermatics_2018.2018.00182.

[31] A. F. Camilleri, "Blockchain in Education," 2017. doi: 10.2760/60649.

[32] M. Aamir, R. Qureshi, F. A. Khan, and M. Huzaifa, "Blockchain Based Academic Records Verification in Smart Cities," *Wirel Pers Commun*, vol. 113, no. 3, pp. 1397–1406, Aug. 2020, doi: 10.1007/s11277-020-07226-0.

[33] D. S. V. Madala, M. P. Jhanwar, and A. Chattopadhyay, "Certificate transparency using blockchain," in *IEEE International Conference on Data Mining Workshops, ICDMW*, IEEE Computer Society, Feb. 2019, pp. 71–80. doi: 10.1109/ICDMW.2018.00018.