The University of Birmingham
School of Computer Science
MSc in Advanced Computer Science

Better Security Over Blockcerts

Student Id: 1709735
Student Name: Rujia Li
Supervisor: Dr. David Galindo

September 2017

DECLARATION

I declare that this thesis is a presentation of my original research work. However, I rewrite or reused some paragraphs based on the report of my mini-project, for the reason that my summer project was extended based on the mini-project (courses number, 21891).

I clarify the reused or rewrite sections as following: the rewrite ones contain the overview section in "Introduction" parts and some portions of blockchain section in "Background" parts, and the reused ones include the multi-signature assessment results in 4.2.4 section.

This declaration is based on the suggestions of the plagiarism director Dr. Alan P. Sexton and approved by my summer project supervisor Dr. David Galindo.

ACKNOWLEDGMENTS

The authors would like to thank the researchers from the MIT for the discussions on this topic and for sharing their project idea. Also, the author thanks the following colleagues for feedback on drafts of this paper: Dr. David Galindo, Dr. Qi Wang, Yifan Wu, Huanyuan Zhang, especially appreciates Dr. David Galindo's advice on optimizing the thesis and his encouragement throughout the entire project. Lastly, The authors gratefully acknowledged funding from the IT Innovation Centre, University of Birmingham.

CONTENT

List of Figures

List of Tables

Abstract

Counterfeit academic certificates have been a longstanding issue in the academic community. Not until the Massachusetts Institute of Technology Media Lab released their project of Block-certs, a technique which is mainly implemented by conflating the hash value of local files to the blockchain but remains numerous issues, did an effective technological approach protecting authentic credential certification and reputation appear.

Based on *Blockcerts*, a series of cryptographic solutions are proposed to resolve the issues above, including, utilizing a multi-signature scheme to ameliorate the authentication of certificates; exerting a safe revocation mechanism to improve the reliability of certificates revocation; establishing a secure federated identification to confirm the identity of the issuing institution.

The project consists in designing and implementing the system which covered the above solutions. The project also involves a comprehensive evaluation of the system security, and the assessment outcomes provide compelling evidence to prove that implementation is practical, reliable, secured, which might give some hints of important architectural considerations about the security attributes of other blockchain-based systems.

Keywords
The counterfeit certificate; *Blockcerts*; Blockchain; Cryptographic protocols;

1. Introduction

Undoubtedly, the emergences of the Internet and the contemporary document/image technologies have contributed to the extensive tendency of forging of educational background and spurious diplomas. For instance, according to a report released by Risk Advisory Group[1], an independent research institution reported in March 2016 that 63% of 5,500 CVs selected by them contained inaccurate or false educational qualifications[2]. It is also pointed out by the Economist that spurious credentials are making employers mad[3]. An employer is impossible to determine whether a candidate is competent for a particular job without background investigation and verification with the aid of qualification granting institutions or other reliable authorities like HEDD[4]. However, this will take a lot of time and efforts since they have to handle thousands of credentials synchronously in this way. In this case, it is widely admitted that counterfeit academic certificate has become a thorny and long-term problem faced by higher education and till now, there is no effective solution to this issue.

1.1 Overview

In 2016, MIT Media Lab released their project named *Blockcerts*, stating that they have successfully developed a solution that can technically protect authentic qualifications by exerting blockchain[5]. The main work-flow of this system is as follows: generate certificate including recipient information, awarding body and issued date in JSON format[6]; use a standard hash function like SHA-256 to compute a unique hash value for the digitized credential; add a range of hash values into a Merkle tree; attached Merkle Root to the blockchain through broadcasting transaction.

1.2 Problems statement

The Blockchain-based method succeeded in protecting the integrity and authenticity of the certificate, however, it is not free from drawbacks. Its shortcomings ranged from forgery preventing, revocation mechanization to identity proving. We summarized the weakness as follows:

Key Security Problem
A single private key combined with BTC address is used in *Blockcerts* to verify the identity of a certificate awarding institution, indicating that such private key can be adopted to pay and attach a student's digital certificate hashes as reference. However, single private key management scheme adopted by MIT is highly risky. Firstly, if the only private key, a randomly selected number on which the limits of authority of issuing digital certificates depend, is stolen or erased, the new possessor becomes able to release or revoke whatever records they want. What is worse, any changes made illegally in blockchain will be immutable. In short, the key security problem is the core one that makes the project impractical to some degree. Secondly, the right of issuing certificates is monopolized by the exact person owning the private key, which brings up lots of rights problem like corruption.

Certificate Revocation Problem
The revocation mechanism in *Blockcerts* version 2.0 is designed to check the URL-based certificate revocation list(UCRL), UCRL is an URL for querying the list of certificates that having been revoked by the awarding institution before their expected expiry dates. In other

words, for verifying each certificate authenticity and integrity, it is required to check the revocation list by this URL. Whereas considering that the URL is fixed and embedded in the issued certificates, it is unreasonable to assume that it was completely reliable. Alternately, the URL represents a single point of failure which caused the revocation mechanism unreliable and vulnerable.

Certificate Awarding Institution’s Identity Problem

In *Blockcerts* version 2.0, to determine whether a certificate is real or not, an independent verifier only needs to verify the timestamp and the issuing key owner. Specifically speaking, such auditor is only required to confirm the certificate awarding institution did own the key when the certificate was awarded. However, the project failed to provide this confirmation functionality, as identity proof was still regarded as an external layer in *Blockcerts* 2.0.

1.3 Our contribution

We mainly had three contributions in this project. Firstly, the problems pointed out above about certificate authority and security were tackled through designing and applying a range of cryptographical protocols.

These cryptographical protocols includes using multiple signature approaches to improve certificate authentication, exerting a safe revocation mechanism to revoke a certificate more reliably, and establishing a secure federated identity aimed at verifying certificate awarding institution's identity. These protocols remedy the shortcomings of the *Blockcerts* to a certain degree, which avail universities of a practical blockchain-based certificate system.

Secondly, we utilizing Java and JavaScript to implement a blockchain-based certificate system embracing all the above protocols. Additionally, the system and protocols can be used in other related fields such as digital right protecting and contract proofing. Case in point, our protocol enables two companies to attach their contract onto the blockchain with multi-signature, which is different from the traditional third party-based work mode and dispel the worries of forging credentials.

Thirdly, we have comprehensively evaluated the system security form four dimensions, namely operation security, data security, network security and protocol security. The assessment outcomes provide compelling evidence to prove that implementation is practical, reliable, secured, which might give some hints of important architectural considerations about the security attributes of other blockchain-based systems.

1.4 Paper organization

The remainder of the thesis is divided into four chapters. Chapter 2 reviews literatures related to the *Blockcerts* and Blockchain; Chapter 3 provides the details of three cryptographic protocol. The implementation and evaluation of the prototype are elaborated in Chapter 4. Ultimately, the project conclusion as well as the future work, is stated in the last Chapter.

2. Background

This section concentrates on the inner working mechanism of *Blockcerts* and Blockchain, which serve as a substratum. Specifically, we explained the background and workflow of *Blockcerts* in part 1 and explored the Blockchain mechanism in part 2. Please noted that many details are obtained by the analysis of the corresponding source code due to lack of authentic paper and official documentation.

2.1 Blockcerts

Blockcerts is an open credentialing system implemented by Python which launched by MIT Media Lab in June 2016. *Blockcerts* utilizing Bitcoin blockchain acts as the infrastructure of trust and database[7]. This infrastructure removes the dependence on third party arbitration which is impermanent and filled with complication[8].

Blockcerts releases the credential by sending a Bitcoin transaction from awarding institution to recipient. A good metaphor of the *Blockcerts* is illustrated in Figure 1, a set of hash value of certificates would be attached to the Bitcoin transaction when Alice paid 5 BTC to Tom. Then, *Blockcerts* allows an independent verifier to verify the authenticity of such credential by accessing hash value on the blockchain and comparing it with the local digital file's.

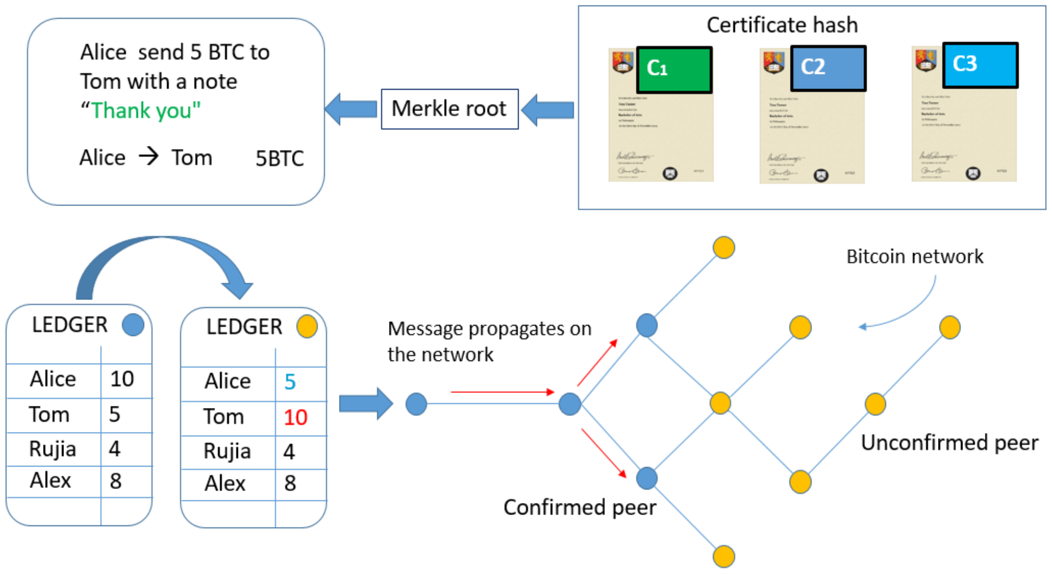


Figure 1. Working mechanism of *Blockcerts*

As designed in version 1.0, each credential has an unique transaction in the blockchain[6] leading to copious amount of transaction, which means high transaction expenditure. What's more, the minimum transaction fee, which required to be paid by an awarding institution to avoid the penny-flooding attack, will be over 000.1 BTC[9].

Fortunately, in version 1.2, the MIT researchers contrive an innovative solution to avoid the high transaction fee by means of merging the hash values into Merkle tree[10], namely, they publish the Merkle Root rather than single hash which economized transaction costs.

2.2 Blockchain

The concept of Blockchain is proposed based on Bitcoin technology. Bitcoin is an open-source cryptographic currency system and conceptualized by Satoshi Nakamoto in 2008[11]. This decentralized system aimed at publishing transactions carrying payments[12]. Concretely, the Bitcoin users launch the transactions that need to be authenticated using the corresponding private key, then, they broadcast these transactions to Bitcoin peer-to-peer (P2P) network[13], lastly, the transactions will be collected into blockchain by miners through solving cryptographic puzzles of controlled hardness[14].

Blockchain is a distributed database which maintains a keep-growing list of ordered records called block[15]. Each block consists a block header and other metadata such as the block size and transactions TX_i[16]. Each header includes a timestamp T_i, a link to a previous block hash H_i-1 and nonce N_i (Table 1).

Table 1. Structure of the block.

Block size	
Block Header	Version Previous block header hash Merkle root Time Ti NBits Nonce
Transaction count	
Transactions (The transaction record in this block)	

Among all the elements in the block, the transaction plays a core role in carrying the payments. A transaction is a formatted data structure associated with identities(address) and bitcoin value. Each transaction is composed of a string of hexadecimal messages including inputs, outputs and lock time. etc., The new transection inputs are identical to the previous transection output, and commits all itself to new outputs[17]. Meanwhile, the output contains a script snippet for triggering the condition of redeeming money[18]. The lock time is defined as the lag time that a transaction is valid or added to the blockchain.

Table 2. Format of a Bitcoin transaction[17]

version	
input count	
inputs	Previous output hash(reversed) Previous output index Script length Script Snippet (public key with script signature) Sequence
output count	
output	Value Script length Script public key
block lock time	

In bitcoin transactions, payers and payees are identified by bitcoin address or pseudonyms[19]. As for the transaction process, initially, a transaction is required to be generated and signed by a payer's private key, followed by the miners adding the signed transaction to blockchain through computing. Eventually, the coin possessor will be able to redeem the money by verifying the signature.

Among all the elements in the header, the Merkle tree is used to merge all the transactions. A Merkle tree[20] is defined as an integrated binary tree, which is associated with each node. each interior node value is a one-way function of the node values of its children.

Meanwhile, the block header plays a key role in collecting all the transactions to the block, since the blockchain is cryptographically secured, for every round, the miner need to find a random number to match the computing difficulty d_i, and this progress is called the proof of work[21]. The computing formula is as follow:

$$f(D_i) > \text{SHA-256}(\text{SHA-256}(H_i{-}1 || T_i || TX_i || d_i || N_i))$$

Note: "Di" is defined as the proof-of-work algorithm difficulty target for this block.

Once the CPU effort has been expended to make it satisfy the condition (some top bits' collision), the block cannot be altered without redoing the previous work owing to the fact that the following blocks has been chained after it.

3. Cryptographic protocol design

The protocol designing, to solve the above problems, acts as an important part of our project. In this section, we described how cryptographic protocol are designed and utilized. We also presented how it coped with technical limitations of *Blockcerts*.

3.1 Multi-signature scheme

As mentioned in “Key Security Problem” section, The *Blockcerts* is facing some grave security problems in private key management and utilizing, specifically, the risk of leaking private key and key supervision issue. This section highlights the multi-signature solution to solve the above problems. Overall, the multi-signature scheme can be divided into three stage: initialization stage, signing signature stage, confirming signature stage. The initialization stage is in charge of generating the redeem script and the combination BTC address; the signing signature stage is responsible for adding the required signature to executable stack; the confirming signature stage is running on the miner and used for releasing the encumbrance.

3.1.1 Initialization stage

In the initialization phase, we generated some pairs of the private key according to the Elliptic Curve Digital Signature Algorithm[22] (ECDSA), then we combined the public keys and signature required count to obtain the BTC address and corresponding redeem script. We illustrated the multi-signature address generating algorithm and the multi-signature redeem script algorithm in “Algorithm I” and “Algorithm II“ respectively.

Table 3. Generating multi-signature address algorithm

```
Algorithm I Generating Multi-Signature Address
Input: a list {Pki}i=1 n of the ECDSA public keys
Input: the total public key number N
Input: the threshold of signatures required for validation M
Input: the multi-signature address version η (0x05)
Output: the multi-signature address
1: κ = OP_1 + M– 1
2: for i ← [1,N] do
3:     λ ← λ + Pki
4: end for
5: κ ← λ + (OP_1 + N - 1)
6: κ ← κ + OP_CHECKMULTISIG
7: μ = η +RIPEMD-160 (SHA-256(κ))
8: δ ←SHA-256(SHA-256( μ ))
9: the multi-signature address = Base58Encode (μ +δ[0:4])
Return: the multi-signature address
```

Note: δ[0:4] means the first 4 bits of δ, the symbol of “OP_1” - “OP_16” represented operating code which range from 1 to 16, the“OP_CHECKMULTISIG” is a special operating code, this operating code has the functionality of comparing the first signature against each public key until it finds an ECDSA match.

Table 4. Generating the redeem script algorithm

```
Algorithm II Generating the Redeem Script
Input: a list {Pki}i=1 n of the ECDSA public keys
Input: the total public key number N
Input: the threshold of signatures required for validation M
Input: the multi-signature address version η (0x05)
Output:  the redeem script
1: κ = OP_1 + M – 1
2: for i ← [1,N] do
3:     λ ← λ + Pki
4: end for
5: κ ← λ + (OP_1 + N - 1)
6: κ ← κ + OP_CHECKMULTISIG
7: the redeem script = BytesToHex (κ)
Return: the redeem script
```

Herein, it is obvious that combination address is another expression of the redeem script. The combination address is used to share and communicate in the Bitcoin market, While the redeem text with signature acts as an encumbrance which is pending until released by the miners. To visualize the redeem script, ‘2 of 3’ redeem script was used as an example below, where 3 is the total number of keys and 2 is the threshold of signatures required for validation. The last redeem text shown in Figure 1.

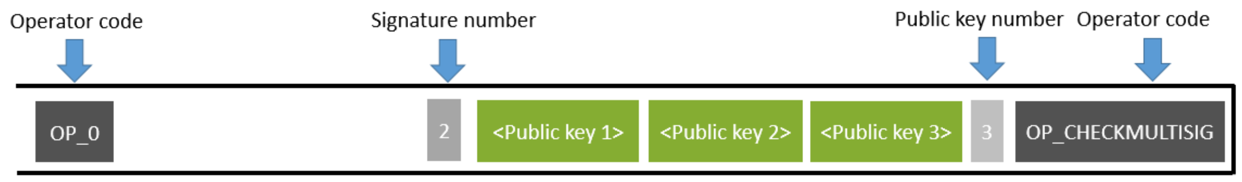


Figure 2. Redeem script without signatures.

3.1.2 Signing signature stage

A signature scheme, constructed in this phase, makes the transaction data accessible to N signers, each of whom receives a portion of the private key and N of them are required to sign(N>M/2+1). This signature scheme uses a standard ECDSA signature albeit put all the signed result in one stack. For an elliptic curve E with a generator point G of order n, The message m is a hexadecimal raw transaction string

- The private key Pr is a number d < n
- The public key Pk is corresponding curve point Q = dG
- F is the cryptographic hash function such as SHA-256

Table 5. Generating the signature stack algorithm

```
Algorithm III Generating the Signature Stack
Input:  The condition we defined above
Input:  An empty Stack S
Output:  The Signature List Stack
1: κ  =  the prefix OP_0
2: for i ← [1,M] do
3:     Ki = Ki < n
4:     R ← (r1, r2) = kG
5:     r ← r1 mod n
6:     s ← k-1(F(m)+dr)
7:      ο i ← (r,s)
8: end for
Return: S = push (κ+oi)
```

After this progress, the private keys, corresponding to the N listed public keys, will generate a combination of M signatures in a signature stack as follow, The order of signatures must match the order of the public keys[23]. Otherwise, the miner will reject the transaction.

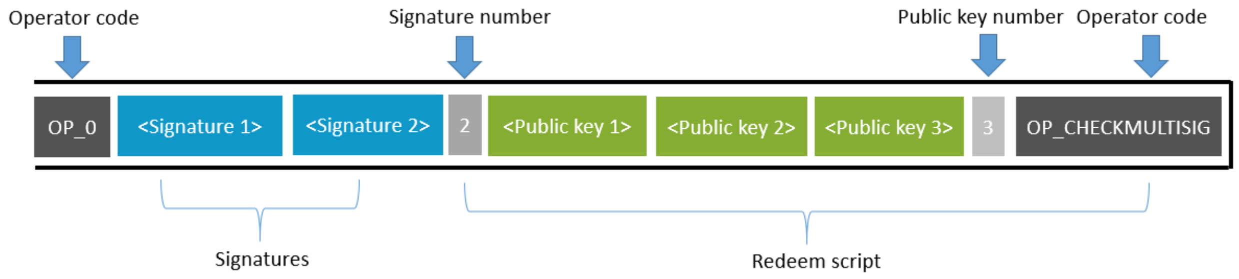


Figure 3. Signature stack with signatures.

3.1.3 Confirming signature stage

Figure 4 outlines all the activities involved in the confirming signature stage. Firstly, in the transaction distribution phase starting at time T1, the raw transactions are broadcasted to the miners through the P2P network. This phase continues until T2, when the miner node may stop accepting new transactions within the block. The miner then checks the endorser in each transaction, which is called transaction verification phase. An example of checking one endorser in transaction is shown in Table 6.

Table 6. Signatures validation algorithm

```
Algorithm 4 Signature List Validation
Input:  The condition we defined in Initialization Phase
Input:  The Signature List ○ m
Input:  The Public Key List (revealing from the redeem script)
Output:  The Validation Result (true or false)
1:    R' = (r1', r2') = s-1F(m)G + s-1 rQ
2:    r' ←r1' mod n
Return: r== r'
```

The endorsement would be correct, if the executed result were 'true'. Then, the miner starts to find a random string to match hash collision condition. Once the first miner found a random string, it would broadcast to other miners on the P2P network. Subsequently, the other miners check whether the solution to the puzzle is correct. If an adequate amount of them grant their approval, the block is cryptographically added to the ledger[24]. And the miners move on to the next set of the transaction in the last phase, i.e. New Blockchain Distribution Phase.

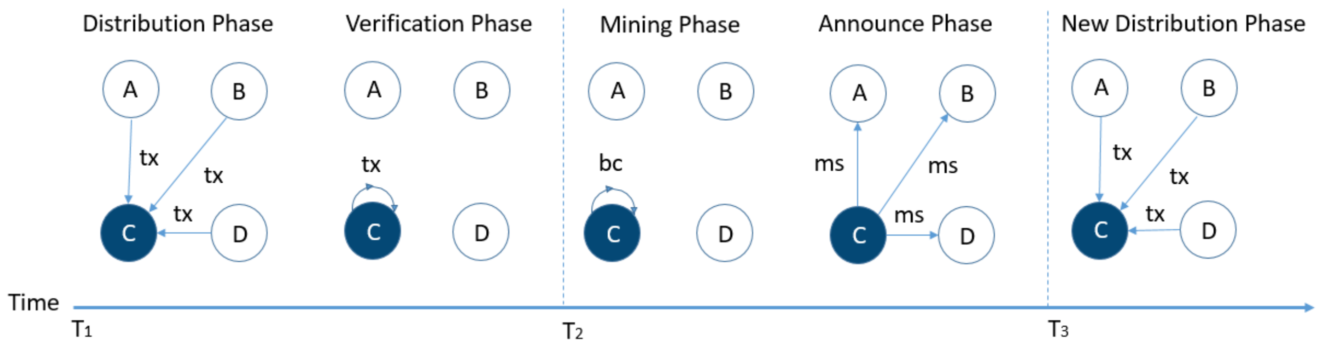


Figure 4. Confirmed signature phases.

3.2 Secured certificate revocation

Certificate revocation is inevitable. Some circumstances such as registration error or individual fraud require revoking the certificate[25]. In this section, we gave the necessary background, then, described the revocation approach in the *Blockcerts*, with emphasis on the innovation method.

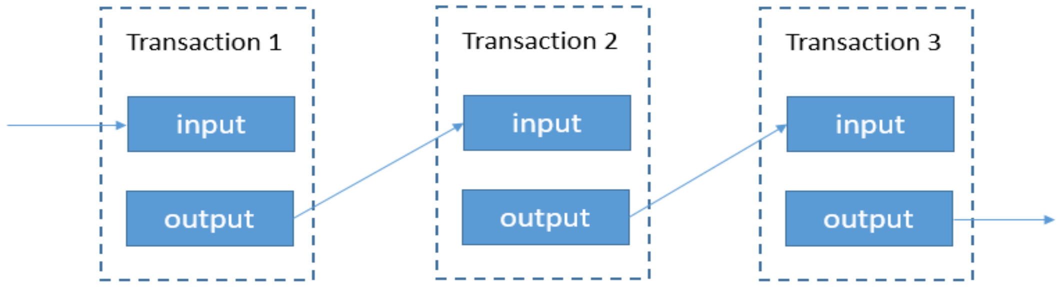


Figure 5. Transaction chain in the blockchain.

As is shown in Figure 5, each transaction has outputs and inputs. The outputs are identical to the new transaction inputs for next transaction, and the input connected the source of the transaction address, by which we can trace back the prior transaction address.

In *Blockcerts* version 1.0, the revocation is designed as spending a specific transaction output. Per the protocol of Bitcoin, only the private key owner can spend the transaction. For revoking the certificate the awarding body needs to spend the transaction using the private key. Thus, Storing the private keys is compulsory, which has the risk of leaking them. What's worse, this revocation is costly because each transaction has to experience over two transfers.

In *Blockcerts* version 2.0, the researcher in MIT improved mechanism by adopting the certificate revocation URL, which had the same work principle with Certificate Revocation List in X.509 Public Key Infrastructure[26]. The awarding body provides a URL for obtaining a list of certificates that have been revoked. The verifier can compare the local document with suppressed ones. This mechanism is recognized and more universal but unreliable. Because, the URL which is fixed and embedded in the issued certificates, cannot commit working persistently or out of being hijacked.

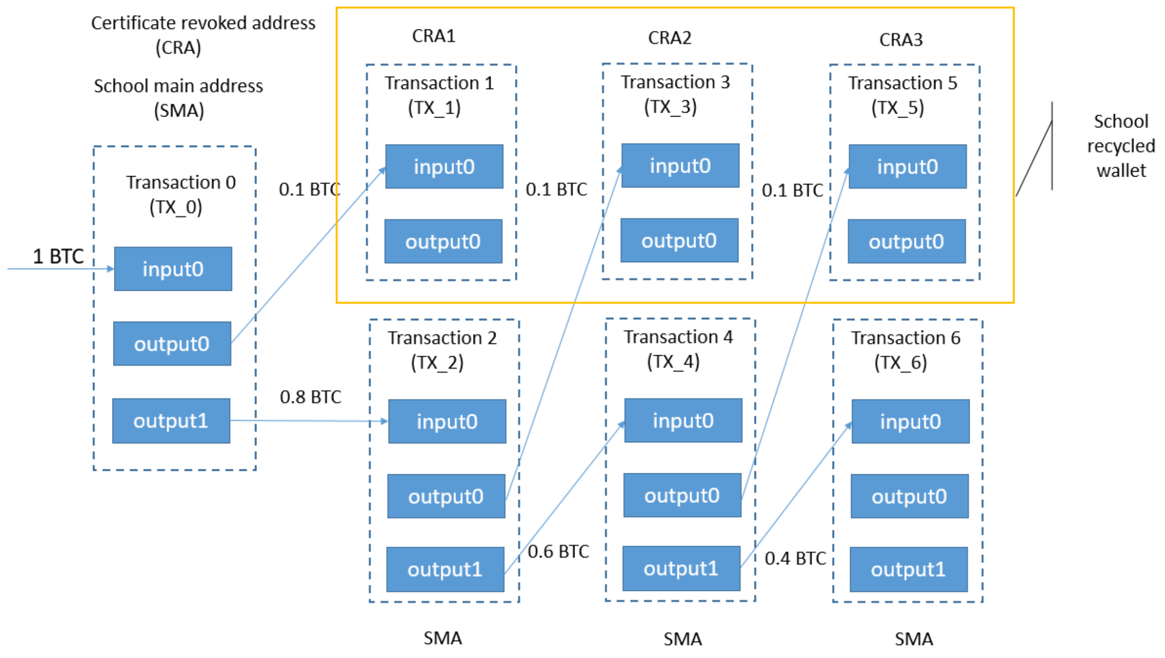


Figure 6. Certificate revocation example.

Our revocation approach is inspired by the approach of *Blockcerts* version 1.0; we try to solve their issues: unsecured keys and uneconomical transaction. We checked the input state rather than output state in the transaction. Alternatively, we care which address transfer money to the revoked address, but the approach of *Blockcerts* version 1.0 focus on whether the money of the revoked address is spent.

Revoking the certificate is performed by means of sending the BTC to the embedded Bitcoin address. The awarding body generated a wallet which contains all the issuing addresses and distributed the issuing address to each certificate. Figure 5 shows the progress of revoking the certificate. If the awarding body wants to revoke the certificate which they issued before, they only need to send Bitcoin to the embedded address of the document using its own main address. Based on our design in last section, the main address is controlled by the majority of the members. Thus, the revoking progress is also democratic.

Confirmation of the revocation of the certificate is performed by checking the input of the transaction. Specifically, only if the embedded Bitcoin address is found on an exists transaction and more importantly, the owner of the output address is proved to be the institution, can the certificate be revoked.

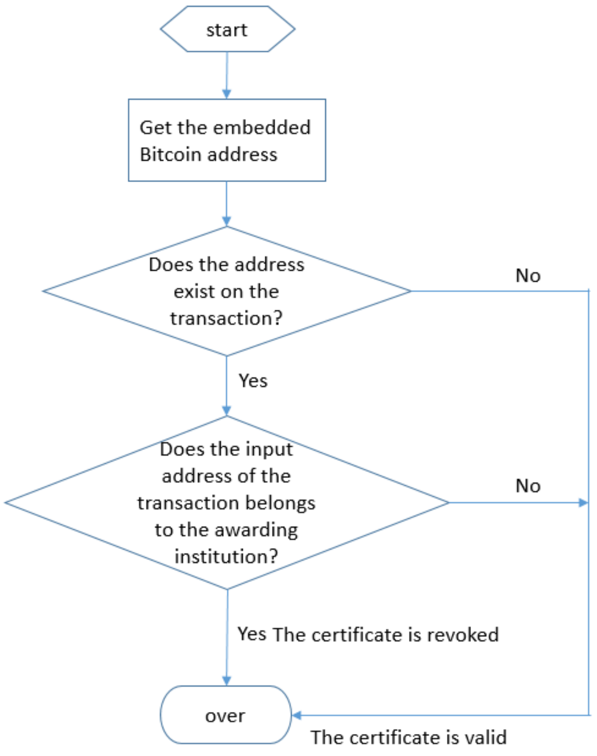

start
Get the embedded Bitcoin address
Does the address exist on the transaction?
No
Yes
Does the input address of the transaction belongs to the awarding institution?
No
Yes The certificate is revoked
over
The certificate is valid

Figure 7. Flow chart of verifying the revocation.

As is shown in Figure 7, we summarized the verifying progress as follows:1. Fetched the embedded revocation address in the issued certificate.2. Checked the availability of embedded revocation address in the issued certificate. 3. Analyzed the result, the certificate can be revoked only if the address is in one transaction and its input contains the main address. On the contrary, the certificate is still valid, if the address is never used or the input transaction does not contain the main address.

3.3 Trusted federated identity

Traditionally, to proof a digital academic certificate as authentic, the efficient and standard way is to prove that this digital academic certificate has been digitally signed by the institution that had created it. In another word, the signer signs the file using the private key and the verifier who purposes to check the authenticity of the certificate can verify the signature with the public key of the signer[27]. However, the signed signature is commonly merged in the certificates for self-sovereign. , this merged signature will be invalid once the institution changes their public key with the certificate authority.

In this section, we explore a new solution named trusted federated identity to solve certificate authenticity problem. This solution is different from the traditional PKI-based way, which is decentralized, trustable, flexible and usable. We describe the trusted federated identity protocol in two parts: the trusted path and federated identity.

3.3.1 Trusted path

The trusted path is to associate the certificate with the issuing address. The trusted path is a route connecting all the necessary resources, including the Merkle tree, the transaction, the blockchain. etc. The entire path is shown in Figure 1, outlined as follows:

the hash of certificate is in the Merkle tree
the Merkle root is in the transaction
the Blockchain has confirmed the transaction
The issuing address did launch the transaction
the issuing address belongs to the particular institution

Each resource in the trusted path interlinked each other closely, for this reason, any error in the path may result in the entire process failure. Thus, to prove that a particular organization does issue the certificate, we only need to demonstrate the correctness of the whole path.

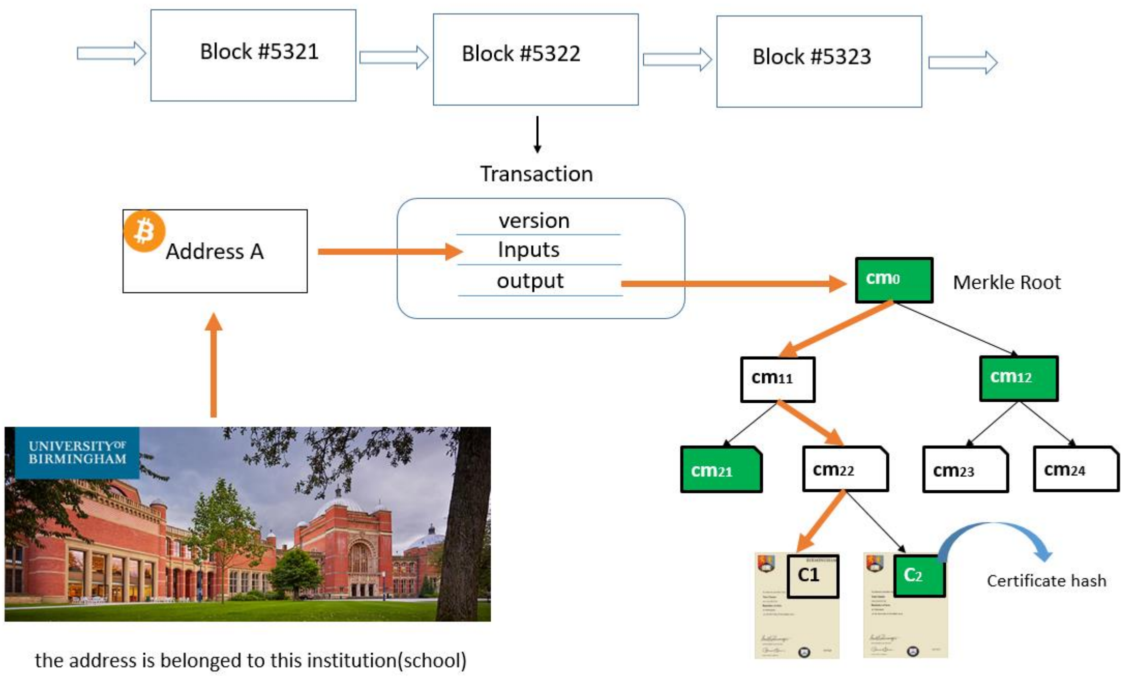


Figure 8. Trusted path with blockchain.

So far we have proved that the certificate indeed issued from the particular address. In the following stage, we will discuss how to determine the issuing address belongs to the particular institution.

3.3.2 Federated identity

The federated identity is to prove that the issuing address belong to the particular institution. The trusted federated identity designed as a JSON-based text that contains all the metadata needed for proving ownership, Especially, the document provided a list of BTC address with its valid timestamp and authorized IP, and a set of identity claim service endpoints which is necessary for pointers to launch trusted interactions with the identity owner. Among the list, the address is to establish authenticity; the timestamp is used to backdate the content once the private key was changed or leaked; the authorized IP is to bind the address with the IP in the institution, last but not least, the identity claim is crucial for establishing the ownership of the address. The complete protocol is shown in Table 7.

Table 7. Federated identity protocol

```
{
        BTC Address List:{
        "address_0": "redeem script 0+ valid timestamp + authorized IP"
        "address_1": "redeem script 1+ valid timestamp + authorized IP"
        ………
        },
        BTC Address List Hash: BTC_Address_List _Hash,
        Identity Claim 0: {
              signature: Well-Known URLs + pubkey1_signature.html
        },
        Identity Claim 1: {
              signature: Well-Known URLs + pubkey2_signature.html
        },
        Identity Claim 2: {
              signature: Well-Known URLs + pubkey2_signature.html
        }
}
```

To verify the ownership of the issuing address, the verifiers only need to download the JSON-based text from the official website and seek the corresponding address by the certificate issued time. Besides, to increase credibility, we came up with a efficient way to authenticate this JSON files. That is to say, we public the signatures to different Well-Known URLs, and these signatures can be verified by the redeem script in the BTC Address List.

Typically, the Well-Known URLs is the recognized social media or official news which attached the signature signed by the member of the academic committee. When attaching the signature on Well-Known URLs, several factors are necessary to be taken into account. The primary cause is that the well-known URLs claimed the relationship between its content and the attached signature. Meanwhile the redeem script list in BTC Address List make it possible to verify the signature, which provides circumstantial evidence to prove the ownership of the redeem script. Another reason is that the subject(news, statement) in well-known URLs is decentralized and widely accepted, which is tough to change the text once it had published.

4. Implementation and Evaluation

We created a prototype implementation employing Java and JavaScript which used the three technical solutions mentioned above. In addition, we extensively evaluated the application regarding operational security, data security, network security and protocol security, which are outlined in "security evaluation" section.

4.1 System Implementation

In this section, we discuss the implementation from the point of view of prototype workflow, system architecture, database architecture and coding implementation. The prototype workflow reveals how the system works from the user's point of view. The system architecture and database architecture show how the system is designed from the engineering point of view. The coding implementation describes the code structure and the core framework.

4.1.1 Prototype workflow

To implement the above design and analysis, we created the prototype model workflow for four main roles, including student, checker, issuer, system and employer. The prototype workflow is shown in the figure below.

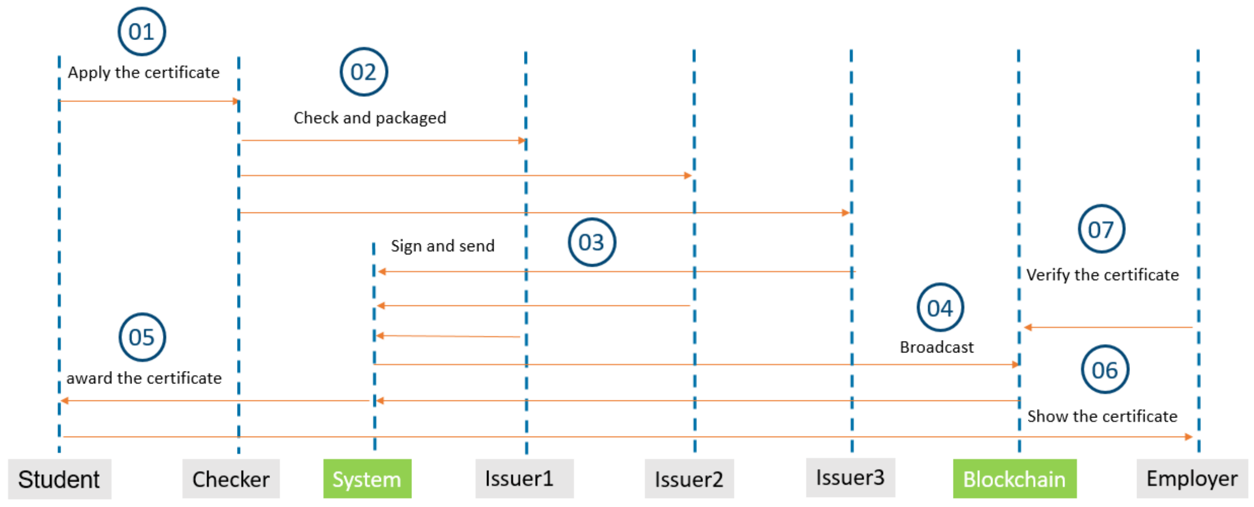


Figure 9. Workflow of prototype.

Specifically, the prototype workflow is as follows: Firstly, the student applies to the school for a credential, and the certifiers check the students' information and merge the credential with a Bitcoin transaction once it is approved. Then the majority of the academic committee members sign it with their private keys. After that, the system broadcasts the transaction which contains the Merkle root for all the certificates. Following the above step, the student receives a JSON-based certificate once the transaction is confirmed by the miners. In the next stage, the student provides the JSON-based certificate to the employer, when he or she applies for a job. Lastly, the company verifies the certificate via access to the Blockchain and checks the authentication code.

4.1.2 System architecture

The system briefly consists four components in our implementation: verification application including federated identity, issuing application involving multi-signature and BTC-address based revocation, Blockchain and local Database adopted by MongoDB.

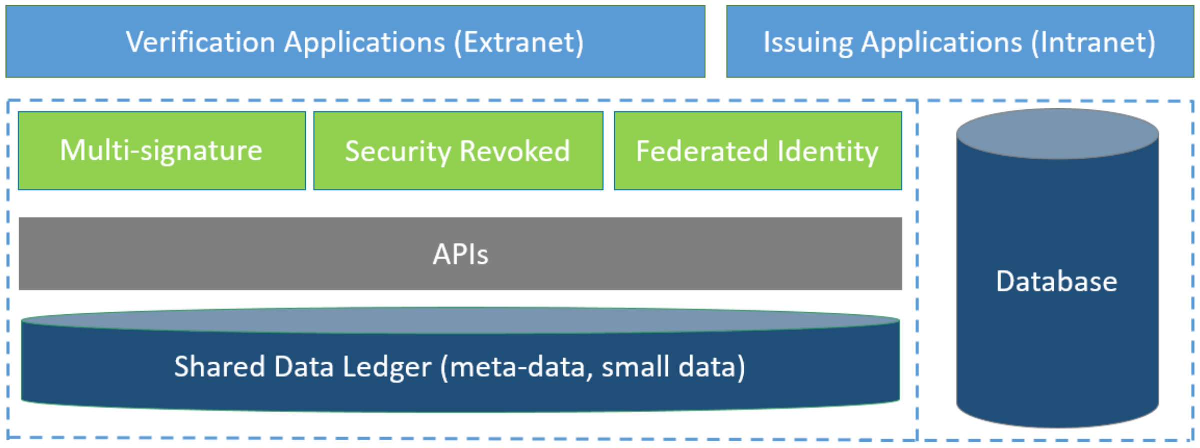

Verification Applications (Extranet)
Issuing Applications (Intranet)
Multi-signature
Security Revoked
Federated Identity
APIs
Shared Data Ledger (meta-data, small data)
Database

Figure 10. System architecture.

The issuing applications are responsible for the main business logic which include the certificates applying, examining, signing and issuing. The issuing applications are designed to merge the hash of the certificate in a Merkle tree and send the Merkle root to Blockchain amidst signing by the majority of community members. Also, the issuing applications involved the revocation of certificate.

The issuing applications are responsible for the main business logic which includes the applying for, examining, signing and issuing of the certificates. The issuing applications are designed to merge the hash of the certificate with a Merkle tree and send the Merkle root to the Blockchain. Also, the issuing applications deal with the revocations of certificates.

The verification application focuses on checking the authenticity and integrity of the certificates that have been issued. It includes two main components: a web-based page and an Android-based application. They use the same mechanism, and fetch the transaction message through the blockchain API and compare the transaction message with the verification data from the receipt. The mechanism can be briefly described in the following way: check the authentication code is valid; check the hash with the local certificate; confirm the hash is in the Merkle tree; ensure the Merkle root is in the blockchain; verify the certificate has not been revoked; validate the expired date of the certificate. Also, it has to be mentioned that for the convenience of sharing the certificates, the Android-based application allows for verification of the documents by scanning the QR code directly.

The blockchain acts as the infrastructure of trust and a distributed database for saving the authentication data. Typically, the authentication data consists of the Merkle root generated using hashed data from thousands of certificates. The MongoDB is employed as our database since the MongoDB successfully manages JSON-based certificates and provides high availability and scalability.

4.1.3 Database architecture

The database has been designed to contain two categories of data: the public authentication data and the private certificate data. The public authentication data is available to the public and released to the blockchain; the private certificate data is stored in the MongoDB where it is securely protected and isolated in the intranet.

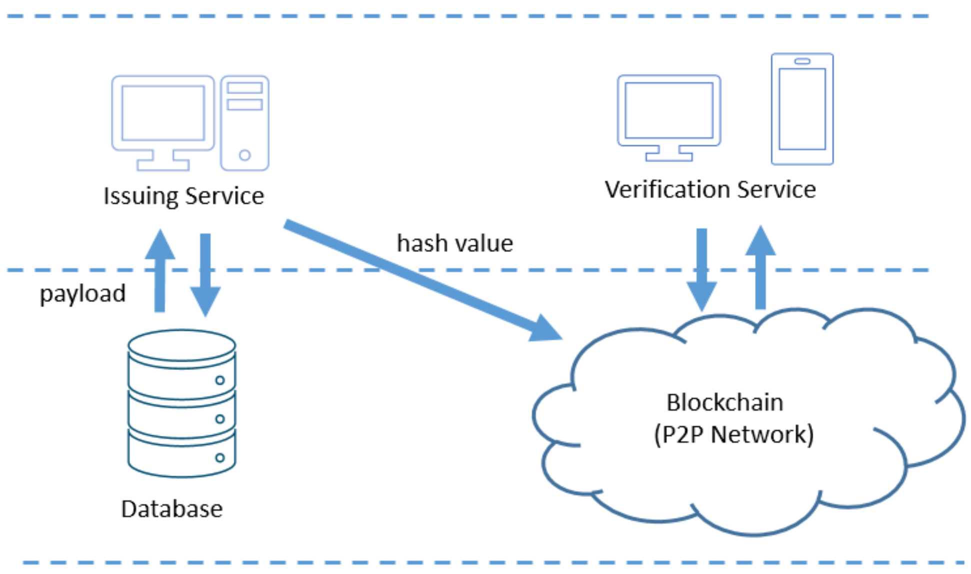


Figure 11. Database architecture.

Figure 11 maps out the top level data flow diagram. It shows that the data flow is unidirectional from the internal areas to the internet. The issuing system reads the certificate from the MongoDB and broadcasts its "point" data to the blockchain. The verification service only needs access to the blockchain to check the authenticity of the certificate.

At the same time, the enterprise MongoDB architecture has been adopted as our local database as shown in Figure 12. In this architecture, the "mongo server" serves as the router to access the primary service, the "configure server" keeps the system working meta data and the "mongo server" saves the core data.

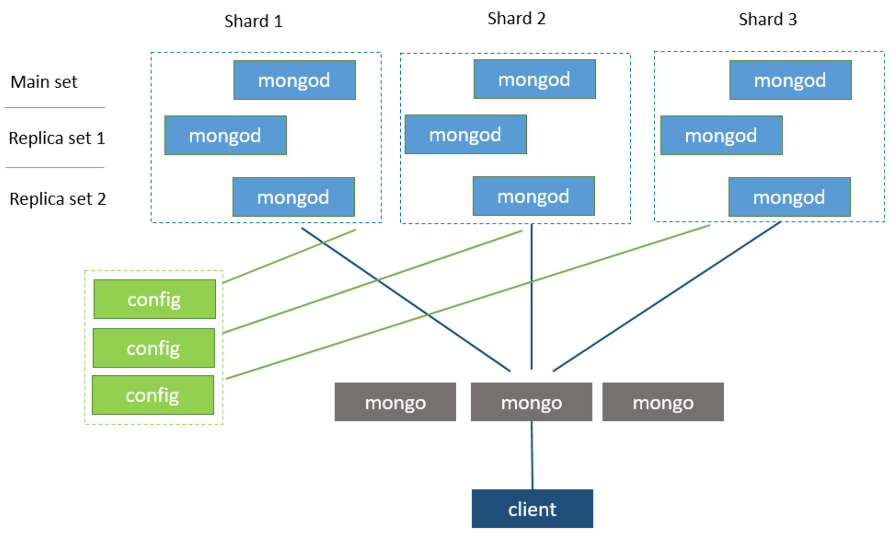
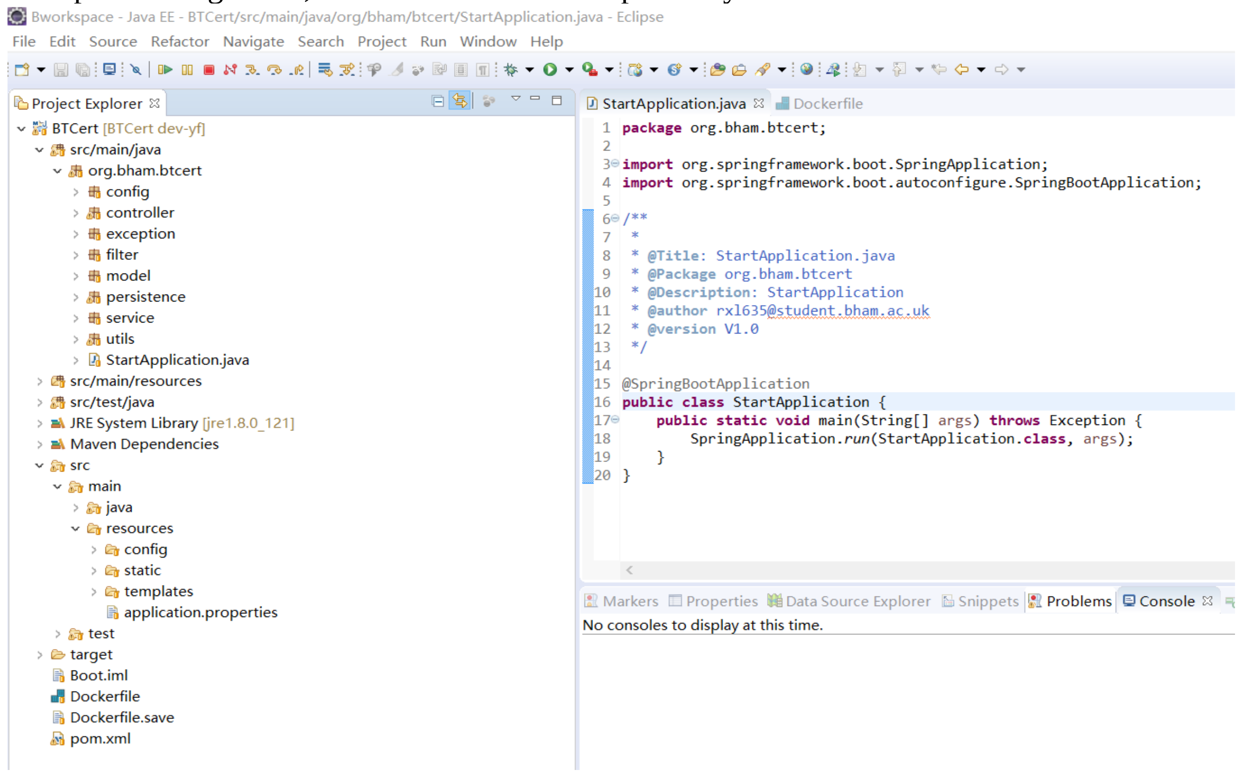


Figure 12. MongoDB Production Cluster Architecture.

4.1.4 Coding Implementation

In this section, we describe the code structure and some code implementations in our project, overall, we show the code structure, the system functionality description and Configuration data description in Figure 13, Table 8 and Table 9 respectively.



```java
package org.bham.btcert;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 *
 * @Title: StartApplication.java
 * @Package org.bham.btcert
 * @Description: StartApplication
 * @author rxl1635@student.bham.ac.uk
 * @version V1.0
 */

@SpringBootApplication
public class StartApplication {
    public static void main(String[] args) throws Exception {
        SpringApplication.run(StartApplication.class, args);
    }
}
```

```
No consoles to display at this time.
```

Figure 13. Project code structure.

Figure 13 shown us the code structure in our project, which contains the Java code, configure code and the JavaScript code. Among them, Java code is in charge of back-end business on the server. This business includes applying, checking and interacting with the local database. The configure code such as docker file is to config the global setting. The JavaScript code is used to implement multi-signature algorithm and make web pages interactive, also JavaScript code is designed to call the APIs for the blockchain. Lastly, it has to mentioned that all the elements in Figure 13 is our own original code except for the "Maven Dependencies" and the "JRE System library".

The Java code consists of the `config`, `controller`, `exception`, `model` ,`service`, `utils` and `filter` modules. We describe the java code functionality as follows in Table 8:

Table 8. Java package functionality description.

Module	Description
org.bham.btcert.config	Defines the configuration management for access control and security provider.
org.bham.btcert.controller	Defines APIs for the WebView, contains Spring's model-view-controller (MVC) and REST Web Services implementation for web applications
org.bham.btcert.exception	Defines the implementation of exception
org.bham.btcert.filter	Provides the implementation of http filter serve as preventing xss attacks
org.bham.btcert.model	Defines Object Model
org.bham.btcert.persistence	Provides the implementation of BaseMongoTemplate used for connecting the MongoDB
org.bham.btcert.service	Provides the implementation of the service used by org.bham.btcert.controller
org.bham.btcert.utils	Defines utility classes inculding Merkle tree, CHexConver,CryptoUtil

The configuration data to define the running environment, the server, and other devices setting that compose the system and its boundary. Mainly, our system concludes three configuration files which are outlined in Table 10.

Table 9. Configuration data description

Configure File	Description
pom.xml	POM stands for "Project Object Model". It is an XML representation of a Maven project
Dockerfile	A text file that contains all the commands, in order, needed to build a given image in our project
application.yml	Spring Boot configure file to specify the properties

JavaScript code plays an important role in our implementation, because we adopt it to implement multi-signature algorithm and interacting with the blockchain. Table 9 shown the part code which used to sign the signatures and assemble the protocol. Table 10 shown how to interact with the blockchain via APIs. Due to the page limitation, The detail of the core coding is outlined in the repository.

Table 10. Part code of signing signatures.

```
// get current public key from the private key
 pubkeyInfo = btcert.wif2pubkey(privatekey);
  pubkey = pubkeyInfo.pubkey;

// get all the public keys from the redeem script
 scripts = btcert.redeem.script;
 pubkeys = btcert.redeemScript2pubkeys(scripts).pubkeys;

// init signature
for(var i = 0 ; i < txins.length; i++){
 var tran_hash = btcert.getTransactionHash(i);
      sigs = Crypto.util.hexToBytes(btcert.getHashSignature(tran_hash,privatekey))
      btcert._tranction_signatures.push({"publickey":pubkey+tran_hash,
           "signature":btcert.getHashSignature(tran_hash,privatekey)})
}

// push all the signature and the redeem script to the stack.
btcert._tranction_inout_temp = [];
 for(var i = 0 ; i < txins.length; i++){
      var scriptResult = [0];
      var tran_hash = btcert.getTransactionHash(i);
      for(var j = 0 ; j < pubkeys.length ; j++){
           var pubkey = pubkeys[j];
           var sig = getsignature(pubkey+tran_hash);
           if(sig != ""){
                sigb = Crypto.util.hexToBytes(sig);
                scriptResult = scriptResult.concat(btcert.numToVarInt(sigb.length));
                scriptResult = scriptResult.concat(sigb);
           }
      }
      scriptResult.push(76);
      var scriptBytes = Crypto.util.hexToBytes(btcert.redeem.script); // redeem script
      scriptResult = scriptResult.concat(btcert.numToVarInt(scriptBytes.length))
      scriptResult = scriptResult.concat(scriptBytes);
      var txin = {  hash:txins[i].hash,index:txins[i].index,
                    scripts:Crypto.util.bytesToHex(scriptResult),
                    scriptBytes:scriptResult,sequence:txins[i].sequence
                 }
      btcert._tranction_inout_temp.push(txin);
 }
```

Table 11. Code example of interacting with the blockchain.

```
// get the Unspent Transaction from the bitcoin transaction.
btcert.getUnspentTransaction = function(callback){
     var redeem = this.redeem;
     $.ajax ({
          type: "GET",
          url: "https://chain.so/api/v2/get_tx_unspent/BTC/"+redeem.addr+"?unconfirmed=1",
          dataType: "json",
          error: function(data) {
          },
          success: function(data) {
               if((data.status && data.data) && data.status=='success'){
                    callback();
               } else {
                    error();
               }
          },
          complete: function(data, status) {
               completeAPIs();
          }
     });
}
```

4.2 Security evaluation

To assess the security of the system, we conducted extensive evaluations of the following aspects: operations security, data security, network access security and protocol security. The results of these assessments are encouraging. Specifically, our system is able to handle almost all the botched operations; the core data of certificates is secured and stable, the authentication data is decentralized and trustworthy; the system succeeds in reducing the attack surface and protecting the security of intranet; the service is distributed and reliable. The multi-signature protocol is democratic and secured with the reasonable configuration, and our blockchain-based revocation is secured and economical.

4.2.1 Operations security

The behavior of users is hard to predict. Inevitably, the users will make some mistakes or perform a wrong action while using the system. Thus, an evaluation considering the handling of user errors is necessary. We conducted the evaluation of the following botched operations: unauthorized user access; certificate issued and revoked inappropriately; leaking the login password; leaking the private key and password

User unauthorized access.
The system is implemented based on RBAC[28] model. From a functional perspective, the operations representing specific actions such as apply, edit, reject, are only allowed to be performed by the users with the corresponding roles such as student, checker, issuer, admin. To have a better understanding of RBAC mode in our system, we defined this mode as follows:

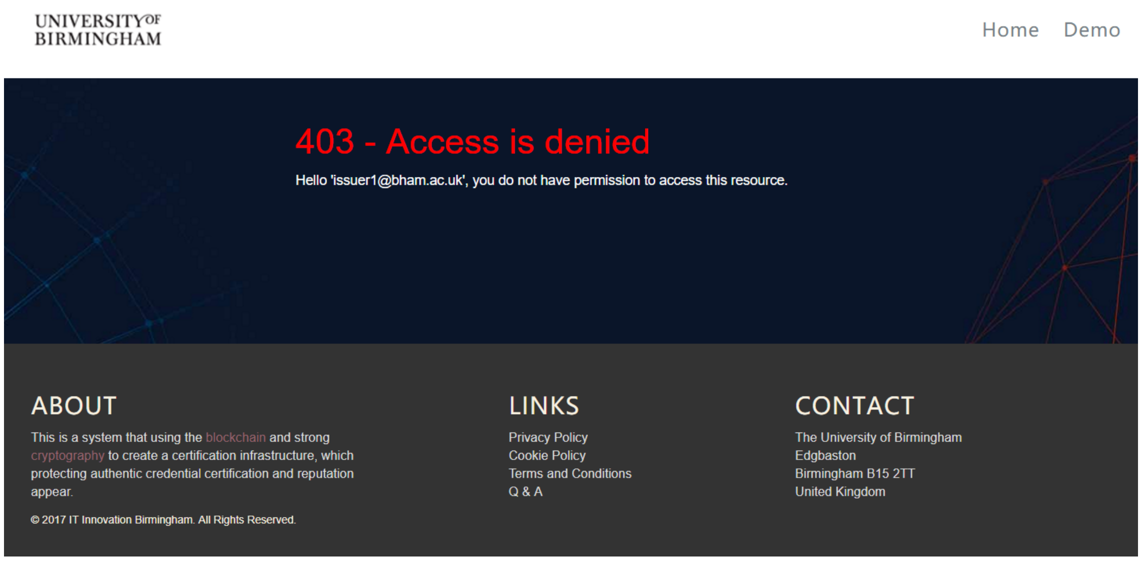
Table 12. RBAC model definition

resource-subject(s: subject) = {the result subject s}
role(R: role) = {all the role associated the subject s}
authorized- resource -role(arr: role) = {the role granted to opera the subject s}
authorized-user-role(aur: role) = {the role connected the user u}
subject-user(u: user) = {the user in the set of role r}

For each operation of user, role and subject, it must satisfied only if:

$\forall$ s:subject, u: user, r:role:
$\exists$ subject-user(s) $\in$ (all- role (R) $\wedge$ authorized - user -role(aur) $\wedge$
authorized - resource -role(arr)) =>
resource-subject s $\in$ authorized - resource -role(arr)

If the operation does not meet the assumptions above, or if a user attempts to access the subject/function which does not belong to him or her, he or she will be rejected by the service directly. Figure 15 presents an example of the unapproved access attempt.


UNIVERSITY OF BIRMINGHAM

403 - Access is denied

Hello 'issuer1@bham.ac.uk', you do not have permission to access this resource.

ABOUT

This is a system that using the blockchain and strong cryptography to create a certification infrastructure, which protecting authentic credential certification and reputation appear.

© 2017 IT Innovation Birmingham. All Rights Reserved.

LINKS

Privacy Policy
Cookie Policy
Terms and Conditions
Q & A

CONTACT

The University of Birmingham
Edgbaston
Birmingham B15 2TT
United Kingdom

Figure 14. Unauthorized error page.

Certificate issued and revoked inappropriately

The certificate issue and revocation process are very secure. This has been achieved in the following three ways. First, the "issue and revoke key" is distributed among different members of the academic committee, and the members are not allowed to access these critical operations individually, what's more, the system records and monitors all the activity related to issuing and revoking certificates. Lastly, the process of certificate issuing and revoking is democratic, requiring the majority of the academic committee members to sign.

Leaking the login password

Leaking the password is a risk in the system, but the system is still secured. A leaked password is difficult to exploit for hackers, due to the lack of the private key that plays an essential role in issuing or revoking certificates. What's more, the system is not allowed to store the private key, and it provides enough protection mechanisms for the saved passwords and credentials. Thus, leaking of passwords would make the system vulnerable, but it would not provide a fraudster with any useful information.

Leaking the private key and password

Leaking the password as well as private key would be a threat of to our system as it would, from the functional point of view, it allows the hacker to access to all the operations and resources of the system. However, in terms of the business logic, we can certainly solve this problem, since each transaction needs to signed by the majority of the academic committee members.

In the most extreme case of all, of the academic committee members leaking the password and the private key at the same time, there is no doubt that the system would be paralyzed. It would be impossible to prevent an attacker from awarding "legal" certificates. In other words, even if the awarding institution revoked the certificate in public at a later date, the independent verifier would have no idea about the difference between a valid and an invalid certificate, except for additional proof from additional authorities about the time when the transaction was performed.

Fortunately, we have dealt with these serious issues using our new solution called the federated identity with a timestamp. Specifically, an independent verifier only needs to verify the timestamp and the principal owner, in another word, such an auditor is only required to confirm the certificate awarding institution owned the key when the academic certificate was granted.

In a word, we successfully deals with the majority of incorrect operations via our system.

4.2.2 Data security

In this section, we carry out an in-depth exploration of data security, and particularly, we discuss data security from four viewpoints: data confidentiality, data integrity and tamper-proofing, data access security, and data availability.

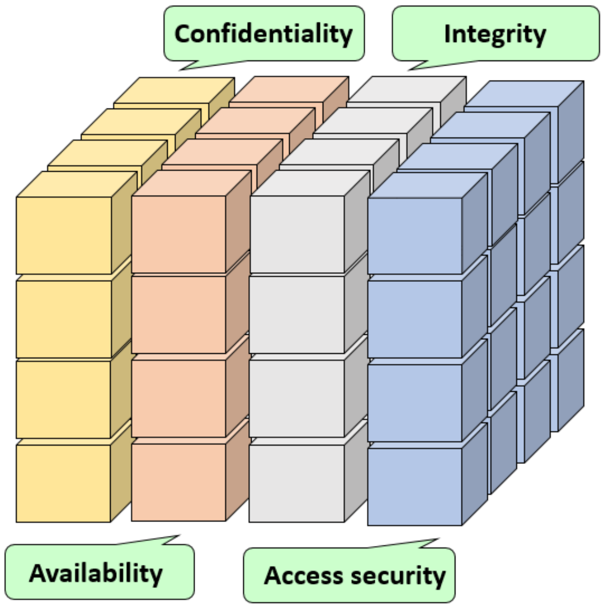


Figure 15. Four evaluation dimensions of data security.

Data confidentiality

The confidentiality of the certificate data is secured by the Spring security framework and MongoDB security scheme, with any sensitive data such as password data encrypted through the SHA256 algorithm. We list an encrypted data example as follows. The authentication information is a set of random numbers and ensuring its confidentiality is unnecessary.

```
[
  id=597157955a3e502eb4153127,
  u_name=issuer1@bham.ac.uk,
  u_passwd=$2a$12$68EMrGC4gcpmyPV3nryAx.NAJCVI/OWnCuOggWFOpMRQRBXy1o17K,
  role=ROLE_ISSUER,
  state=1
]
```

Data integrity

Our storage scheme provides the features of data integrity and tamper-proofing naturally: as the authentication data acts as the “point” of the document data, any illegal operation such as modifying the existing records in a local database by the service or manipulated database will be detected.

Data access security
Data access security crucially involves different users or roles having separate data access permissions. The data is only accessible to authorized users. By employing the RBAC mode mentioned in the last section, our system fully meets this requirement. For example, the checker may have the ability to remove and update all the certificate data, while the student may only have "read" rights for their certificate due to their lack of access to a particular authority within the system.

Data availability
As mentioned in above section, a standard MongoDB production cluster consists of the "mongo" instances, the "data" instances and the config instances. We will discuss every potential failure scenario to highlight the data availability of our system.

- The "mongo" instances become unavailable
As can be shown in Figure 13, we commonly set over two servers as "mongo" instances, even one or two servers are unavailable, the remainder can continue to allow the service access the database. Besides, since the "mongo" instances do not maintain persistent state or data, the whole database will not lose any state or data after the restart.

- The "data" instance become unavailable
the replica sets in the group, provide high availability using automated failover. This mechanism allows a secondary node convert to primary if the current primary node becomes unavailable.

- The "config " instance become unavailable
If the config instance becomes unavailable, the cluster’s metadata becomes read only. However, the "data" node is still readable and writeable[29].

In short, We have made a series of assessments of data security. The assessments result indicates that our system is secure enough to for practical using.

4.2.3 Network security

Currently, it's dangerous to open server for receiving the internet request directly[30]. Most companies buffer their contact with the outside world by employing a DMZ. The DMZ create a security gap between intranet and internet. Ideally, to access internal resources, every request has to go through this DMZ[31]. However, even DMZ has the most stringent rules. there is still the probability that hacker from the outside still can access the internal side.

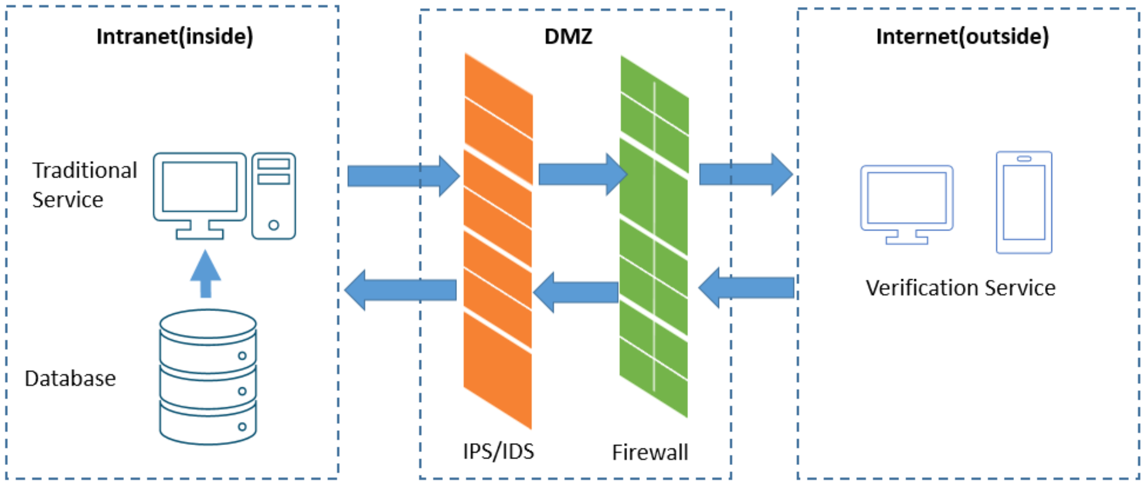


Figure 16. Traditional certificate authentication network flow diagram.

In our project, our core database and issuing service run on the internal network that is an envelope environment, in other words, our deployed environment is physically isolated from the outside. A hacker from the outside has no possibility of accessing the internal side except for physical intrusion. This is much more secure than traditional service running inside a DMZ.

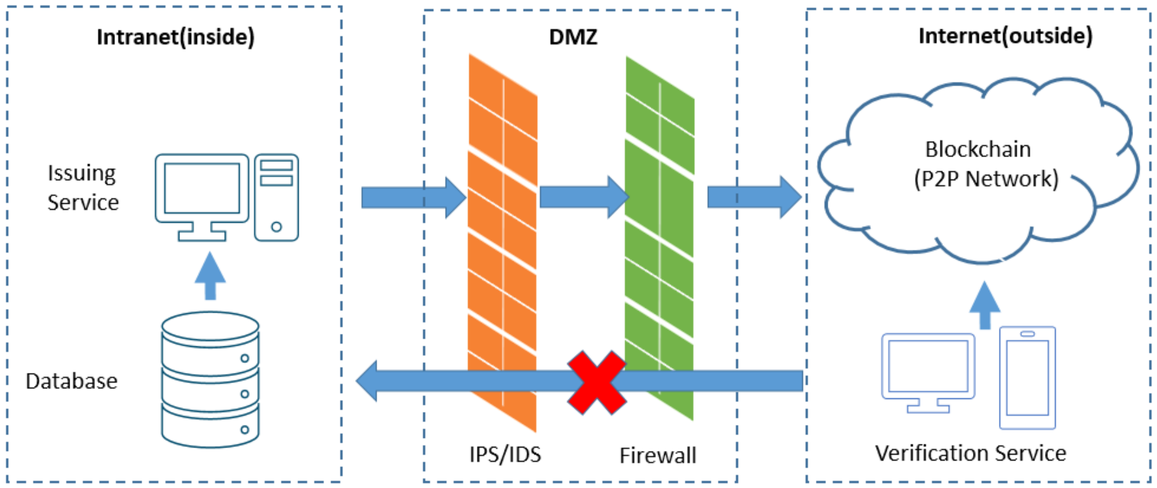


Figure 17. Blockchain-based certificate authentication network flow diagram.

4.2.4 Protocol security

In this section, we focus on the protocol security which proposed in the section of "Cryptographic protocol design". To set up a secured configuration and of multi-signature protocol, we did two experiments to verify the two parameters' effects on signing progress. Also, we evaluated the revocation mechanism from the aspect of reliability, security, cost .etc.

Multi-signature secure evaluation
As mentioned in section of "Multi-signature scheme", the multi-signature is the mixture of all the necessary public keys and required conditions, specifically, an M | N address, where N is the total

number of public keys and M is the minimum number of private keys required for validation. Note that we used payment successfully to represent the issuer certificates in all three experiments.

In experiment I, the parameter M is a fixed number and N is set to be a varied number. we created five groups of combined addresses that joined different numbers (3,5,7,9,11) of original ECDSA public keys. As is demonstrated in Table 1, these combined addresses are represented as a0, a1, a2, a3, a4 and all required two private keys to validate when redeeming the coins. Then we transferred some coins to these addresses and used these addresses to assemble raw transaction strings by attaching the Merkle root. After that, we used two private keys to sign every raw transaction string while broadcasting them to make a payment to an appointed address (using appointed address for recycling these coins). We found that all the transactions satisfied the corresponding conditions that have successfully been accepted by the blockchain. (Results are demonstrated in Table 14)

Table 14. Result of experiment I

Addresses	Required Number M	Total number N	Executed results
a0	2	3	Accepted
a1	2	5	Accepted
a2	2	7	Accepted
a3	2	9	Accepted
a4	2	11	Accepted

Note: “Accepted” means the transaction is accepted by the blockchain. “Rejected” means the transaction is rejected by the blockchain.

In the experiment II, the parameter N is a fixed number 7 and M is set to be a varied number that ranged from number 1 to 7, we applied the same steps as experiment I and providing two private keys to sign the raw transaction strings. The result is shown in Table 15.

Table 15. Result of experiment II

Addresses	Required Number M	Total number N	Executed results
a0	1	7	Accepted
a1	2	7	Accepted
a2	3	7	Rejected
a3	5	7	Rejected
a4	6	7	Rejected

Note: “Accepted” means the transaction is accepted by the blockchain. “Rejected” means the transaction is rejected by the blockchain.

The results of these two experiments suggest that whatever the number of N, as long as it has two signatures, it will be accepted by the blockchain. It did not end up being as positive as we had hoped, which was that the majority of the key owners would be needed to sign the raw transaction strings before making a payment. In other words, we wanted the blockchain to accept the raw transaction string once it met the required number of N.

In a word, our results indicate that the designers should consider the democratic conditions as well as the secure conditions when setting up the configuration. Setting the signature threshold with M > N/2 + 1 is compulsory to achieve security against an adversary that might corrupt any minority of the M privacy peers.

Revocation mechanism evaluation
We conduct the evaluations form the aspect of reliability, security, applicability and cost. For easy to compare the mechanism, we use the abbreviation to represent the approach, precisely, the abbreviation of "BV1", "BV2", "OP" represent the method utilized in the version of 1.0 and 2.0 of *Blockcerts* and our project respectively.

When it comes to the reliability, BV1 and OP almost have the same performance but higher than BV2, since BV1 and OP are all based on the BTC transaction state, and the BV2 adopted the certificate revocation list based on the fixed URL, which checking the BTC transaction state is stable than the querying the certificate via URL.

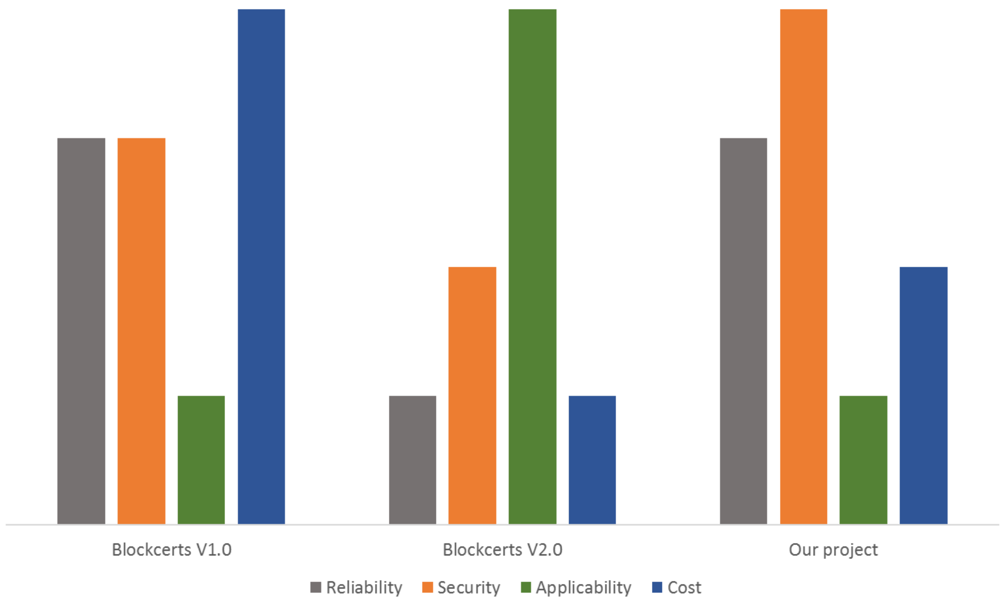


Figure 18. Revocation protocol comparison from four perspectives.

Regarding the security property, OP ranked the first position for the reason that checking transaction state in the blockchain is stable all the revoked address belongs one wallet . Followed by BV1, As the certificate state query service cannot commit working persistently or out of being hijacked. The fact that BV2 has the most security problems is due to storing the all private keys for the awarding body.

Admittedly, BV2 have been widely accepted by the public. Hence it got the highest applicability among all the approaches, while BV1 and OP were reckoned to be less applicable owing to no institution adopting the blockchain-based approach.

In term of the expenditure, BV1 is most costly arisen by each revocation need at least two payments, while BV2 is proved to be the most economical result from it do not need to spend the bitcoin, OP is estimated to be medium because it only needs to experience one transaction for each revocation.

In conclusion. OP succeed in remedying the defect in BV1 and BV2 to a certain extent and become more secured than the other two.

4.2.5 Summary

We conducted extensive evaluations from the operations security, data security, network security and the protocol security. We outlined all the assessment in Table 4. The evaluation result indicates us that our system is secure enough to meet the production requirements.

Table 13. Summaries of security evaluation outcomes.

Safety assessment index	Index category	Index details	Performance
Operations security	Operations security	User unauthorized access.	++++
		Certificate issued and revoked inappropriately	+++
		Leaking the login password	++++
		Leaking the private key and password	++
Data security	Certifcate data	Data confidentiality	++++
		Data integrity & tamper-proofing	+++++
		Data access security	++++
		Data availability	++++
	Blockchain data	Data confidentiality	+++++
		Data integrity & tamper-proofing	+++++
		Data access security	+++++
		Data availability	+++++
Network security	Network security		++++
protocol security	Multi-signature		++++
	Revocation mechanism		++++
Summary			++++

5. Conclusions

In June 2016, the MIT media lab released their blockchain-based credential system which is more secure, more reliable and harder to forge, in contrast to existing technologies that based on the third party arbitration. However, there are some serious authentication defects and vulnerable revocation mechanism which limits the prevalence and application of the project.

In our project, to solve these problems and make its concept more practical, we proposed and designed a set of innovative cryptographic protocols which includes multi-signature, BTC-address-state-based revocation mechanism and trusted federated identity.

Among these protocols, the multi-signature scheme most notably increases the difficulty of forging owing to the fact that each issuing progress is obliged to be signed by the majority of the academic committee members. Besides, it enhances the safety of the private keys storing for the reasons that the private keys are possessed by separated devices and people. Besides, BTC-address-based revocation mechanism improved the stability of the certificate revocation because BTC address is accessible and stable at any time. Moreover, this approach reduced the failure probability of revocation, because the cancellation process adheres the same the multi-signature algorithm, alike, involving several people. Trusted federated identity innovatively proved the authenticity of the certificate through the trusted path and federated identity. What's more, the protocol of our project can be used in other related realms such as digital right protecting and contract proof. Case in point, our protocol enables two companies to attach their contract onto the blockchain with multi-signature, which is different from the traditional third party-based work mode and dispel the worries of forging credentials

Moreover, we implemented a blockchain-based certificate system, which embraced all the above protocols, by utilizing Java and JavaScript. This system has remedied the defect in Blockcerts to a certain extent, which makes the theory of blockchain-based certificate more practicable. Eventually, we conducted a series of security assessment from the perspective of operation safety, data security, network security and protocol security. The assessment outcomes provide compelling evidence that system is secured enough to meet the enterprise application standards.

Lastly, there are some limitations remained to be discussed, albeit, these considerations fall outside the scope of this paper: Our project is based on the Bitcoin blockchain, the maintenance of which relies on thousands of participants in the cryptocurrency ecosystem. Admittedly, it is imprudent to assume that the Bitcoin would work well continuously in the future because myriad types of stakeholders influence blockchain ecosystem or business model. In the years to come, we will adopt multiple blockchain sources such as Hyperledger[32]and Ethereum[33] to eliminate the factors of instability.

6. Appendix

6.1 Project screenshots

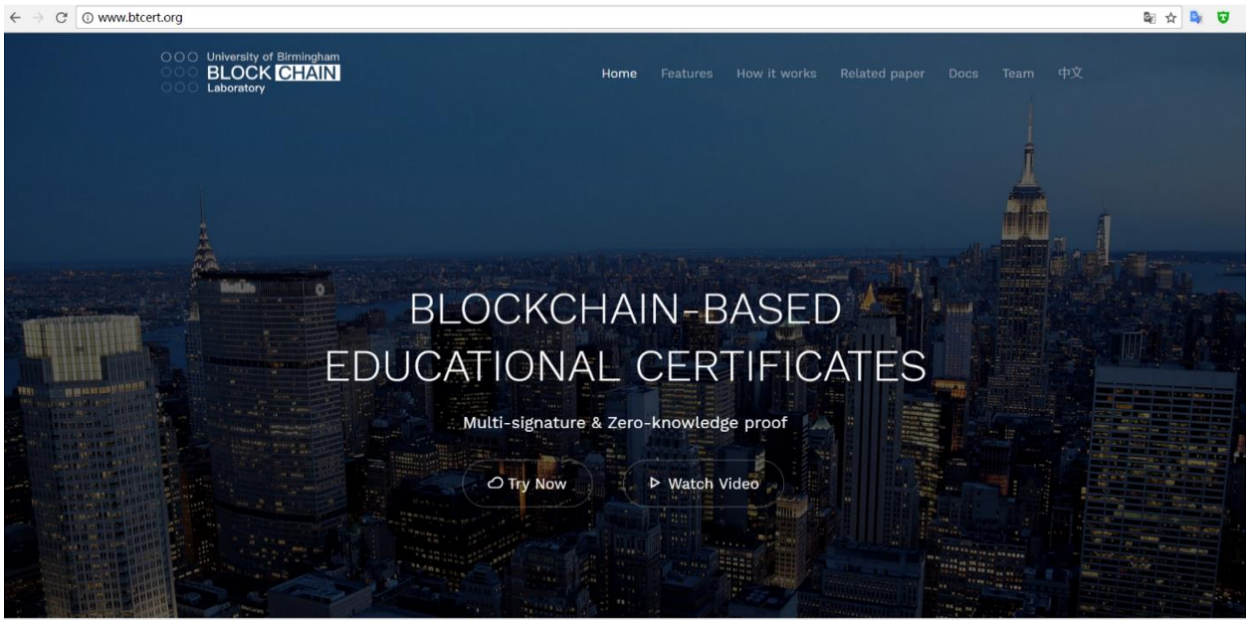
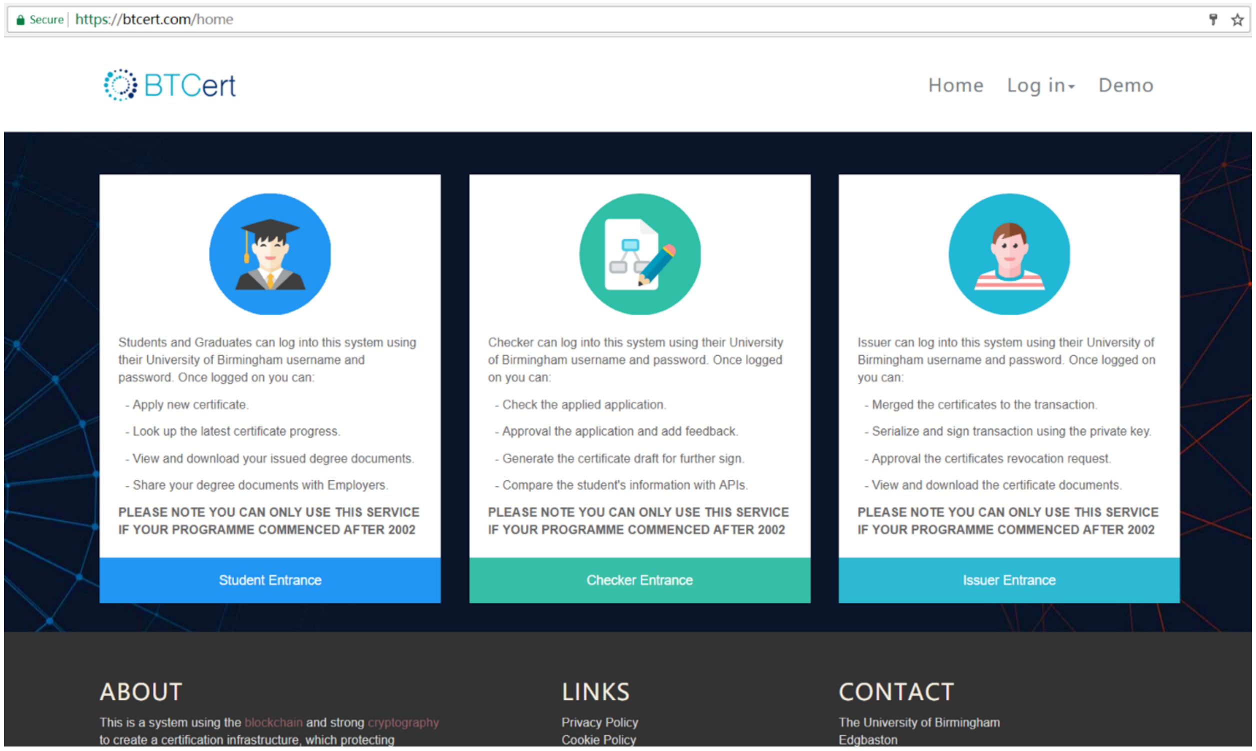

www.btcert.org
University of Birmingham BLOCK CHAIN Laboratory
Home
Features
How it works
Related paper
Docs
Team
中文
BLOCKCHAIN-BASED EDUCATIONAL CERTIFICATES
Multi-signature & Zero-knowledge proof
Try Now
Watch Video

Figure 19. Public home page.



Students and Graduates can log into this system using their University of Birmingham username and password. Once logged on you can:

- Apply new certificate.
- Look up the latest certificate progress.
- View and download your issued degree documents.
- Share your degree documents with Employers.

PLEASE NOTE YOU CAN ONLY USE THIS SERVICE IF YOUR PROGRAMME COMMENCED AFTER 2002

Student Entrance

Checker can log into this system using their University of Birmingham username and password. Once logged on you can:

- Check the applied application.
- Approval the application and add feedback.
- Generate the certificate draft for further sign.
- Compare the student's information with APIs.

PLEASE NOTE YOU CAN ONLY USE THIS SERVICE IF YOUR PROGRAMME COMMENCED AFTER 2002

Checker Entrance

Issuer can log into this system using their University of Birmingham username and password. Once logged on you can:

- Merged the certificates to the transaction.
- Serialize and sign transaction using the private key.
- Approval the certificates revocation request.
- View and download the certificate documents.

PLEASE NOTE YOU CAN ONLY USE THIS SERVICE IF YOUR PROGRAMME COMMENCED AFTER 2002

Issuer Entrance

ABOUT

This is a system using the blockchain and strong cryptography to create a certification infrastructure, which protecting

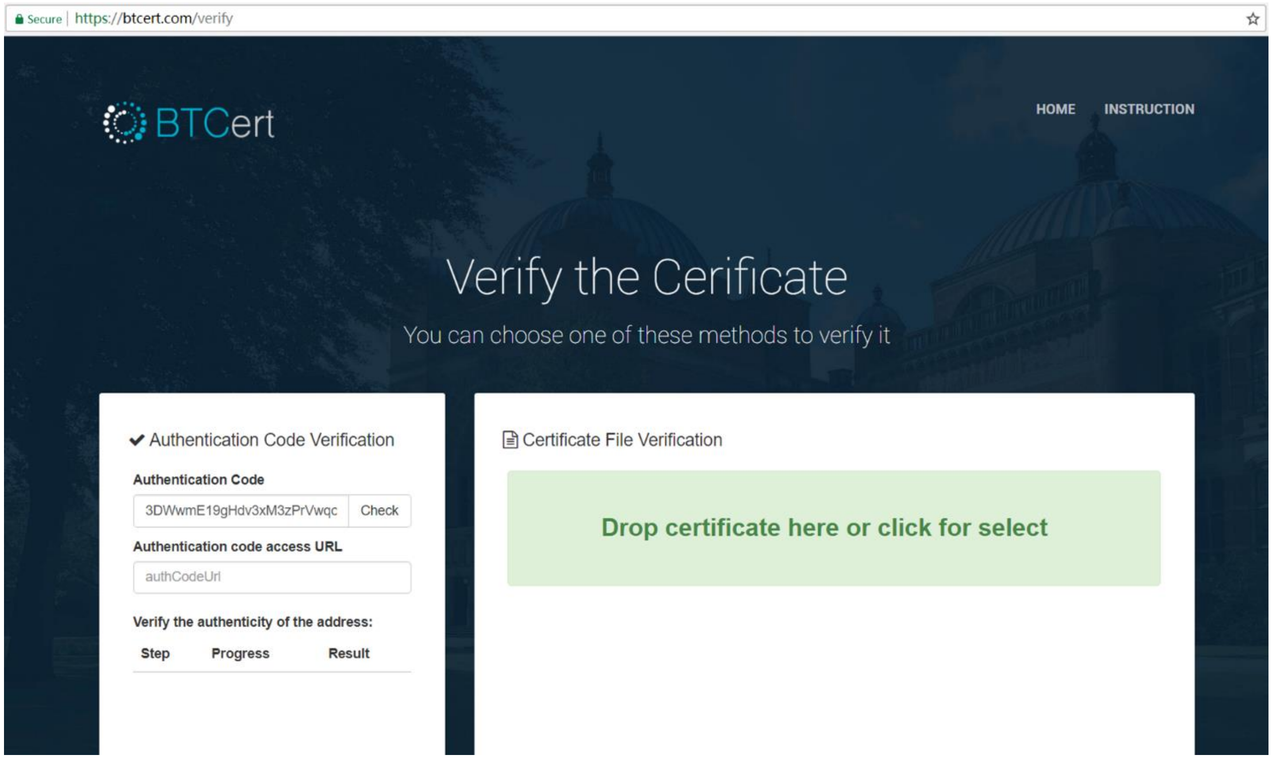
LINKS

Privacy Policy
Cookie Policy

CONTACT

The University of Birmingham
Edgbaston

Figure 20. Intranet home page.


Secure | https://btcert.com/verify

BTCert

HOME INSTRUCTION

Verify the Cerificate

You can choose one of these methods to verify it

✔ Authentication Code Verification

Authentication Code

3DWwmE19gHdv3xM3zPrVwqc Check

Authentication code access URL

authCodeUrl

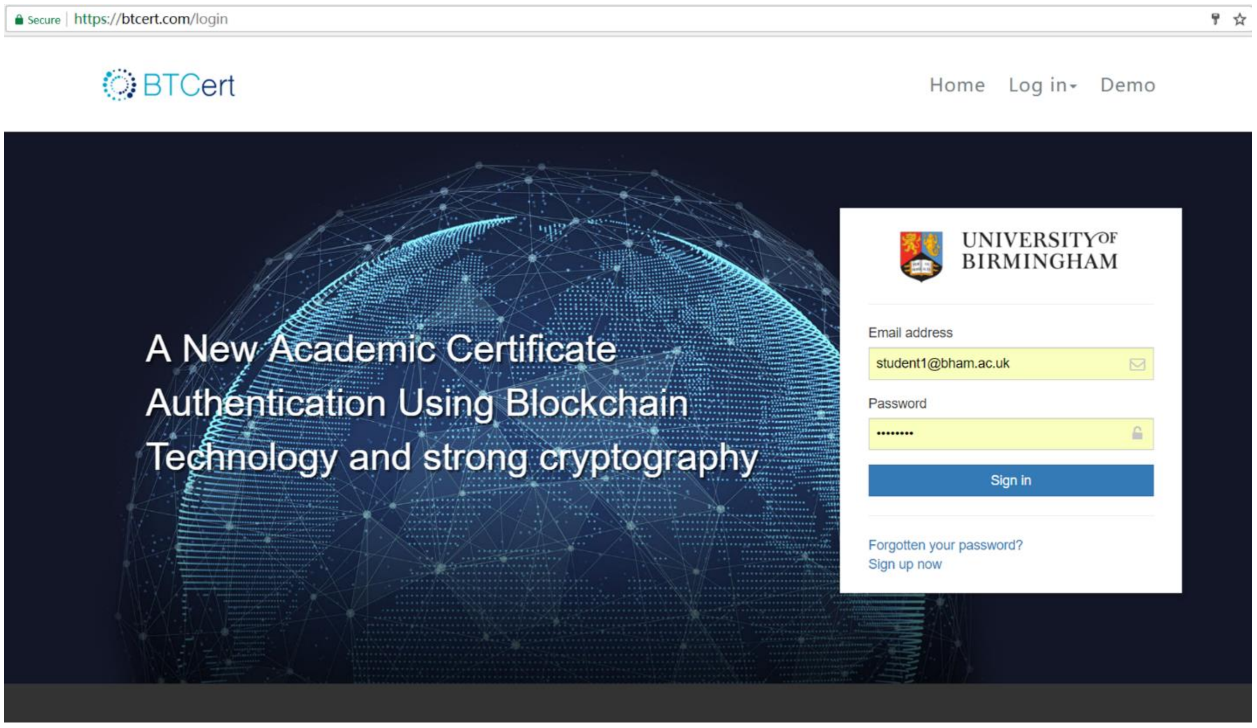
Verify the authenticity of the address:

Step	Progress	Result

Certificate File Verification

Drop certificate here or click for select

Figure 21. Public verifying page.


Secure | https://btcert.com/login

BTCert

Home Log in▾ Demo

A New Academic Certificate Authentication Using Blockchain Technology and strong cryptography

UNIVERSITY OF BIRMINGHAM

Email address

student1@bham.ac.uk

Password

••••••••

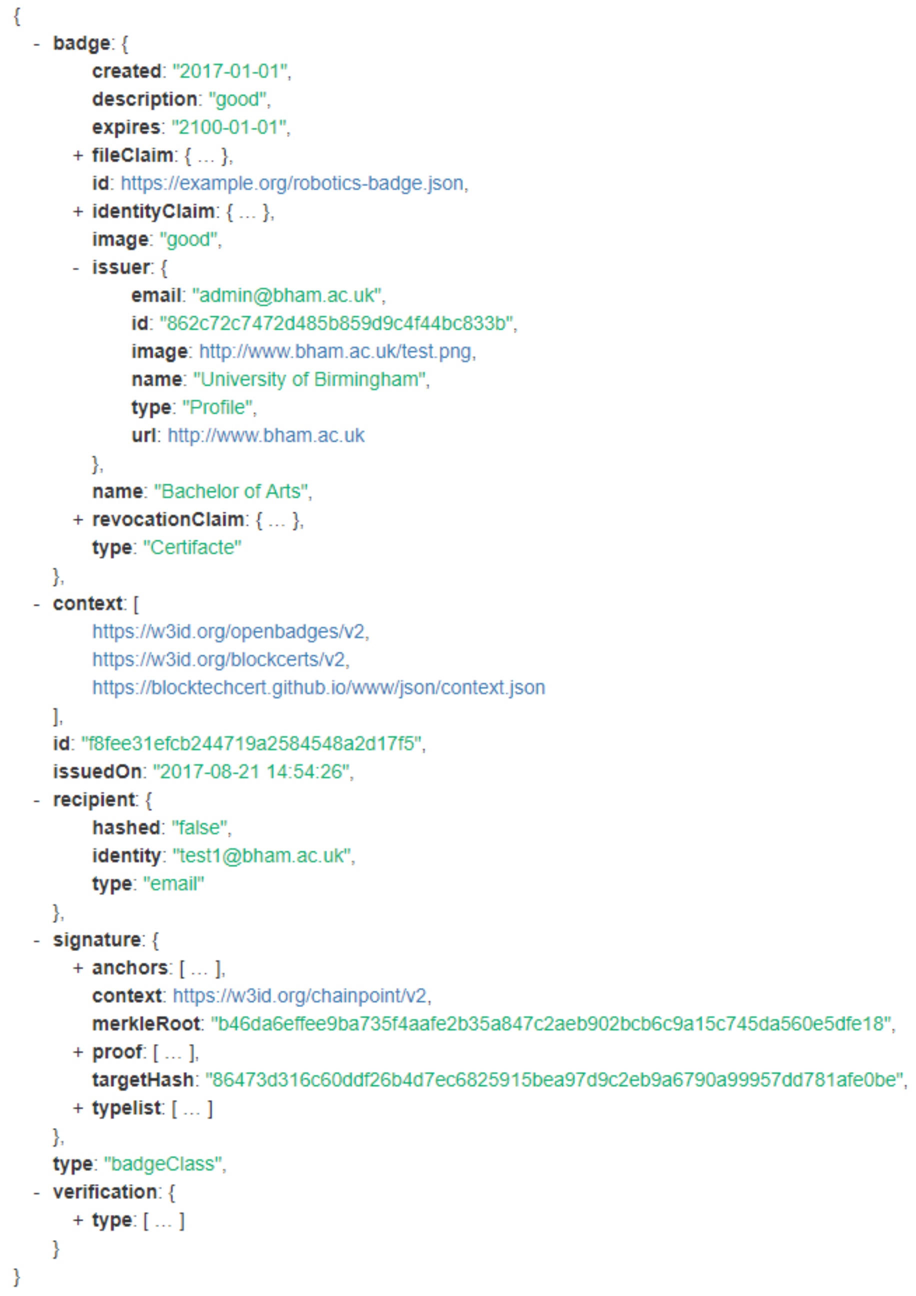
Sign in

Forgotten your password?

Sign up now

Figure 22. Intranet login page.

6.2 Certificate example



```
{
  - badge: {
        created: "2017-01-01",
        description: "good",
        expires: "2100-01-01",
      + fileClaim: { … },
        id: https://example.org/robotics-badge.json,
      + identityClaim: { … },
        image: "good",
      - issuer: {
            email: "admin@bham.ac.uk",
            id: "862c72c7472d485b859d9c4f44bc833b",
            image: http://www.bham.ac.uk/test.png,
            name: "University of Birmingham",
            type: "Profile",
            url: http://www.bham.ac.uk
        },
        name: "Bachelor of Arts",
      + revocationClaim: { … },
        type: "Certifacte"
    },
  - context: [
        https://w3id.org/openbadges/v2,
        https://w3id.org/blockcerts/v2,
        https://blocktechcert.github.io/www/json/context.json
    ],
    id: "f8fee31efcb244719a2584548a2d17f5",
    issuedOn: "2017-08-21 14:54:26",
  - recipient: {
        hashed: "false",
        identity: "test1@bham.ac.uk",
        type: "email"
    },
  - signature: {
      + anchors: [ … ],
        context: https://w3id.org/chainpoint/v2,
        merkleRoot: "b46da6effee9ba735f4aafe2b35a847c2aeb902bcb6c9a15c745da560e5dfe18",
      + proof: [ … ],
        targetHash: "86473d316c60ddf26b4d7ec6825915bea97d9c2eb9a6790a99957dd781afe0be",
      + typelist: [ … ]
    },
    type: "badgeClass",
  - verification: {
      + type: [ … ]
    }
}
```

Figure 23. An certificate example

6.3 Project instructions

Project resource:

Project home page: http://www.btcert.org/
Intranet home page: https://btcert.com/verify
Public verifying page: https://btcert.com
Intranet login page: https://btcert.com/login

Project code repository:

Main repository: https://git.cs.bham.ac.uk/BTCert/BTCert.git
Auxiliary repository: https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2016/rxl635.git

Project prerequisites:

Operating system: Centos 7 x86_64
JDK: 1.8
Docker: Docker CE 17.6
Maven: Maven 3.3

Project deploy instruction:

1. Install the running environment.
2. Import the code from repository
3. Switch to directory
4. Start the project without Docker
   ```
   $ > Cd /BTCert
   $ > mvn package && java -jar target/Boot-0.0.1-SNAPSHOT.jar
   $ > nohup java -jar target/Boot-0.0.1-SNAPSHOT.jar &
   ```
5. Start the project with Docker
   ```
   $ > mvn package docker:build
   $ > docker images
   $ > docker run image_name:tag_name
   ```

1 "Risk Advisory Group." Risk Advisory Group - Wikispooks. Accessed September 03, 2017. https://wikispooks.com/wiki/Risk_Advisory_Group.

2 "Increasing number of jobseeker CVs contain inaccuracies, finds The Risk Advisory Group." Risk Advisory. Accessed September 03, 2017. http://risk.sozowebdesign.co.uk/news/increasing-number-of-jobseeker-cvs-contain-inaccuracies-finds-the-risk-advisory-group.php.

3 "A quick study." The Economist. July 07, 2012. Accessed September 03, 2017. http://www.economist.com/node/21558318.

4 "Higher Education Degree Datacheck." HEDD Higher Education Degree Datacheck. Accessed September 03, 2017. https://hedd.ac.uk/.

5 "Blockchain Certificates About." Blockcerts. Accessed September 03, 2017. http://www.blockcerts.org/about.html.

6 Lab, MIT Media. "What we learned from designing an academic certificates system on the blockchain." Medium. June 02, 2016. Accessed September 03, 2017. https://medium.com/mit-media-lab/what-we-learned-from-designing-an-academic-certificates-system-on-the-blockchain-34ba5874f196#.kqvwzwvom.

7 Learning, ML. "Blockcerts-An Open Infrastructure for Academic Credentials on the Blockchain." Medium. October 24, 2016. Accessed September 03, 2017. https://medium.com/mit-media-lab/blockcerts-an-open-infrastructure-for-academic-credentials-on-the-blockchain-899a6b880b2f.

8 "MIT Launches Blockcerts Certification Using Bitcoin." Bitcoin News. October 28, 2016. Accessed September 03, 2017. https://news.bitcoin.com/mit-blockcerts-certification-bitcoin/.

9 Bonneau, Joseph, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. 2015. "Sok: Research Perspectives And Challenges For Bitcoin And Cryptocurrencies". Security And Privacy (SP), 2015 IEEE Symposium On.

10 Szydlo, Michael. 2004. "Merkle Tree Traversal In Log Space And Time". Advances In Cryptology - EUROCRYPT 2004, 541-554. doi:10.1007/978-3-540-24676-3_32.

11 S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Oct. 2008

12 Ober, Micha, Stefan Katzenbeisser, and Kay Hamacher. "Structure and Anonymity of the Bitcoin Transaction Graph." Future Internet 5, no. 2 (2013): 237-50. doi:10.3390/fi5020237.

13 Germanus, Daniel, Stefanie Roos, Thorsten Strufe, and Neeraj Suri. "Mitigating Eclipse attacks in Peer-To-Peer networks." 2014 IEEE Conference on Communications and Network Security, 2014. doi:10.1109/cns.2014.6997509.

14 Biryukov, Alex, and Ivan Pustogarov. "Bitcoin over Tor isnt a Good Idea." 2015 IEEE Symposium on Security and Privacy, 2015. doi:10.1109/sp.2015.15.

15 "Blockchain". 2017. En.Wikipedia.Org. accessed April 16, 2017, https://en.wikipedia.org/wiki/Blockchain.

16 "Bitcoins In Space". 2017. accessed April 2, 2017, Virgin. https://www.virgin.com/richard-branson/bitcoins-in-space.

17 "Transaction." Transaction - Bitcoin Wiki. Accessed September 03, 2017. https://en.bitcoin.it/wiki/Transaction.

18 "Script - Bitcoin Wiki". 2017. En.Bitcoin.It. accessed September 03, 2017, https://en.bitcoin.it/wiki/Script.

19 Princeton University,. 2015. Securing Bitcoin Wallets Via A New DSA/ECDSA Threshold Signature Scheme.

20 R. Merkle, "A Digital Signature Based on a Conventional Encryption Function," Proceedings of Crypto '87, pp. 369–378.

21 Alex Biryukov, Dmitry Khovratovich and Ivan Pustogarov. 2014. "Deanonymisation Of Clients In Bitcoin P2P Network". 2014 ACM SIGSAC Conference On Computer And Communications Security, 15-29.

23 "Pay to Script Hash Execution." Transactions - Pay to Script Hash Execution - Bitcoin Stack Exchange. Accessed September 04, 2017. https://bitcoin.stackexchange.com/questions/42521/pay-to-script-hash-execution.

24 "How bitcoin mining works." The Economist. January 20, 2015. Accessed September 09, 2017. https://www.economist.com/blogs/economist-explains/2015/01/economist-explains-11.

25 Gentry, Craig. "Certificate-Based Encryption and the Certificate Revocation Problem." Lecture Notes in Computer Science Advances in Cryptology — EUROCRYPT 2003, 2003, 272-93. doi:10.1007/3-540-39200-9_17.

26 "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." IETF Tools. Accessed September 05, 2017. https://tools.ietf.org/html/rfc5280#section-5.3.1.

27 "How Cybercrime Exploits Digital Certificates." InfoSec Resources. July 25, 2014. Accessed September 03, 2017. http://resources.infosecinstitute.com/cybercrime-exploits-digital-certificates/#gref.

28 Kern, A. "Advanced features for enterprise-wide role-based access control." 18th Annual Computer Security Applications Conference, 2002. Proceedings.doi:10.1109/csac.2002.1176305.

29 "Sharded Cluster High Availability." Sharded Cluster High Availability — MongoDB Manual 3.0. Accessed September 03, 2017. https://docs.mongodb.com/v3.0/core/sharded-cluster-high-availability/.

30 "Why is it dangerous to let an internal server talk to the internet (to a specific IP)?" Network - Why is it dangerous to let an internal server talk to the internet (to a specific IP)? - Information Security Stack Exchange. Accessed September 03, 2017. https://security.stackexchange.com/questions/116963/why-is-it-dangerous-to-let-an-internal-server-talk-to-the-internet-to-a-specifi.

31 "How to Safely Publish Internal Services to the Outside World." K. Scott Morrison's Blog. February 10, 2010. Accessed September 03, 2017. https://kscottmorrison.com/2010/02/09/how-to-safely-publish-internal-services-to-the-outside-world/.

32 Kienzler, Romeo. "Hyperledger – eine offene Blockchain Technologie." Blockchain Technology, 2016. doi:10.1515/9783110488951-005.

33 Dannen, Chris. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. Berkeley, CA: Apress, 2017.