TRIBHUVAN UNIVERSITY INSTITUTE OF ENGINEERING PULCHOWK CAMPUS

A MAJOR PROJECT REPORT ON

DIGITAL CERTIFICATION USING BLOCKCHAIN

SUBMITTED BY:

AMIT KUMAR PATEL (071-BCT-502)
MAHESH SHARMA (071-BCT-519)
MANISH JAYSWAL (071-BCT-520)
SUJEET KC (071-BCT-545)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL

November, 2018

COPYRIGHT

The author has agreed that the Library, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or other use of this report for financial gain without approval of the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering and author's written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering
Lalitpur, Nepal

ACKNOWLEDGEMENT

First of all, we would like to express our sincere gratitude towards Department of Electronics and Computer Engineering, IOE Pulchowk Campus for including final year major project as part of our syllabus for final year B.E. in Computer. We would like to extend our gratitude towards Dr. Diwakar Raj Pant, Head of Department, Electronics and Computer Engineering, for assisting us in our project.

We would also like to thank Mrs. Bibha Sthapit, our supervisor and Deputy Head of Department, Electronics and Computer Engineering, for supervising and providing us guidance in selecting our major project.

We would like to take the privilege to express our gratitude towards Mr. Gajendra Jung Katuwal and Mr. Sandeep Pandey for assisting and guiding us in our project.

Last but not the least we would like to thank our friends and classmates for their help and valuable suggestions.

We would also like to acknowledge all the authors of the research papers that helped us understand the required concepts and algorithms better.

Every attempt has been made to include each and every aspects of the project in this report so that the reader can clearly understand about our project. We would be pleased to get the feedback on this project.

Sincerely,
Amit Kumar Patel (071/BCT/502)
Mahesh Sharma (071/BCT/519)
Manish Jayswal (071/BCT/520)
Sujeet KC (071/BCT/545)

ABSTRACT

We are going through a very exciting phase of technological transformation. Blockchain community is redesigning a lot of existing use cases and trying to solve a lot of unsolved use cases with the help of decentralized, peer-to-peer technology. Blockchain is an undeniably ingenious invention – the brainchild of a person or group of people known by the pseudonym, Satoshi Nakamoto. By allowing digital information to be distributed but not copied, blockchain technology created the backbone of a new type of internet. Originally devised for the digital currency, Bitcoin, the tech community is now finding other potential uses for the technology. Certificate generation and verification is one of the uses cases which is being attempted to be gracefully handled by blockchain. What would an academic or a professional degree look like if it was designed today? These are the questions that are being answered by the blockchain technology. Blockchain provides a decentralized credentialing system. The blockchain acts as the provider of trust, and credentials are tamper-resistant and verifiable. Blockchain can be used in the context of not only academic but also professional, and workforce credentialing. Blockchain is an open standard. Only an open standard allows individuals to remain fully in control of their own academic history. That is important, because academic certificates are markers of our lives. They can be tickets to a better job or education, and we can use them to tell a story about who we are and how we have become that person.

Keywords: *Blockchain, digital certificate*

TABLE OF CONTENTS

ACKNOWLEDGEMENT.....iv
ABSTRACT.....v
LIST OF FIGURES.....ix
LIST OF ABBREVIATIONS.....xi
1. INTRODUCTION.....1
1.1. Background.....1
1.2. Problem Statement.....2
1.3. OBJECTIVES.....3
1.4. Scope of the Work.....3
2. LITERATURE REVIEW.....5
3. BLOCKCHAIN.....8
3.1. Types of Blockchains.....8
3.1.1. Public Blockchains.....9
3.1.2. Private Blockchains.....9
3.1.3. Federated Blockchains or Consortium Blockchains.....10
3.2. Classification Schemes.....10
3.3. P2P Blockchain Applications.....11
3.3.1. IPFS (ipfs.io).....11
3.3.2. Filecoin (filecoin.io).....11
3.3.3. Sia (sia.tech).....12
3.3.4. Etheria (Etheria.world).....12
3.4. Smart Contracts.....12
3.5. BaaS.....14
3.6. Ethereum.....14
3.7. What can Ethereum be used for?.....15
3.8. What are the benefits of Ethereum decentralized Platform?.....16
3.9. What’s the downside of decentralized applications?.....16

4. CRYPTOGRAPHY.....18
4.1. Introduction.....18
4.1.1. Cryptographic Terms.....19
4.2. Asymmetric Key Cryptography.....20
4.3. Digital Signature.....21
4.4. ECDSA.....23
4.5. Keccak-256 Hash Algorithm.....25
5. METHODOLOGY.....27
5.1. Digital Certification.....27
5.1.1. Issuer.....30
5.1.2. Certificate Generator.....33
5.1.3. Certificate Verifier:.....33
5.1.4. Web App:.....37
5.2. SDLC.....37
5.3. UML Diagrams.....40
5.3.1. Use Case Diagram.....40
5.3.2. Sequence Diagram.....41
5.4. Tools and Technologies.....42
5.4.1. Python.....42
5.4.2. Flask Web Framework.....42
5.4.3. JSON.....42
5.4.4. Bootstrap framework.....43
5.4.5. Jquery Autocomplete.....43
5.4.6. Solidity.....43
5.4.7. Web3py.....44
5.4.8. Ethereum Node.....44
5.4.9. IPFS.....44
6. PROJECT APPLICATIONS.....46
7. CONCLUSION.....47

8. LIMITATIONS AND FUTURE ENHANCEMENTS.....48
REFERENCES.....50
APPENDIX A. GANTT CHART.....51
APPENDIX B. OUTPUT SCREENSHOTS.....52

LIST OF FIGURES

Figure 1: Cryptography.....19
Figure 2: Asymmetric key cryptography.....20
Figure 3: Digital Signature working mechanism.....22
Figure 4: Elliptic curve used in ECDSA.....23
Figure 5: Signature computation.....23
Figure 6: Signature Verification.....24
Figure 7: Digital Certification Architecture.....27
Figure 8: Merkle Tree of certificate hashes.....30
Figure 9: Typical Blockchain Certificate.....31
Figure 10: Incremental Software Development Life Cycle.....38
Figure 11: Use case diagram.....40
Figure 12: Sequence Diagram.....41
Figure 13: Gantt Chart.....51
Figure 14: Home Page.....52
Figure 15: Issuer Profile Page.....52
Figure 16: Certificate Request Page.....53
Figure 17: Public/Private key-pair generation.....54
Figure 18: Making request to issuer.....55
Figure 19: Request submitted.....56
Figure 20: Issuer profile update.....57
Figure 21: Deploying issuer info to ipfs.....57
Figure 22: Requests arrived to issuer.....58
Figure 23: Issuer id added to smart contract.....58
Figure 24: Certificates' zip downloaded after issue.....59
Figure 25: Verification page for issued certificate.....60
Figure 26: Issuer not verified error.....61
Figure 27: Certificate Tampered Error.....62
Figure 28: Verified status for certificate.....63

LIST OF TABLES

Table 1: Comparison between Public and Private Blockchain.....10

LIST OF ABBREVIATIONS

BaaS	Blockchain as a Service
DLT	Distributed Ledger Technology
ECDSA	(Elliptic Curve Digital Signature Algorithm)
IOE	Institute of Engineering
IPFS	Inter-Planetary File System
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
PoW	Proof of Work
SDLC	Software Development Life Cycle
TU	Tribhuvan University
UML	Unified Modeling Language

1. INTRODUCTION

1.1. Background

Certificates are signals of achievement or membership and some are more important than others. Certificates are something that are issued by a central authority whether it is a college or an organization or a government body. University degrees (a particular type of certificate) can help you get the job you want, or prevent you from getting it if you don’t have the right certificate. Ideally we should be in charge of our own credentials. But most of the time we have to rely on third parties, such as universities or employers to store, verify, and validate our credentials. Job seekers have to request official transcripts from their alma maters (and typically pay a small fee), and employers still need to call the university if they want to be sure that a transcript wasn’t faked. The current system for sharing official records is slow, complicated, expensive and broken for everyone in a myriad of ways. This generation of students who grew up entirely during the internet age have started applying for college, and many admissions officers can share stories about applicants trying to text photos of their academic records. This being seemingly humorous, conveys an honest impression about the way things should work. It should be that easy for people to share certified records directly with others and have them trusted as authentic.

Lying about education credentials is also a common problem in the present context, and today it is very easy to counterfeit academic diplomas and certificates. In addition, the process to check someone's degree is quite laborious, as you need to get the school's contact information, reach the right person and do it again for every candidate. Medium and big companies often delegate this task to third parties as it's very time and money consuming. This is where Blockchain comes into play. Blockchain provides a solution to all of these problems. Using the blockchain and strong cryptography, it is now possible to create a certification infrastructure that puts us in control of the full record of our achievements and accomplishments. It will allow us to share a digital degree with an employer while giving the employer complete trust that the degree was in fact issued to the person presenting it.

Blockchain technology, despite of being relatively new to the tech world, has taken up the tech world by storm. It is being called the next big thing after the invention of internet. A

blockchain is essentially just a set of —blocks‖ which are linked together forming a chain. It is a digitized, decentralized, public ledger for transactions happening between any two parties. Any transaction stored in the blockchain is public i.e. anyone can see those transactions and make a copy. But despite of being a public ledger, the blockchain is tamper-proof because of strong cryptography involved in each block. Because of this meddle-proof nature of blockchain, it provides a way for issuing digital certificate without having to worry about any kind of hack and temperament of the certificates.

Hence, we have developed a system that is much more efficient, secure, and simple than what you can find today in the industry. This system that we have developed makes sure that certificates will always remain valid and verifiable by employers and also keeps them safe and impossible to copy or hack. It puts oneself in charge of their own credentials and helps them verify their credentials easily to the employers when necessary.

1.2. Problem Statement

In the present context, almost all the institutions issue their certificates in printed format. Therefore, anyone who can exactly emulate the original certificate by using graphical tools like photoshop can trick the employers or institutes. There is no means by which one can distinguish the original and duplicate certificates, except checking each certificates with physical eyes manually. This causes problem, since the ones who are honest and hardworking are ranked in the same category as the ones who have never put any effort in their academic performance.

There have been several cases in Nepal where fake certificates are being used and only very few of them have been caught, that too, is through the informants. Thus, one can sense that a concrete tool to distinguish the real certificates from the fake ones is lacking.

1.3. OBJECTIVES

The project's main goal is to store certificates digitally using blockchain to achieve decentralization, ensuring immutability.

Hence, the functional objectives can be pointed out as follows:

- To develop a system that uses blockchain technology to issue digital certificates to students in a university.
- To use cryptography to make the digital certificates impossible to copy or hack.
- To prevent frauds from making false university degrees.
- To make the certificate issuing process more efficient and less time and money consuming.

1.4. Scope of the Work

Digital certification can be used in various universities in order to issue, store and verify the certificates using a secure and efficient mechanism. When certification systems are not working well, the consequences can be more than just inefficient, such as the cumbersome and expensive process of requesting a university transcript: they can be disastrous, such as when a refugee is unable to provide a certificate of completed study, and is therefore prevented from continuing his/her education. Digital systems could help in both of these situations.

Moreover, the certificates maintained on the blockchain cannot be tampered with or deleted or hacked easily. The digital storage and verification of the certificates can keep up the importance of the certificates even after the university is no longer in existence or even when the recipient is away from the university.

One of the most important aspects of this system is to make fraud certificates impossible to be verifiable. The certificates shall be verified based upon the signature on the blockchain and hence the fraudulent certificates will be easily detected.

Hence, this system is likely to be adopted by many universities in the future and also be a basis for certificate verification in a secure, easy and faster way as it uses a newer and tamper-resistant blockchain technology.

2. LITERATURE REVIEW

When we lived in small tight-knit communities, people knew whom they could turn to when they needed an expert. However, as we started moving around more and our lives and networks grew, we needed to come up with portable ways to signal our expertise to new acquaintances. This is where certificates came into play. Certificates provided us with a way to verify our expertise in our field. In today's context, certificates are required in more ways than just to verify context. They are extremely vital to establish the authenticity of an applicant, whether for visa-applications, employment or education. Nowadays many business processes in government institutions and organizations alike require original documents to be provided for verification purposes before any further processing can occur. Since some documents (e.g. birth certificates) may exist only once, the parties involved may experience reluctance in handling these important documents as they could get damaged, lost in the mail, etc.

The friction is a result of

- Requiring original documents to be presented to comply with the institution’s vetting and verification processes

- The reluctance by the parties involved of having the responsibility to protect these important physical pieces of paper throughout the process.

One way to solve this problem is by digitizing the assets. But even with digital assets questions like how can we create a digital proof that a digital asset (soft copy of birth certificate, your University diploma, business contract or legal document) has been certified (signed) by an authorized organization or government institution and secondly, how can anybody around the world verify the authenticity of a particular digital asset without having to rely on a 3rd party or intermediary arise. Moreover the problems of fraud and tampering are also not eliminated with this approach. Many sectors, mainly the educational sector, have suffered from such problems. A survey by one of the largest online job finder sites, CareerBuilder, shows that a staggering 58% of employers have

caught a lie on a resume. The site has more than 23 million unique visitors and over 1.6 million jobs. So the need of valid, verifiable and impossible to fake digital certificates has been realized and such kind of technology has been made possible with the rise of Blockchain technology.

Blockchain is essentially a continuously growing list of records, called *blocks*, which are linked and secured using cryptography. The blockchain is best known for its connection to the cryptocurrency bitcoin. But in essence it is a just a distributed ledger to record transactions. What makes it special is that it is durable, time-stamped, transparent, and decentralized. Those characteristics are equally useful for managing financial transactions and for other decentralized applications. Using such a secure and durable system, it is now possible to create a certification infrastructure that puts us, users, in control of the full record of our achievements and accomplishments. This solution does not suggest to store the digital asset on the Blockchain (this would be a very inefficient and expensive approach). Instead it only stores the proof that a digital asset has been certified (or signed) by an institution on the Blockchain. If anybody would want to verify the legitimacy of a digital asset they can simply verify the digital asset by vetting it using the proof provided. Hence, the Blockchain's role in this solution is to provide an immutable storage container for these proofs. At a high level this solution includes the following steps:

- Creation of a digital asset and storage of the digital proof (signature) to the Blockchain
- Transmission of the digital asset (Email, file sharing, etc.)
- Validation of the digital asset using the signature stored on the Blockchain and vetting of the institution that issued the asset.

The data stored in blockchain essentially stays there forever without getting tampered in any way by hackers or malicious software. This opens a new path for us to share a digital degree with an employer while giving the employer complete trust that the degree was in fact issued to the person presenting it. This is exciting because it is not only a better way to deal with the way certificates work today, but it is also an opportunity to think about what certificates may look like in the future. For students, the benefits go beyond mere novelty.

They can share their diplomas almost immediately with whomever they please, free of charge, without involving an intermediary. This is particularly important for students who need to prove to an employer or another university that they have a University diploma. And thanks to the blockchain, the third party can easily verify that the diploma is legitimate without having to contact the Registrar's Office. Using a portal, employers or schools can paste a link or upload a student's digital diploma file and receive a verification immediately. The portal essentially uses the blockchain as a notary, locating the transaction ID (which identifies when the digital record was added to the blockchain), verifying the keys, and confirming that nothing has been altered since the record was added.

Obviously, Blockchain being a comparatively new technology, we had to do extensive research to know about it and it's functionality. During our research we found out that there has already work being done on digital certificate using blockchain which was good news for us since it would help us understand our project idea quickly and in more depth. From the limited documentation that had been written on this topic we found out that digital certificate implementation using blockchain was a modular concept. We could implement it the way we liked. There was no hard and fast rule that the digital certificate should be implemented in one specific way. We could show our own creativity in our own implementation rendering it different and unique from the existing implementations. And we intend to this by implementing our blockchain in Ethereum since every work that has been done in the field of digital certification till date has been done in Bitcoin blockchain which being the first and pioneer blockchain implementation is comparatively primitive, slow and expensive when compared to Ethereum.

3. BLOCKCHAIN

Blockchain, the technology behind Bitcoin, Ethereum and other cryptocurrencies, seems to be the driving technology behind the next generation of Internet, also referred to as the Decentralized Web, or the Web3. Blockchain is a novel solution to the age-old human problem of trust. It provides an architecture for so-called trustless trust. It allows us to trust the outputs of the system without trusting any actor within it. A Blockchain protocol operates on top of the Internet, on a P2P Network of computers that all run the protocol and hold an identical copy of the ledger of transactions, enabling P2P value transactions without a middleman though machine consensus.

Blockchain is a shared, trusted, public ledger of transactions, that everyone can inspect but which no single user controls. It is a distributed database that maintains a continuously growing list of transaction data records, cryptographically secured from tampering and revision. The ledger is built using a linked list, or chain of blocks, where each block contains a certain number of transactions that were validated by the network in a given timespan.The crypto-economic rulesets of the blockchain protocol (consensus layer) regulate the behavioral rulesets and incentive mechanism of all stakeholders in the network.

3.1. Types of Blockchains

The idea emerged that the Bitcoin blockchain could be in fact used for any kind of value transaction or any kind of agreement such as P2P insurance, P2P energy trading, P2P ride sharing, etc. Colored Coins and Mastercoin tried to solve that problem based on the Bitcoin Blockchain Protocol. The Ethereum project decided to create their own blockchain, with very different properties than Bitcoin, decoupling the smart contract layer from the core blockchain protocol, offering a radical new way to create online markets and programmable transactions known as Smart Contracts.

Private institutions like banks realized that they could use the core idea of blockchain as a DLT, and create a permissioned blockchain (private or federated), where the validator is a member of a consortium or separate legal entities of the same organization. The term blockchain in the context of permissioned private ledger is highly controversial and disputed.

Private blockchains are valuable for solving efficiency, security and fraud problems within traditional financial institutions, but only incrementally. It's not very likely that private blockchains will revolutionize the financial system. Public blockchains, however, hold the potential to replace most functions of traditional financial institutions with software, fundamentally reshaping the way the financial system works.

3.1.1. Public Blockchains

State of the art public Blockchain protocols based on PoW consensus algorithms are open source and not permissioned. Anyone can participate, without permission. (1) Anyone can download the code and start running a public node on their local device, validating transactions in the network, thus participating in the consensus process – the process for determining what blocks get added to the chain and what the current state is. (2) Anyone in the world can send transactions through the network and expect to see them included in the blockchain if they are valid. (3) Anyone can read transaction on the public block explorer. Transactions are transparent, but anonymous/pseudonumous.
Examples: Bitcoin, Ethereum, Monero, Dash, Litecoin, Dodgecoin, etc.

3.1.2. Private Blockchains

Write permissions are kept centralized to one organization. Read permissions may be public or restricted to an arbitrary extent. Example applications include database management, auditing, etc. which are internal to a single company, and so public readability may in many cases not be necessary at all. In other cases public audit ability is desired. Private blockchains are a way of taking advantage of blockchain technology by

setting up groups and participants who can verify transactions internally. This puts you at the risk of security breaches just like in a centralized system, as opposed to public blockchain secured by game theoretic incentive mechanisms. However, private blockchains have their use case, especially when it comes to scalability and state compliance of data privacy rules and other regulatory issues. They have certain security advantages, and other security disadvantages.

Examples: MONAX, Multichain

3.1.3. Federated Blockchains or Consortium Blockchains

Federated Blockchains operate under the leadership of a group. As opposed to public Blockchains, they don't allow any person with access to the Internet to participate in the process of verifying transactions. Federated Blockchains are faster (higher scalability) and provide more transaction privacy. Consortium blockchains are mostly used in the banking sector. The consensus process is controlled by a pre-selected set of nodes; for example, one might imagine a consortium of 15 financial institutions, each of which operates a node and of which 10 must sign every block in order for the block to be valid. The right to read the blockchain may be public, or restricted to the participants.

Example: R3 (Banks), EWF (Energy), B3i (Insurance), Corda

3.2. Classification Schemes

Many people have tried to classifying blockchains, but there is no consensus on how to accurately distinguish between different types of Blockchains. Here is a listing of selection of different classification schemes.

Table 1: Comparison between Public and Private Blockchain

	Public	Private/Federated
Access	Open read/write	Permissioned read and/or write
Speed	Slower	Faster

Security	Proof of Work Proof of Stack Other consensus mechanisms	Pre-approved participants
Identity	Anonymous Pseudonymous	Known
Asset	Native Asset	Any Asset

3.3. P2P Blockchain Applications

A totally peer-to-peer (P2P) internet seems like a far-off reality. The web as we know it today relies on a number of centralized servers and platforms — and the blockchain protocol makes it possible to bypass these intermediaries when sending digital currency, or any form of data. In fact, this is one of the innovations that helped make Bitcoin the most popular cryptocurrency to date. Digital currency is far from the only P2P blockchain application, however. Because it was designed with disintermediation in mind, P2Pfriendly developers are using blockchain technology to build software that aims for web decentralization.

3.3.1. IPFS (ipfs.io)

IPFS, or the Inter-Planetary File System, is a system for decentralized web applications that uses blockchain protocol to archive and sustain digital files. Billing itself as the ―permanent web,‖ one goal of IPFS is to solve the problem of web fragility by harnessing the same technology that makes the blockchain ledger difficult to manipulate or erase. This is the reason behind the name: the inventors of IPFS have aimed for it to be so sturdy that it could even withstand transmission into outer space. Here on Earth, IPFS is being touted an alternative to HTTP (hypertext transmission protocol), a current core technology of the web. This may become more useful as the volume of data on the web continues to exponentially increase and issues of data management that are difficult to solve with HTTP keep growing.

3.3.2. Filecoin (filecoin.io)

Filecoin is a project from the same team that developed IPFS. Filecoin is both a platform for storing data and its own eponymous cryptocurrency. Filecoin is earned by hosting files: the program's mining software allows users to exchange their unused storage space for the currency, which can then be exchanged for US dollars, Ether, Bitcoin, and other currencies. This differs from IPFS in that it's incentive-based, although it shares many of the same benefits, including stability and resistance to censorship.

3.3.3. Sia (sia.tech)

Sia is another protocol that uses the power of blockchain to decentralize data storage and transmission. Like Filecoin, Sia features its own cryptocurrency — Siacoin — which can be mined with its software. The Sia website makes the project's goals very clear: —the long-term goal of Sia is to be the backbone storage layer of the internet,‖ it says. Right now, its competition is the current internet structure (i.e. dropbox, google drive), but in the not-too-distant future, the potential of Sia may only be threatened by similar programs.

3.3.4. Etheria (Etheria.world)

For something a bit more lighthearted, there's Etheria. Etheria is a Minecraft-like video game based on the Ethereum blockchain: the —world's first decentralized virtual world,‖ as it says. In Etheria, players own tiles and —farm‖ them for building blocks, which is a process similar to cryptocurrency mining. As with Minecraft, the idea behind Etheria is just to build new things in the virtual world, although future versions may include more gaming features.

Because it is based on Ethereum, Etheria is strongly resistant to censorship and takedown by any state, company, or individual, including its developer. It requires a bit more tech savvy than Minecraft to set up, but as a P2P blockchain-based video game, it's breaking new ground. Along with other games, like Beyond The Void and Spells of Genesis, it may become part of the first generation of blockchain gaming.

3.4. Smart Contracts

Blockchain was initially designed for P2P money only. But it soon showed the potential to be used for any kind of P2P value transaction on top of the Internet. The Ethereum project thus introduced the idea of decoupling the contract layer from the blockchain layer, where the ledger itself is used by smart contracts that trigger transactions automatically when certain pre-defined conditions are met. By decoupling the smart contract layer from the blockchain layer, blockchains like Ethereum aim to provide a more flexible development environment than the Bitcoin blockchain.

These smart contracts are a piece of code running on top of a blockchain network, where digital assets are controlled by that piece of code implementing arbitrary rules. They have properties of contractual agreements but should not be confused with legal contracts. If and when all parties to the smart contract fulfill the pre-defined arbitrary rules, the smart contract will auto execute the transaction. These smart contracts aim to provide transaction security superior to traditional contract law and reduce transaction costs of coordination and enforcement. Smart contracts can be used for simple economic transactions like sending money from A to B. They can also be used for registering any kind of ownership and property rights like land registries and intellectual property, or managing smart access control for the sharing economy, just to name a few. Furthermore, smart contracts can be used for more complex transactions like governing a group of people that share the same interests and goals.

With blockchains and smart contracts, we can now imagine a world in which contracts are embedded in digital code and stored in transparent, shared databases, where they are protected from deletion, tampering, and revision. In this world, every agreement, every process, task, and payment would have a digital record and signature that could be identified, validated, stored, and shared. Intermediaries like lawyers, brokers, and bankers, and public administrators might no longer be necessary. Individuals, organizations, machines, and algorithms would freely transact and interact with one another with little friction and a fraction of current transaction costs.

Therefore, blockchains & smart contracts:

- Radically reduce transaction costs (bureaucracy) through machine consensus and auto-enforceable code.
- Bypass the traditional principal-agent dilemmas of organizations, thus providing an operating system for what some refer to as —trustless trust‖. This means that you don’t have to trust people and organizations, you trust code, which is open source and provides transparent processes.

3.5. BaaS

Setting up an environment to test and research blockchain requires an ecosystem with multiple systems to be able to develop research and test. The big players in the cloud industry like Amazon(AWS), Microsoft(Azure), IBM(BlueMix) have seen the potential benefits of offering blockchain services in the cloud and started providing some level of BaaS to their customers. Users will benefit from not having to face the problem of configuring and setting up a working blockchain. Hardware investments won’t be needed as well. Microsoft has partnered with ConsenSys to offer Ethereum Blockchain as a Service (EBaaS) on Microsoft Azure. IBM(BueMix) has partnered with Hyperledger to offer BaaS to its customers. Amazon announced they would be offering the service in collaboration with the Digital Currency Group. Developers will have a single-click cloudbased blockchain developer environment, that will allow for rapid development of smart contracts.

Examples: Accenture, ConsenSys, Cognizant, Deloitte, IBM, PricewaterhouseCoopers (PWC), Ernst & Young.

Blockchain technology allows students to cryptographically store their academic records and control who can view them. Blockchain credentials are a type of open badge that provides additional security, longevity, interoperability, and recipient ownership for highstakes claims. A badge is a digital record of achievement, typically verified by a vendor. These badges can be anchored on the blockchain, however, when a greater level of security and recipient independence is required. In addition to helping students to own their education, records based on the blockchain will assist employers seeking to authenticate a prospective employee’s scholastic record. Rather than contacting multiple colleges or universities, an employer can rely on a student’s verified history on the blockchain.

3.6. Ethereum

Although commonly associated with Bitcoin, blockchain technology has many other applications that go way beyond digital currencies. In fact, Bitcoin is only one of several hundred applications that use blockchain technology today.

Until relatively recently, building blockchain applications has required a complex background in coding, cryptography, mathematics as well as significant resources. But times have changed. Previously unimagined applications, from electronic voting & digitally recorded property assets to regulatory compliance & trading are now actively being developed and deployed faster than ever before. By providing developers with the tools to build decentralized applications, Ethereum is making all of this possible.
At its simplest, Ethereum is an open software platform based on blockchain technology that enables developers to build and deploy decentralized applications.

Like Bitcoin, Ethereum is a distributed public blockchain network. Although there are some significant technical differences between the two, the most important distinction to note is that Bitcoin and Ethereum differ substantially in purpose and capability. Bitcoin offers one particular application of blockchain technology, a peer to peer electronic cash system that enables online Bitcoin payments. While the Bitcoin blockchain is used to track ownership of digital currency (bitcoins), the Ethereum blockchain focuses on running the programming code of any decentralized application.

In the Ethereum blockchain, instead of mining for bitcoin, miners work to earn Ether, a type of crypto token that fuels the network. Beyond a tradeable cryptocurrency, Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.

Smart contract is just a phrase used to describe computer code that can facilitate the exchange of money, content, property, shares, or anything of value. When running on the blockchain a smart contract becomes like a self-operating computer program that

automatically executes when specific conditions are met. Because smart contracts run on the blockchain, they run exactly as programmed without any possibility of censorship, downtime, fraud or third party interference.

3.7. What can Ethereum be used for?

Ethereum enables developers to build and deploy decentralized applications. A decentralized application or Dapp serve some particular purpose to its users. Bitcoin, for example, is a Dapp that provides its users with a peer to peer electronic cash system that enables online Bitcoin payments. Because decentralized applications are made up of code that runs on a blockchain network, they are not controlled by any individual or central entity.

Any services that are centralized can be decentralized using Ethereum. Think about all the intermediary services that exist across hundreds of different industries. From obvious services like loans provided by banks to intermediary services rarely thought about by most people like title registries, voting systems, regulatory compliance and much more. Ethereum can also be used to build Decentralized Autonomous Organizations (DAO). A DAO is fully autonomous, decentralized organization with no single leader. DAO's are run by programming code, on a collection of smart contracts written on the Ethereum blockchain. The code is designed to replace the rules and structure of a traditional organization, eliminating the need for people and centralized control. A DAO is owned by everyone who purchases tokens, but instead of each token equating to equity shares & ownership, tokens act as contributions that give people voting rights.

3.8. What are the benefits of Ethereum decentralized Platform?

- Immutability – A third party cannot make any changes to data.
- Corruption & tamper proof – Apps are based on a network formed around the principle of consensus, making censorship impossible.
- Secure – With no central point of failure and secured using cryptography, applications are well protected against hacking attacks and fraudulent activities.

- Zero downtime – Apps never go down and can never be switched off.

3.9. What's the downside of decentralized applications?

Despite bringing a number of benefits, decentralized applications aren't faultless. Because smart contract code is written by humans, smart contracts are only as good as the people who write them. Code bugs or oversights can lead to unintended adverse actions being taken. If a mistake in the code gets exploited, there is no efficient way in which an attack or exploitation can be stopped other than obtaining a network consensus and rewriting the underlying code. This goes against the essence of the blockchain which is meant to be immutable. Also, any action taken by a central party raises serious questions about the decentralized nature of an application.

4. CRYPTOGRAPHY

4.1. Introduction

Cryptography is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. It is the practice and study of techniques for secure communication in the presence of third parties. Moreover, cryptography is about constructing and analyzing protocols that prevent the public from reading private messages. Various aspects in information security such as the following are central to cryptography.

- Data confidentiality: Confidentiality refers to protecting information from being accessed by unauthorized parties.
- Data integrity: Integrity refers to assuring that the transmitted has not been tampered with.
- Authentication: Authentication has two elements. The first element is authenticating that you're getting data from the correct entity and the second element is validating the integrity of that data.
- Non-repudiation: Non-repudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.

Cryptography can be thought of as the intersection of the disciplines of mathematics, computer science, electrical engineering, communication science, and physics. Cryptography is applicable in various fields like electronic commerce, chip-based payment cards, digital currencies, computer passwords, and military communications.

Certificates stored in blockchain need to be secure. Hence, cryptography has an important role to play in achieving this task.

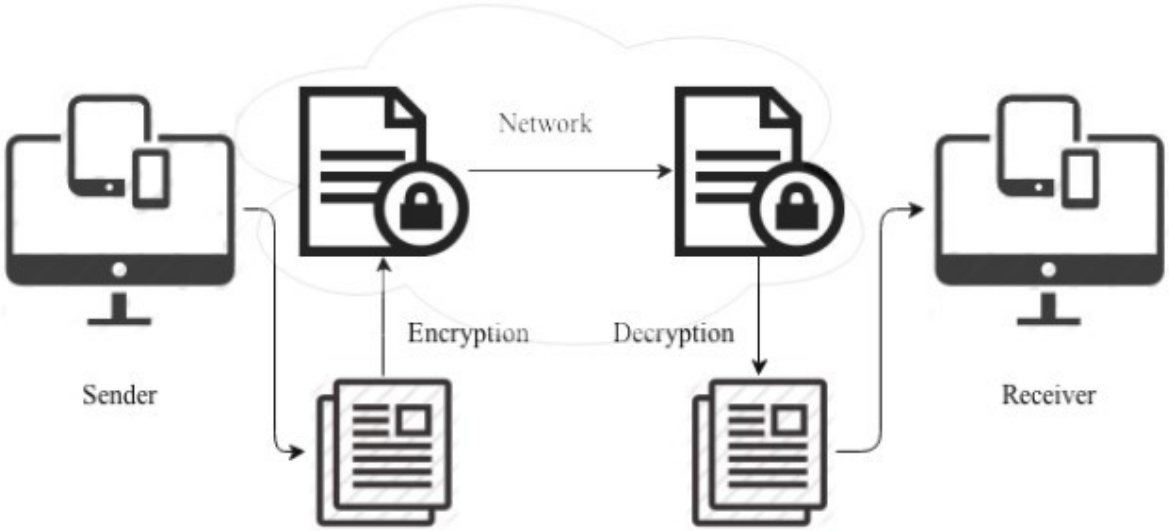

Figure 1: Cryptography

4.1.1. Cryptographic Terms

- Plaintext**:** Data that can be read and understood without any special measures.

- Encryption**:** Encryption is the process of applying a mathematical function to a file that renders its contents unreadable and inaccessible, unless one has the decryption key.

- Cipher text**:** Encrypting plaintext results in unreadable gibberish.

- Decryption**:** The process of reverting cipher text to its original plaintext.

- Key**:** Some secret information that converts between plaintext and cipher text. Keys are of two types:
    - Public Key: The public key can be known to the entire public
    - Private Key: The private key is only accessible to its owner. It should be kept private, hence the name private key.

- Hash**:** A hash is a value returned by a cryptographic hash function. A cryptographic hash function is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size.

Modern cryptography can be classified into two types:

- **Symmetric cryptography**, where there is one key, shared between sender which is used for both encryption and decryption.

- **Asymmetric cryptography**, where there are two keys; one is used for encryption at sender and another is used for decryption at receiver's end.

4.2. Asymmetric Key Cryptography

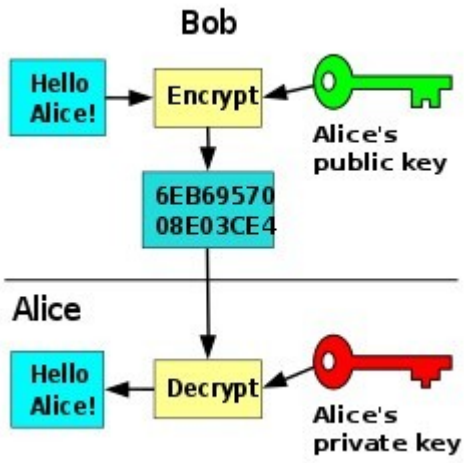

Figure 2: Asymmetric key cryptography

Also known as public-key cryptography, it is an encryption scheme that uses two mathematically related, but not identical, keys - a public key and a private key. Unlike symmetric key algorithms that rely on one key to both encrypt and decrypt, each key performs a unique function. The public key is used to encrypt and the private key is used to decrypt.

It is computationally infeasible to compute the private key based on the public key. Because of this, public keys can be freely shared, allowing users an easy and convenient method for encrypting content and verifying digital signatures, and private keys can be kept secret, ensuring only the owners of the private keys can decrypt content and create digital signatures.

Since public keys need to be shared but are too big to be easily remembered, they are stored on digital certificates for secure transport and sharing. Since private keys are not shared, they are simply stored in the software or operating system one uses, or on hardware (e.g., USB) containing drivers that allow it to be used with their software or operating system.

Public Key Cryptography is an essential part of Ethereum's protocol and is used in several places to ensure the integrity of messages created in the protocol. Wallet creation and signing of transactions, which are the core components of any currency rely heavily on public key cryptography. Ethereum's protocol uses what's called the Elliptic Curve Digital Signature Algorithm (ECDSA) to create a new set of private key and corresponding public key. The public key is then used with a hash function to create the public address that Ethereum users use to send and receive funds. The private key is kept secret and is used to sign a digital transaction to make sure the origin of the transaction is legitimate.

4.3. Digital Signature

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message, and that the message was not altered in transit.
Digital signatures employ asymmetric cryptography.

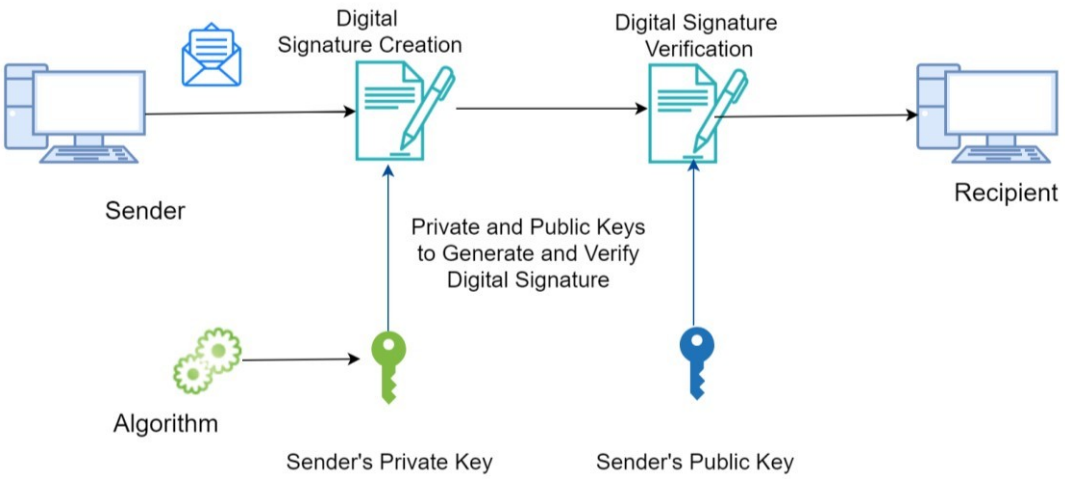

Figure 3: Digital Signature working mechanism

Working of digital signature is explained below:

- Bob signs m by encrypting with his private key K_B, creating ―signed‖ message, $K_B(m)$.
- Suppose Alice receives msg m, digital signature $K_B(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B to $K_B(m)$ then checks $K_B(K_B(m)) = m$.
- If $K_B(K_B(m)) = m$, whoever signed m must have used Bob's private key.
- Alice thus verifies that:
  - ✓ Bob signed m.
  - ✓ No one else signed m.
  - ✓ Bob signed m and not m'.

In Ethereum, once the transaction is signed by the owner, the transaction is sent to the memory pool where it sits to be processed by miners. The miners use the sender's public key to ensure that the digital signature is authentic so that a hacker cannot spend a user's funds without their consent. If the ownership and digital signature check out, they include the transaction in the next block, and the money is sent from one wallet to another.

4.4. ECDSA

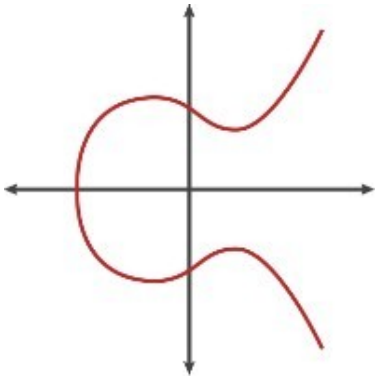

Figure 4: Elliptic curve used in ECDSA

Elliptic Curve Cryptography (ECC) and ECDSA are a specific flavor of asymmetric cryptography. They are widely used in blockchain technology because of three reasons:

- Their computational performance is economical compared to a lot of other algorithms
- The keys that are generated are relatively short
- Bitcoin started it, so most new blockchain projects have copied it ECDSA uses the algebraic structure of elliptic curves over finite fields.

Signature Computation:

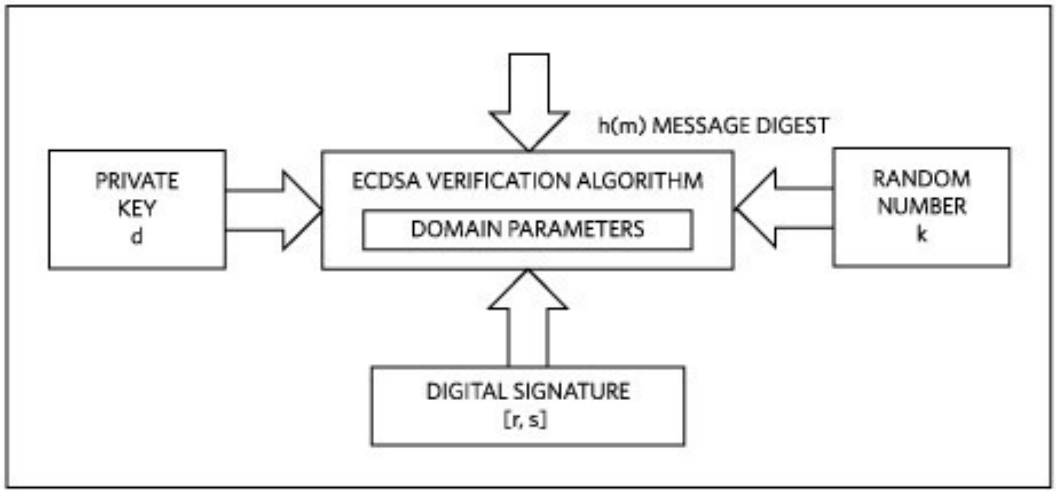

Figure 5: Signature computation

The signature consists of two integer numbers, r and s. The following equation shows the computation of r from the random number k and the base point G(x,y) on the elliptic curve, with a prime number p and a prime order n:

$(x1, y1) = k \times G(x, y) \mod p \quad r = x1 \bmod n$

To be valid, r must be different from zero. In the rare case when r is 0, a new random number, k, must be generated and r needs to be computed again.

After r is successfully computed, s is computed according to the following equation using scalar operations. Inputs are the message digest h(m); the private key d; r; and the random number k:

$s = (k^{-1} (h(m) + d * r) \bmod n$

To be valid, s must be different from zero. If s is 0, a new random number k must be generated and both r and s need to be computed again.

Signature Verification:

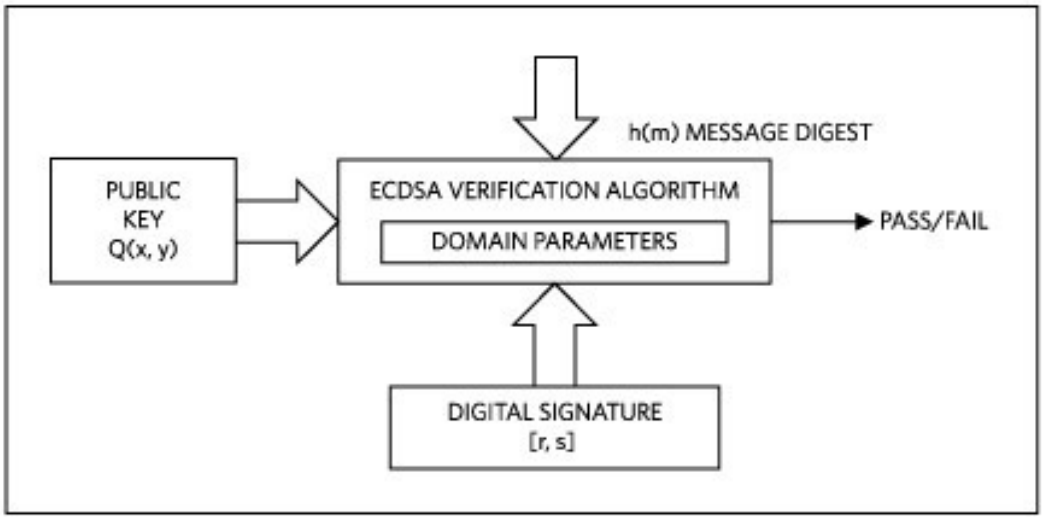

Figure 6: Signature Verification

The following equation shows the individual steps of the verification process. Inputs are the message digest h(m), the public key Q(x,y), the signature components r and s, and the base point G(x,y):

$w = s^{-1} \bmod n$ $u1 =$
$(h(m) * w) \bmod n$ $u2 =$
$(r * w) \bmod n$
$(x2, y2) = (u1 \times G(x, y) + u2 \times Q(x, y)) \bmod n$

The verification is successful, if x2 is equal to r, thus confirming that the signature was indeed computed using the private key.

4.5. Keccak-256 Hash Algorithm

Hashing is the process of taking an input of any length and turning it into a cryptographic fixed output through a mathematical algorithm. Ethereum uses KECCAK-256 algorithm to generate hash of data blocks and merkle tree. It is an alias to sha3, meaning that keccak256 produces identical results to sha3. Keccak uses an innovative "sponge engine" to hash the message text. It is fast, with a reported average speed of 12.5 cycles per byte on an Intel Core 2 processor.

Keccak can resist known attacks with a minimum complexity of 2^n, where n is the hash size. It has a wide safety margin. To date, third-party cryptanalysis has shown no serious weaknesses in Keccak.

In Keccak, the underlying function is a permutation chosen in a set of seven Keccak-f permutations, denoted Keccak-f[b], where $b \in \{25,50,100,200,400,800,1600\}$ is the width of the permutation. The width of the permutation is also the width of the state in the sponge construction.

The state is organized as an array of 5×5 lanes, each of length w ∈ {1,2,4,8,16,32,64} and b=25w. When implemented on a 64-bit processor, a lane of Keccak-f[1600] can be represented as a 64-bit CPU word.

We obtain the Keccak[r,c] sponge function, with parameters capacity c and bitrate r, if we apply the sponge construction to Keccak-f[r+c] and by applying a specific padding to the message input.

Given an input bit string N, a padding function pad, a permutation function f that operates on bit blocks of width b, a rate r and an output length d, we have capacity $c = b - r$ and the sponge construction Z = sponge[f, pad, r] (N, d), yielding a bit string Z of length d, works as follows:

- Pad the input N using the pad function, yielding a padded bit string P with a length divisible by r (such that n = len(P)/r is integer), □ Break P into n consecutive r-bit pieces P_0, ..., P_{n-1} □ Initialize the state S to a string of b 0 bits.
- Absorb the input into the state: For each block P_i,
- Extend P_i at the end by a string of c 0 bits, yielding one of length b,
- XOR that with S and
- Apply the block permutation f to the result, yielding a new state S
- Initialize Z to be the empty string
- While the length of Z is less than d:
- Append the first r bits of S to Z
- If Z is still less than d bits long, apply f to S, yielding a new state S. □ Truncate Z to d bits

For Keccak, the state S consists of a 5 × 5 array of w = 64-bit words, b = 5 × 5 × w = 5 × 5 × 64 = 1600 bits total.

5. METHODOLOGY

5.1. Digital Certification

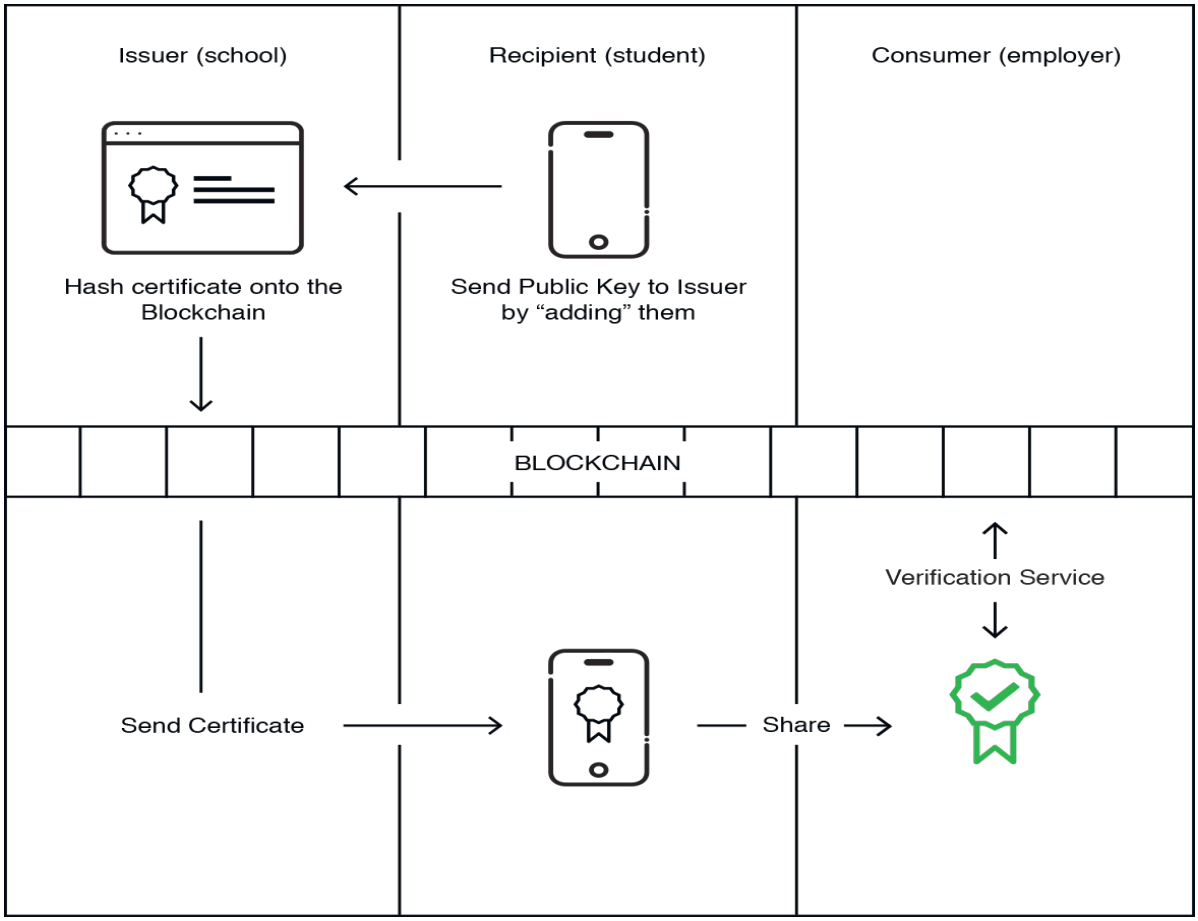

Figure 7: Digital Certification Architecture

The above diagram describes the overall digital certification process and architecture. The recipient requests a certificate to an issuer sending their public key, email id and name. This forms a strong association to identify a recipient. The issuer prepares a JSON certificate for each recipient and issues a batch of certificates to ethereum blockchain. The issuer then sends the certificates to the recipients. The recipient can verify their certificates via blockchain. The recipient may also forward their certificates to third party(may be an employer) who can also easily verify the validity of the certificate.

An issuer may register to our web app and add their information including the list of their public keys with the date of creation, date of revocation and expiry date. The issuer adds the revocation list url and other issuer specific information to their profile. They have to

deploy these information to ipfs through our web app. The ipfs hash of the issuer information file issued on ipfs is incorporated to their profile. The issuer, in order to become a legitimate issuer has to make a digital signature of the ipfs hash as the message and send the digital signature, ipfs hash(message) and the claimed public key and the name of issuer(university or institution) to the admin email address. Digital Signature may be generated using myetherwallet website.

The admin verifies the whether the claimed public key belongs to the issuer by looking at an assumed public ledger containing the address mapping of all the institutions. If the issuer name has the claimed public key mapping in the ledger, the public key is used to verify the message(ipfs hash). The admin also checks the profile of the issuer for validation. Whenever an issuer updates their profile it is mandatory they issue the issuer information to ipfs and get their ipfs hash added to smart contract by the admin after verification.

Now the issuer whose ipfs hash is added to the smart contract is a legitimate issuer unless they update their profile. The certificates issued by a legitimate issuer is a valid certificate. While issuing a certificate the issuer creates a JSON certificate for each recipient. The hash of each certificate is added to a Merkle Tree and the Merkle root is added to data field of an ethereum transaction. The transaction is signed by the issuer's private key thus creating a Elliptic Curve Digital Signature. The transaction is broadcasted to ethereum network using remote node. If the transaction succeeds then the proof fields are appended to the certificate thus making the certificate verifiable. The fields like targetHash which is the local certificate hash, Merkle root, Merkle proof which includes the hashes required to compute the root node for a given certificate at a particular index on the Merkle Tree, sourceId which is the transaction Id of the blockchain transaction required to fetch the transaction information in the future, the type of network(i.e ethereum ropsten). The certificates with proof field or receipt form a signed blockchain certificate which gets downloaded as a ziip file containing certificates for a batch. The certificates are then distributed to the respective recipients.

The recipient may verify the certificate by just uploading the JSON file on the website and pressing the verify button. The web app renders the JSON fields in an attractive and

userfriendly way with issuer logo, university or institution name, their signature and description. After the verify button is pressed, the blockchain transaction id in the certificate proof field is used to fetch the transaction data. The ipfs hash in the certificate is checked whether it is contained in smart contract and if the ipfs hash is in smart contract the issuer is considered to be legitimate and the issuer information file is fetched from ips using the hash.

The verification is done in four steps:

a. Integrity check: At this stage, the proof field is checked for a predefined format of proof filed. Then, The hash of the certificate is obtained and checked with the tagetHash in the certificate. The targetHash and the Merkle proof fields are used to generate a Merkle root which is compared to the Merkle root obtained from blockchain transaction. If all three checks are valid then the integrity is ensured.
b. Authenticity check: At this stage, the transaction signing key and the time stamp are obtained from blockchain transaction and the list of issuer keys and their validity period from issuer information file. If the signing key is contained in issuer information and the transaction time stamp is after the creation time of the key and before expiration time then the Authenticity is ensured. This ensures that the transaction was done via the same public key valid life span which is claimed by the issuer in the issuer information file which comes from a legitimate issuer validated by the admin.
c. Revocation check: This stage, checks whether the certificate is enlisted in a list of revoked certificates maintained on the issuer's url. If the certificate is in revoked list then the certificate fails the verification.
d. Expiration check: This is a simple check that ensures the certificate is still valid in the current time checking the certificate's expire date field.

If all the four step validation passes then the certificate is a valid digital blockchain certificate. The application consists of modules that function independently and are finally integrated as one complete application. There are mainly four modules in this system.

5.1.1. Issuer

The issuer module is responsible for issuing a batch of certificates and thus generate signed blockchain certificates from an equal number of unsigned certificates. This module accepts a batch of unsigned certificates. These certificates are hashed using SHA-256 encryption algorithm and the hashed certificates are added to the leaf of Merkle Tree and thus Merkle root is obtained.

Suppose the batch contains n certificates, and certificate i contains recipient i's information.

The issuer hashes each certificate and combines them into a Merkle tree:

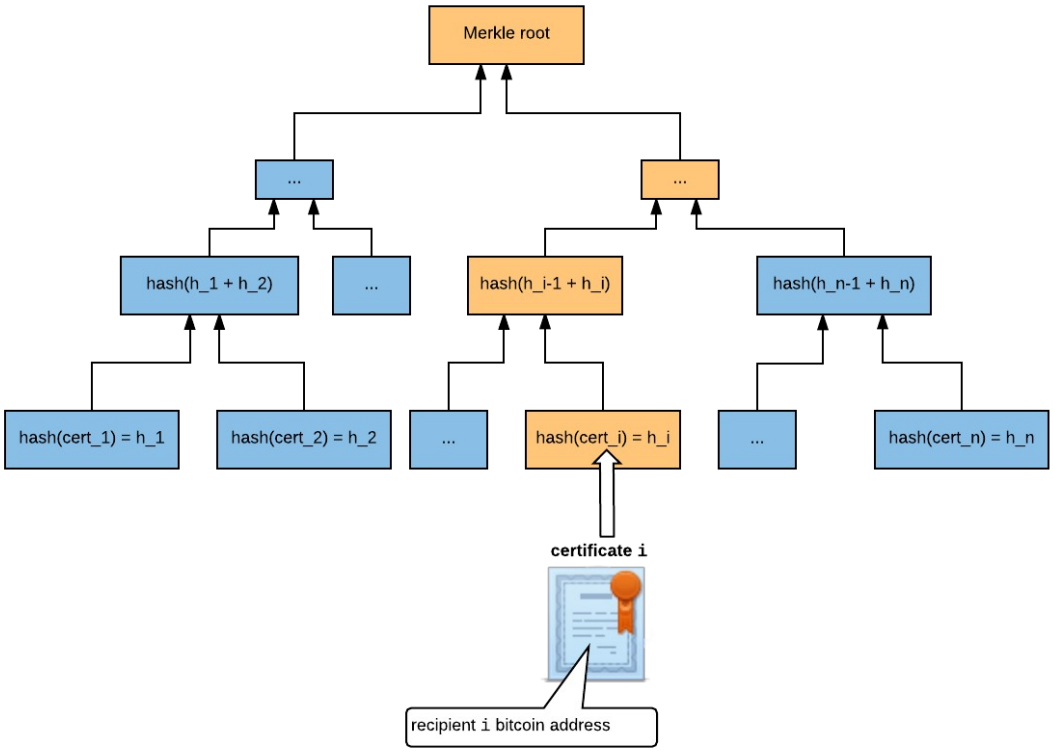

Figure 8: Merkle Tree of certificate hashes

The root of the Merkle tree, which is a 256-bit hash, is added to the data field of ethereum transaction.

SHA-256 :

The SHA (Secure Hash Algorithm) is one of a number of cryptographic hash functions. A cryptographic hash is like a signature for a text or a data file. SHA-256 algorithm generates an almost-unique, fixed size 256-bit (32-byte) hash. Hash is a one way function – it cannot be decrypted back. This makes it suitable for password validation, challenge hash authentication, anti-tamper, digital signatures.

SHA-256 is one of the successor hash functions to SHA-1, and is one of the strongest hash functions available.

We have used this hashing algorithm in order to hash the certificates that can be easily included in the Merkle tree and after the Merkle root is transacted we shall issue the certificates to the recipient email and other social sites provided. The certificate contains the JSON certificate created by the issuer and a receipt containing transaction and signature information.

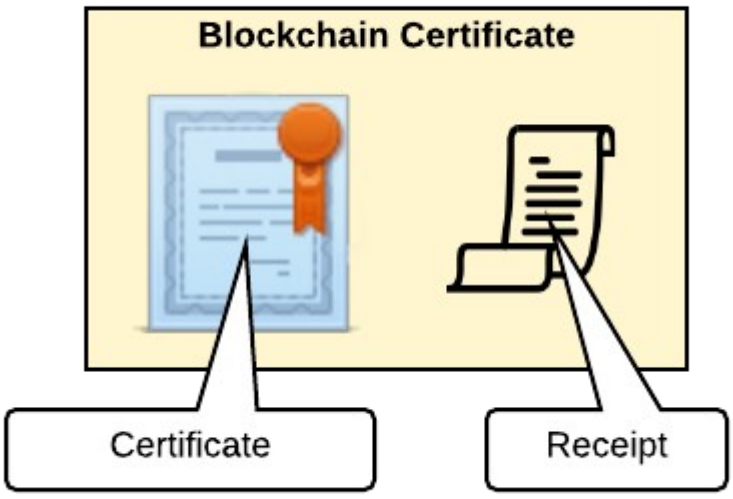

Figure 9: Typical Blockchain Certificate

This receipt contains:

- The Ethereum transaction ID storing the Merkle root
- The expected Merkle root on the blockchain
- The expected hash for recipient i's certificate
- The Merkle path from recipient i's certificate to the Merkle root, i.e. the path highlighted in orange above. h_i -> … -> Merkle root

The above information in the receipt which is a signature of the blockchain certificate is be used to verify that the particular certificate is contained in the Merkle tree and is a valid certificate.

Signature Algorithm:

Signature Algorithm takes an array of tbs, a privateKey, and options as inputs and produces a signatureValue as output.

1. Take an array of tbs, which is an array of the data to be signed, as input and generate a Merkle Tree.
2. Include the root of the Merkle Tree ("Merkle Root") in the data field of the Blockchain transaction.
3. Sign the transaction with encoding of privateKey appropriate for the blockchain.
4. For each tbs, create the signature element, which includes:
    - The hash of the element as targetHash
    - The merkle root as merkleRoot
    - The Merkle Proof for tbs, which is the path from targetHash to merkleRoot in the Merkle Tree, as proof. There should not include the endpoints of the path (targetHash and merkleRoot)
    - The blockchain anchor(s) as anchors. This describes how to look up the blockchain transaction. Its fields include the type of blockchain transaction as type and the transaction id as sourceId.
5. Return the result as signatureValue.

5.1.2. Certificate Generator

Certificates are open badges compliant, which is important, because there is an entire community of open badges issuers that we want to support, and because open badges is becoming an IMS standard.
Blockchain Certificate schemas extend those of Open Badges. As with Open Badges, we will provide both a JSON-LD context and JSON schema. The purpose of the JSON-LD context is to map types to Internationalized Resource Identifiers (IRIs), providing semantic context for data. The JSON Schema is used for syntactic validation.

This Open Badge specification describes a method for packaging information about accomplishments, embedding it into portable image files as digital badges, and establishing resources for its validation. It includes term definitions for representations of data in Open Badges.

The generator takes the issuer configurations to generate a template JSON file populating the issuer-specific information. Then for each recipient maintained in a csv file, the template is populated with recipient-specific informations. This generates a certificate for each recipient which can then be provided to Issuer module to transact and sign and thus issue.

5.1.3. Certificate Verifier:

The verifier checks whether the certificate is valid or not. Anyone should be able to verify independently. The following are the processes that occur in verification process:

Blockchain Transaction:

A Blockchain Certificate must contain at least one anchor to a blockchain transaction. The anchors entry says that the transaction was performed on the Ethereum blockchain, and the field needed to verify integrity of the certificate is the data field depending on the type of blockchain. This also gives the transaction id.

Issuer Identity:

The issuer id field in the Blockchain Certificate says where to find the issuer's current information about which keys are valid. This is typically an ipfs hash, which (when dereferenced) contains an array of public keys claimed by the issuer.

{
"@context": [
"https://w3id.org/openbadges/v2",
"https://w3id.org/blockcerts/v2"
],
"id": "QmcTAhUEnsDuMzbk15rPoerN9cymsAkxT5VrK9viY3RwDS",
"url": "https://www.issuer.org",
"name": "University of Learning",
"email": contact@issuer.org,

"publicKey": [
{
"id": "ecdsa-koblitz-pubkey:0x37f5257621fE96835BbB49E453E3dB37428b8A55",
"created": "2016-01-03T14:34:57.907890+00:00"
},
{ "created": "2016-01-03T14:34:57+0000", "id": "ecdsa-koblitzpubkey:0xede6839340C2c2ef1D4Ea00096590b83261b9cb1", "expires": "2017-0103T14:34:57+0000", }
],
"revocationList": "https://www.digicerts.org/samples/2.0/revocation-list-testnet.json",
"type": "Profile"

}

This information is required to cross check the public keys claimed by the issuer with the information from the blockchain transaction.

Issuer revocation information:

The revocation List field in the Blockchain Certificate says where to obtain the issuer's list of revoked certificates (a.k.a. assertions).

Check certificate Integrity:

Checking the certificate integrity ensures that the certificate has not been tampered with. This consists of 3 steps:

- Validate the Merkle proof in the certificate.
- Compare the hash of the local certificate with the value in the receipt.
- Compare the merkleRoot value in the certificate with the value in the blockchain transaction.

Check certificate authenticity:

This step verifies that the certificate was authored by the issuer. This is verified by ensuring the signing key for the blockchain transaction is indeed claimed by the issuer, and the key was valid at the time the transaction was issued.

This uses the timestamp and input address from the blockchain transaction details, and the issuer identification provided in "Issuer Identity".

This also ensures that the issuer is indeed the person with the public key as the issuer is verified by a centralized admin and the ipfs hash of issuer identity is added on smart contract. So an issuer legitimacy and mapping to public kay is done via smart contract and then whether the transaction is done using that public key or not is done using the signing key and timestamp on blockchain hence giving a complete mapping from the institution to blockchain transaction.

From blockchain transaction information, we obtain the timestamp and input address. This will vary depending on which service is used. For Blockchain.info, we need the addr field from the inputs array and the time field.

"inputs":[
{ "prev_out":{ ...

"addr":"0x37f5257621fE96835BbB49E453E3dB37428b8A55", ... } }]..."time":1475524375,

This is a Unix epoch time format, and a tool can convert it to a human readable format. This example yields 03 Oct 2016 19:52:55 GMT.

This public key is valid:

- The issuer claims the public key 0x37f5257621fE96835BbB49E453E3dB37428b8A55 is valid starting 03 Jan 2016 14:34:57 GMT
- The transaction occurred 03 Oct 2016 19:52:55 GMT, which is after the created date, and the key had not expired or been revoked when the transaction occurred.

This rules out exceptional (and possibly fraudulent) cases, such as:

- the public key is not claimed by the issuer
- the transaction was issued after the public key was revoked or expires

A critical distinction in this example is that the transaction is considered valid even though the key expired. This is ok -- all that matters is that the transaction was performed when the key was active.

A key expiration is different from a certificate expiration; expiring keys is a good security practice for issuers.

Check not revoked by issuer:

The input obtained from "Issuer revocation information" contains the list of revoked certificates (or "assertions").If the certificate has been revoked, the (optional) revocation reason may provide more information about why the certificate was revoked.

Check certificate has not expired:

The certificate may contain an expiration date (an ISO-8601 date). If present, verification must compare this value, available in the expires field, against the current time.

5.1.4. Web App:

The web app is developed using flask web framework in python. In this web app there is a user friendly interface for anyone trying to verify a credential/certificate. There is an upload certificate field in the middle of index page. A user can upload a JSON file and hence verify it pressing verify button. It also renders the JSON file in a readable and graphical format. A recipient can search for issuer in the search box and select the issuer to make request to. Our web app provides facility to create public/private key pair for a user before making request. The request gets piled up to the requested issuer. There is issuer sign up option where an issuer can sign up , update their information, issue issuer identity file and issue certificates. On the other hand there is an admin page to validate issuers and manage smart contract.

5.2. SDLC

The adapted methodology for this project is incremental SDLC. Incremental Model helps to deliver the sequence of releases in incremental basis which speeds up the progress of development of each functionality. Each developed functionality is usable independently, one after another. First increment is always a base feature and other features are added in next increments with new releases, after the review of first release. This process is carried out till the complete product is developed.

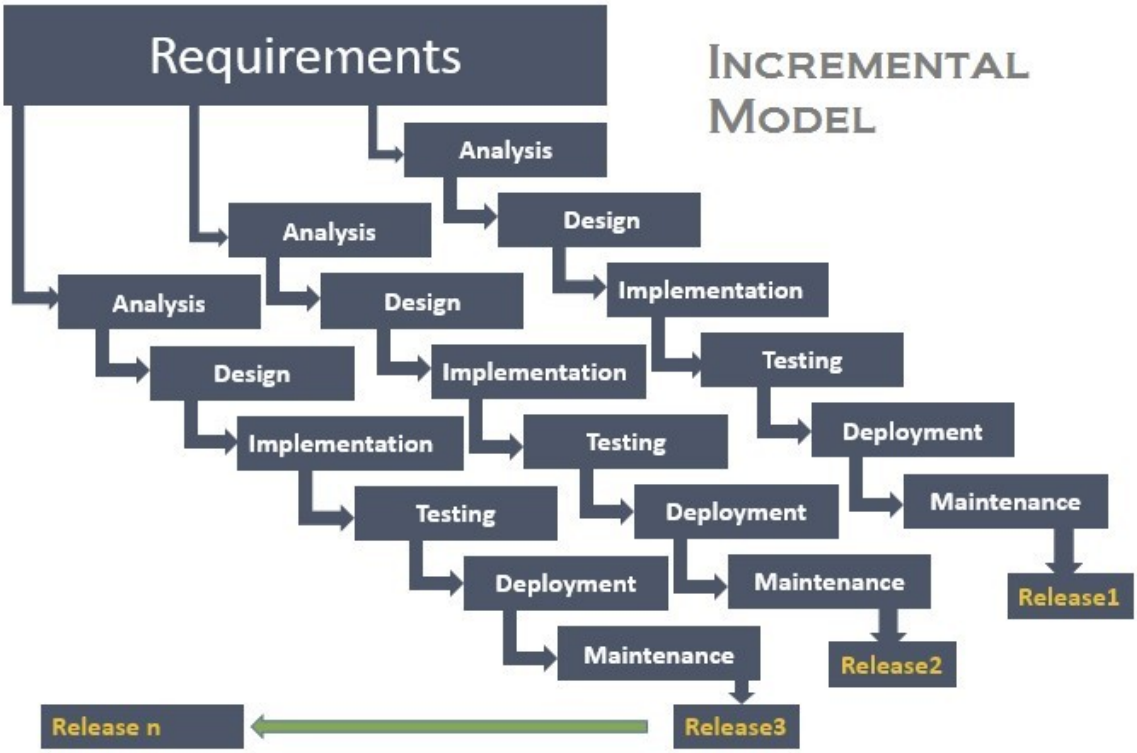

Figure 10: Incremental Software Development Life Cycle

The reasons behind using this model are as follows:

- Can develop prioritized requirements first.
- Initial product delivery is faster.
- Important functionality can be seen early.
- Lowers initial delivery cost.
- Each release is a product increment, so that there is a working product at hand all the time.
- Feedback to each product increment can be heard, thus avoiding surprises at the end of development.
- Requirements changes can be easily accommodated.

The core component of our software is cert-issuer. So the first increment involved the development cert-issuer module. This module was initially operated in console and all the logs were seen in console. We assumed arbitrary JSON certificates which was manually written and provided them as input to cert-issuer and then issued the certificate on ethereum ropsten test network. This was the first release in our project.

The second increment was the verifier module. This module verified the certificates issued in the first increment via console. This provided log of the verification status. We had to manually specify the location of the certificate to be verified such that the certificate was read and then verification steps were carried out.

The third increment was the certificate generator module. After verification ans issuing was successful we were focused to autogenerate the certificates that we had manually created. For this we used certain standards of certificates and openbadges. In this mdule, the issuer specific template was created and then populated with recipient specific information in the second cycle.

Finally, we developed the web app that made use of all of the above independent components and rendered their outputs on a webpage. The web app has issuer profile and makes use of cert-issuer, verifier, and all the features all in one package. Each individual components went through analysis, design, implementation, testing, deployment and maintenance.

5.3. UML Diagrams

5.3.1. Use Case Diagram

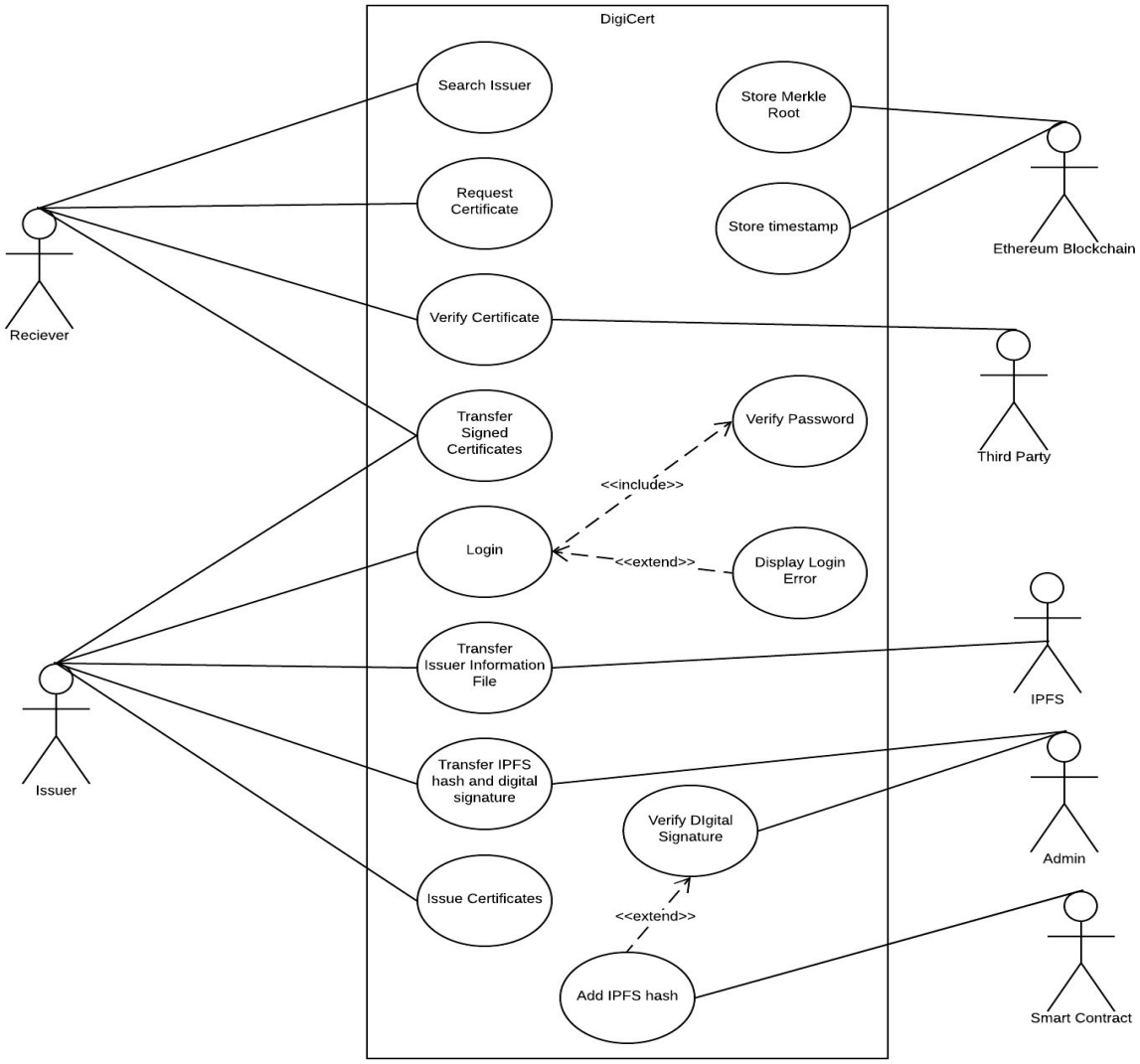

Figure 11: Use case diagram

The use case diagram of the system is shown above. As shown in the figure, there are 7 actors and their respective use cases are shown by the associations. The recipient is the primary actor as it initiates all the processes that are run by the system. The recipient can search for issuer, request certificate, receive signed certificate and finally verify the certificate. The issuer can login to the system, send information file to the ipfs, get ipfs hash, send ipfs hash and digital signature to the admin for verification and when verified the issuer can issue certificates. In the same way the admin verifies the issuer and adds the

ipfs hash to the smart contract. Any sort of third parties such as employers and potential education destinations can also verify the certificates using the system.

5.3.2. Sequence Diagram

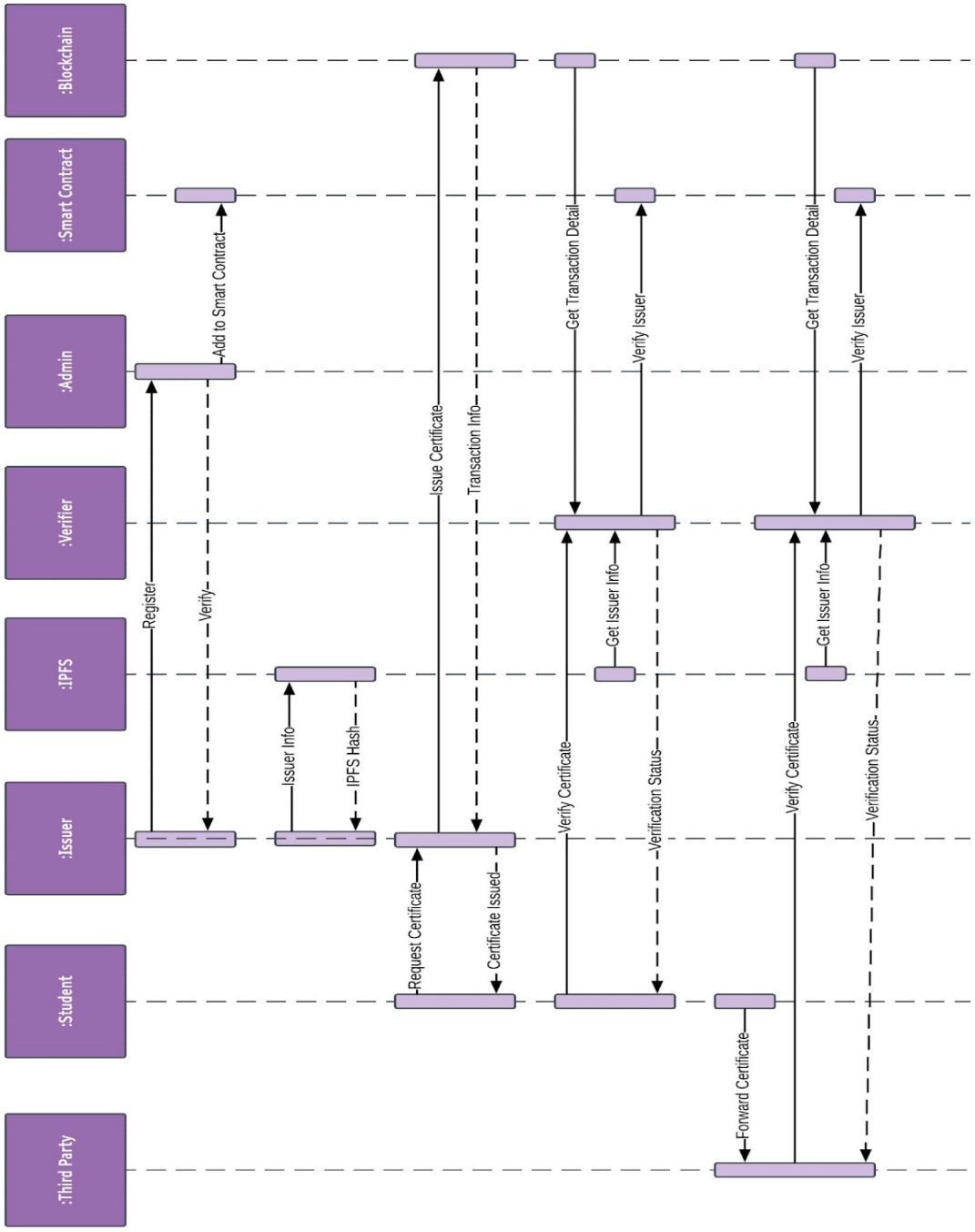

Figure 12: Sequence Diagram

The sequence diagram of the system is shown above. Firstly, the issuer registers with necessary information which the admin verifies and adds the issuer information to smart contract. The issuer's information is then stored in ipfs and ipfs hash is returned for integrity of issuer's information. The student then requests certificate from the issuer. The issuer issues him certificate and stores the information in blockchain. Now anyone wishing to verify the certificate, like third party or the student himself, can verify the certificate by getting transaction details from blockchain, issuer information from ipfs and, verifying it with smart contract.

5.4. Tools and Technologies

5.4.1. Python

Python is an interpreted high-level programming language. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python has large number of libraries that help in writing codes rapidly and efficiently, without having to spend time on low level programming. Therefore, python was chosen as the primary language to develop this project.

5.4.2. Flask Web Framework

Flask is a light-weight framework is used to develop web applications using python. It is often considered a micro-framework as it does not require particular tools or libraries. Flask is also easy to get started with as a beginner because there is little code required for getting a simple app up and running.

5.4.3. JSON

JSON (Javascript Object Notation) is a lightweight data interchange format which arranges data as label-value pairs, often with ordered list of values. It is represented in text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages. This makes JSON ideal to use as datainterchange language.

5.4.4. Bootstrap framework

Bootstrap is a free and open-source front-end framework for designing websites and web applications. It contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It helps build elegant and stylish websites with minimal effort in coding.

5.4.5. Jquery Autocomplete

Autocomplete enables users to quickly find and select from a pre-populated list of values as they type, leveraging searching and filtering. When typing in the autocomplete field, the plugin starts searching for entries that match and displays a list of values to choose from.
By entering more characters, the user can filter down the list to better matches.

This can be used to choose previously selected values, such as entering tags for articles or entering email addresses from an address book. Data can be pulled in from a local or remote source: Local is good for small data sets, e.g., an address book with 50 entries; remote is necessary for big data sets, such as a database with hundreds or millions of entries to select from. Any HTML field that can receive input can be converted into an Autocomplete, namely, <input> elements, <textarea> elements, and elements with the content editable attribute.

5.4.6. Solidity

Solidity is a contract-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms. It was developed by Gavin Wood, Christian Reitwiessner, Alex Beregszaszi, Liana Husikyan, Yoichi Hirai and several former Ethereum core contributors to enable writing smart contracts on blockchain platforms such as Ethereum.

Solidity is a statically-typed programming language designed for developing smart contracts that run on the EVM. Solidity is compiled to bytecode that is executable on the

EVM. With Solidity, developers are able to write applications that implement selfenforcing business logic embodied in smart contracts, leaving a non-repudiable and authoritative record of transactions.

5.4.7. Web3py

Web3.py is a python library inspired by the original Javascript based web3library. This library exposes a standard and familiar way to interact with the JSON-RPC interface exposed by Ethereum nodes. In addition, it exposes a number of common utilities such as Ethereum currency denomination conversions, a contract class for interacting with smart contracts, and encoding/decoding utilities.

5.4.8. Ethereum Node

Ethereum Node (maybe a light node or a full node) is a peer on the ethereum network that holds the entire(full node) blockchain on the local machine. They enforce consensus rules of ethereum. A node can either be locally deployed or a remote node hosted by several providers can be used.

5.4.9. IPFS

InterPlanetary File System (IPFS) is a protocol and network designed to create a contentaddressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system.IPFS was initially designed by Juan Benet, and is now an open-source project developed with help from the community.

IPFS is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files. In some ways, IPFS is similar to the World Wide Web, but IPFS could be seen as a single BitTorrent swarm, exchanging objects within one Gitrepository. In other words, IPFS provides a high-throughput, content-addressed block storage model, with content-addressed hyperlinks.This forms a generalized Merkle directed

acyclic graph (DAG). IPFS combines a distributed hash table, an incentivized block exchange, and a self-certifying namespace. IPFS has no single point of failure, and nodes do not need to trust each other not to tamper with data in transit. Distributed Content Delivery saves bandwidth and prevents DDoS attacks, which HTTP struggles with.

The filesystem can be accessed in a variety of ways, including via FUSE and over HTTP. A local file can be added to the IPFS filesystem, making it available to the world. Files are identified by their hashes, so it's caching-friendly. They are distributed using a BitTorrentbased protocol. Other users viewing the content aid in serving the content to others on the network.

6. PROJECT APPLICATIONS

Students can use this application, to get a verifiable, tamper-proof version of their diploma that they can share with employers, schools, family and friends. They can share their diplomas almost immediately with whomever they please, free of charge, without involving an intermediary. This is particularly important for students who need to prove to an employer or another university that they have a University degree or any other academic degree. Without Digital Certificate Issuance and storage costs upto 40$(it may vary) per students but with the help of Digital Certificate Instant Digital Verification of certificate is possible. Students can link the certificates to their linkedin profile.Individuals can safely store their certificates, and share them with others, for example, an employer. It can be used to issue any kind of digital credentials, including professional certificates, transcripts, credits, or degrees. Employers can start verifying incoming certificates.

7. CONCLUSION

In this modern age of technology where everything is being digitized, it is high time that academic and other credentials also be changed to digital forms. Even though it has been decades since internet took over the world and computers and other digital devices are the norms, the traditional way of using hard copy of credentials was the way everyone was following up until recent years. This was mainly because it's always a risk to a dive into a new ocean and also because of the drawbacks associated with digital assets like hacking and tampering. But with the support of a robust and fool proof technology like blockchain, it is finally the time for the digital certificates to take over. With blockchain the perks of digital certificates overshadow the cons and hence, they are being massively taken in to implementation in last 2 years. Actually, digital certificates are already a thing in North America and Europe and several countries in Africa and South East Asia. They have already taken initiatives towards this digital future.

Therefore, with this project we plan to introduce the Nepalese community to a better way of handling credentials and inspire them not to follow but to move along with the rest of the world in this regard.

8. LIMITATIONS AND FUTURE ENHANCEMENTS

We have tried to alleviate many of the limitations in this project and have made it a really useful application which can even be deployed in the real field after considering some deployment issues that will arise while hosting the site on a server commercially.
However, some limitations still prevail that we have a plan to look after in the future.

Some of the considerable limitations are as follows:

a. We have assumed a public ledger containing the issuer and public key mapping that is publicly available and maintained with high security and reliability. This might not seem a friendly approach currently but as the tradition of digital certificates will evolve these sorts of practices will also be a general thing.
b. We have used third party platforms like https://www.myetherwallet.com/signmsg.html and https://etherscan.io/verifySig to verify digital signature of the issuer such that the issuer legitimacy can be verified. We may incorporate them within our project such that external dependency can be removed.
c. Our application requires a good internet connection to operate properly. Some features like making transactions and fetching and dumping data to ipfs may not work properly in slow internet connection.
d. Since we use decentralized network, sometimes availability may be an issue. The issuer information hosted on decentralized database may not be available sometimes.
e. In order to achieve and enhance security and reliability we have compromised decentralization to some extent by introducing an admin who is responsible for verifying legitimacy of an issuer. But once the issuer is verified everything else (except checking of revocation information) are decentralized.
f. We have created a basic certificate for a recipient that gives certification of certain accomplishment. More detailed recipient-specific information can be added for complete information of certificates. Information regarding marks obtained credits

and all the minor details that are present in a typical certificate shall be incorporated in the project in future.

g. We shall extend our support for android, ioS and other platforms as well in the future.

h. It can be cumbersome for new users to learn how to use the system. So we plan to implement simplify the user experience by incorporating simpler user interfaces and extensive documentation so that the users can find their way within the web app.

REFERENCES

[1] Andreas M. Antonopoulos,‖Mastering Bitcoin‖,O'Reilly Media,Inc., United States of America, June 2017

[2] S. Nakamoto, —Bitcoin: A peer-to-peer electronic cash system,‖ 2008

[3] Dec 2017, [Online]. Available: https://www.eventbrite.com/e/blockchain-applicationsseminar-tickets-41331447472

[4] —Learn Blockchain Coding & much more,‖ Dec 2017, [Online]. Available: https://blockgeeks.com

[5] —Blockcerts-An Open Infrastructure for Academic Credentials on the Blockchains,‖ Dec 2017, [Online]. Available: https://medium.com/mit-media-lab/blockcerts-an-openinfrastructure-for-academic-credentials-on-the-blockchain-899a6b880b2f [6] Dec 2017, [Online]. Available: https://www.reddit.com/r/crypto/

[7] Dec 2017, [Online]. Available: https://www.reddit.com/r/BlockChain/

[8] Dec 2017, [Online]. Available: https://www.reddit.com/r/btc/

[9] Dec 2017, [Online]. Available: https://www.reddit.com/r/Bitcoin/

[10] Dec 2017, [Online]. Available: https://www.reddit.com/r/ethereum/

[11] Dec 2017, [Online]. Available: https://www.reddit.com/r/ethtrader/

[12] "Medium", Jan 2018,[Online]. Available: https://medium.com/@preethikasireddy/howdoes-ethereum-work-anyway-22d1df506369

[13] "Hackernoon", Jan 2018,[Online]. Available: https://hackernoon.com/ethereum-smartcontracts-in-python-a-comprehensive-ish-guide-771b03990988

[14] "BlockchainHub", April 2018,[Online]. Available: https://blockchainhub.net/blockchainintro/

[15] March 2018,[Online]. Available: https://www.instantssl.com/https-tutorials/digitalsignature.html

[16] —Learning Machine‖, March 2018, [Online]. Available: https://www.learningmachine.com/blockchain-credentialing/

[17] "Blockcerts The Open Standard For Blockchain crecedentials", April 2018,[Online]. Available: https://www.blockcerts.org/

[18] April 2018,[Online]. Available: https://en.wikipedia.org/wiki/Public-key_cryptography [19] April 2018,[Online]. Available: https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

APPENDIX A. GANTT CHART

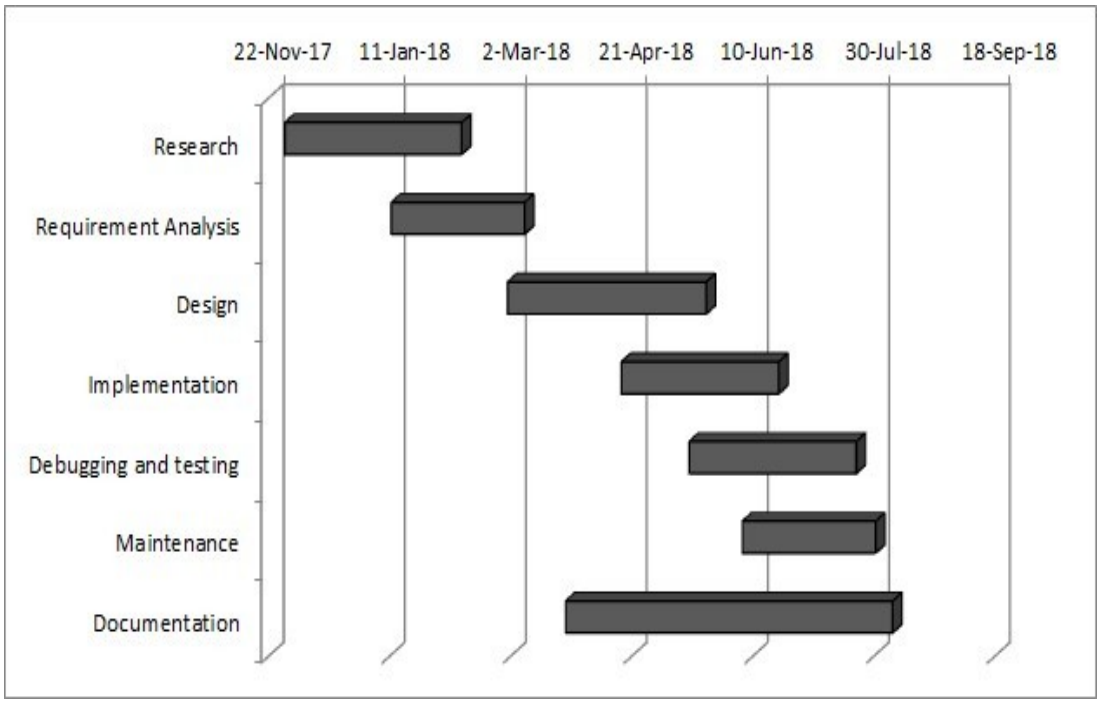

Figure 13: Gantt Chart

APPENDIX B. OUTPUT SCREENSHOTS

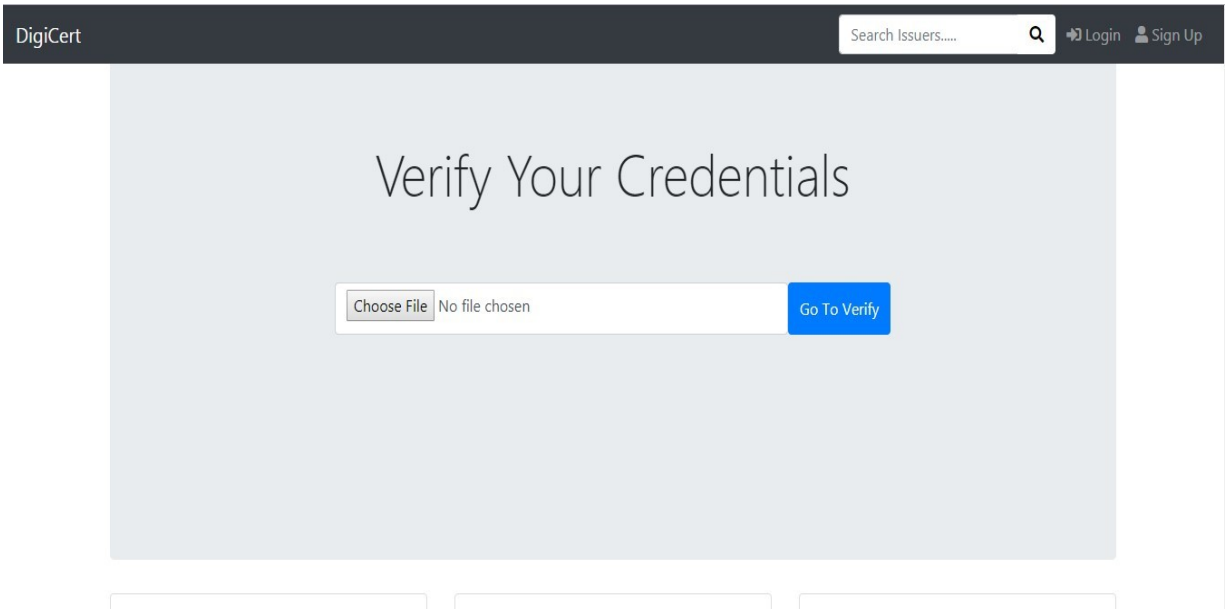

Figure 14: Home Page

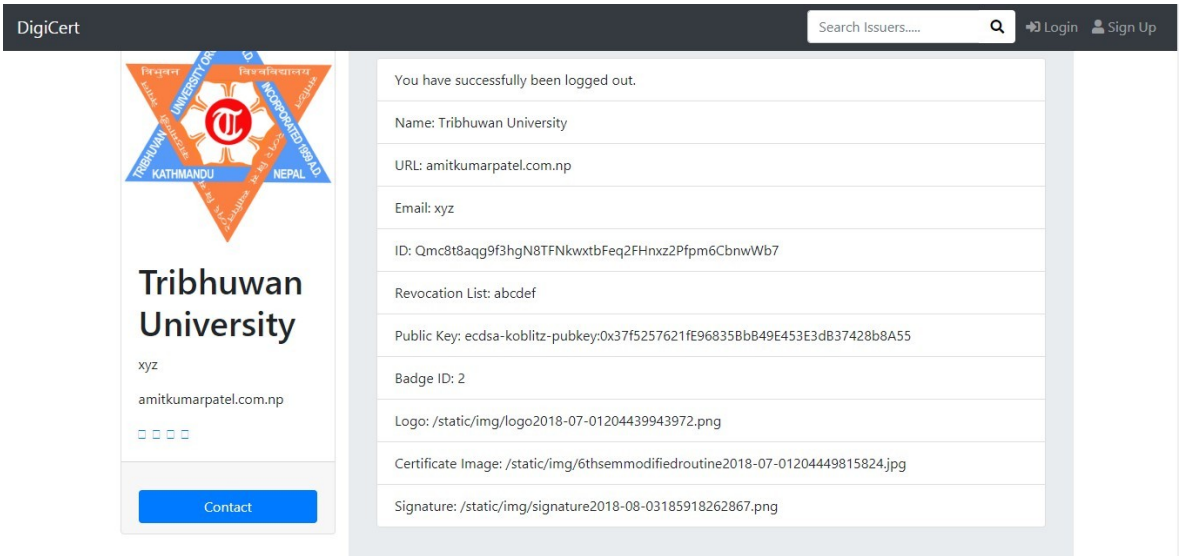

Figure 15: Issuer Profile Page

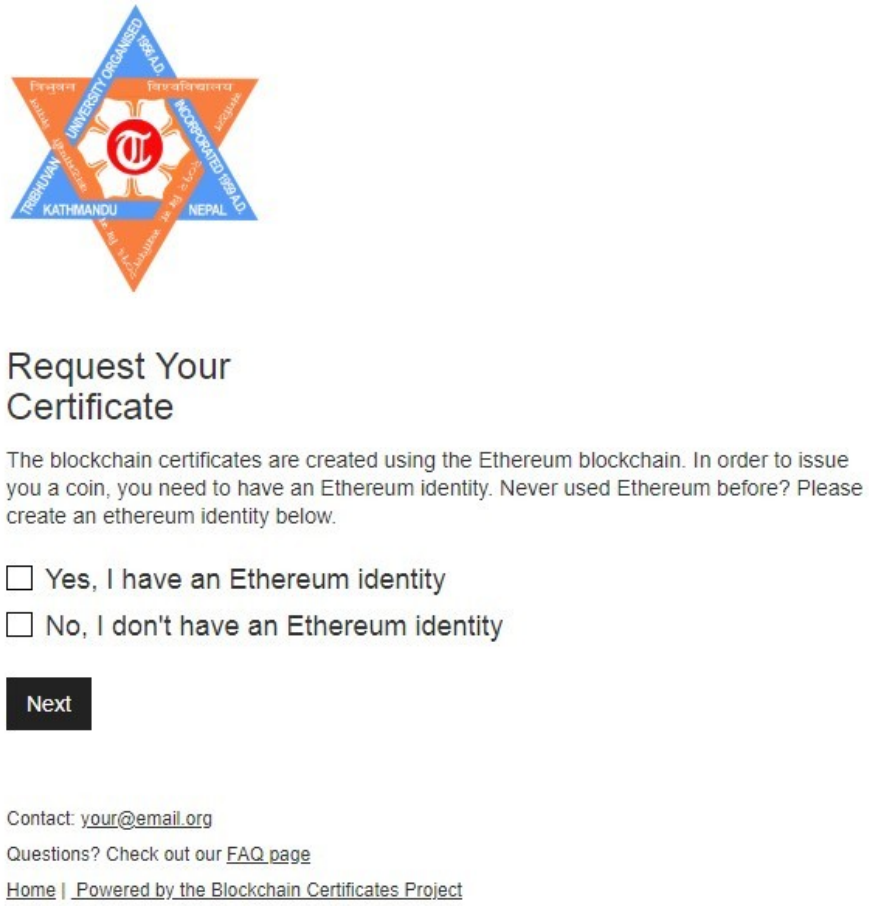

Figure 16: Certificate Request Page

Figure 17: Public/Private key-pair generation

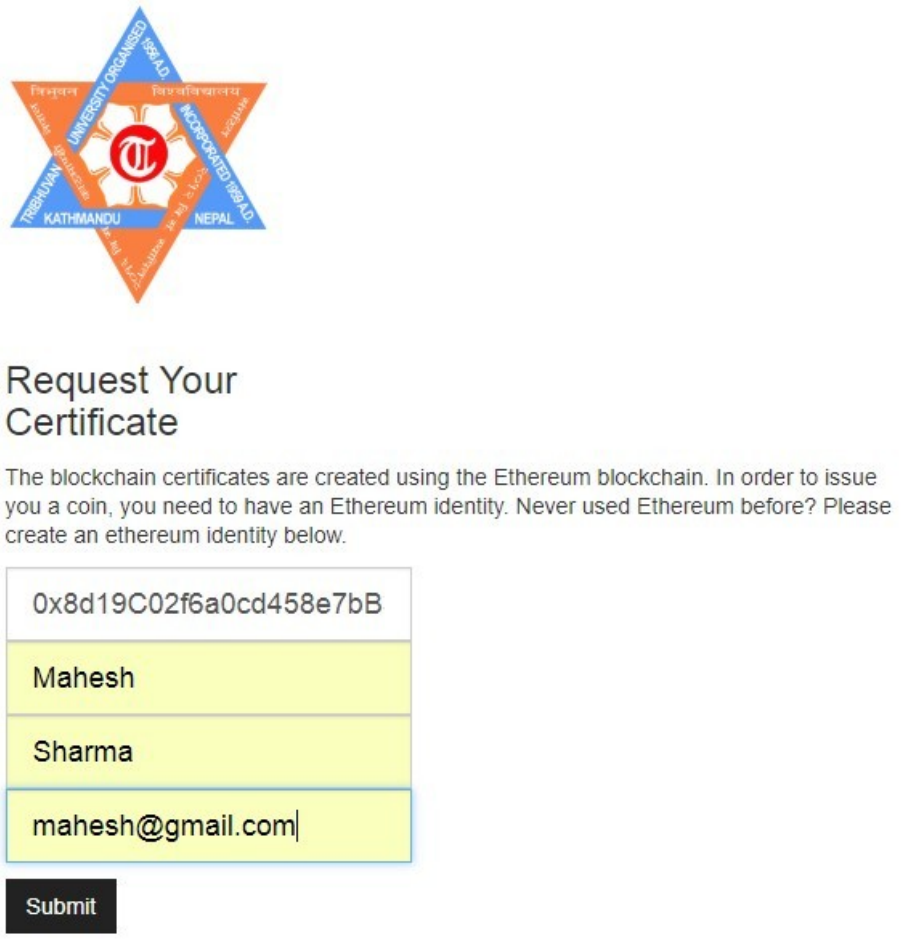

Figure 18: Making request to issuer

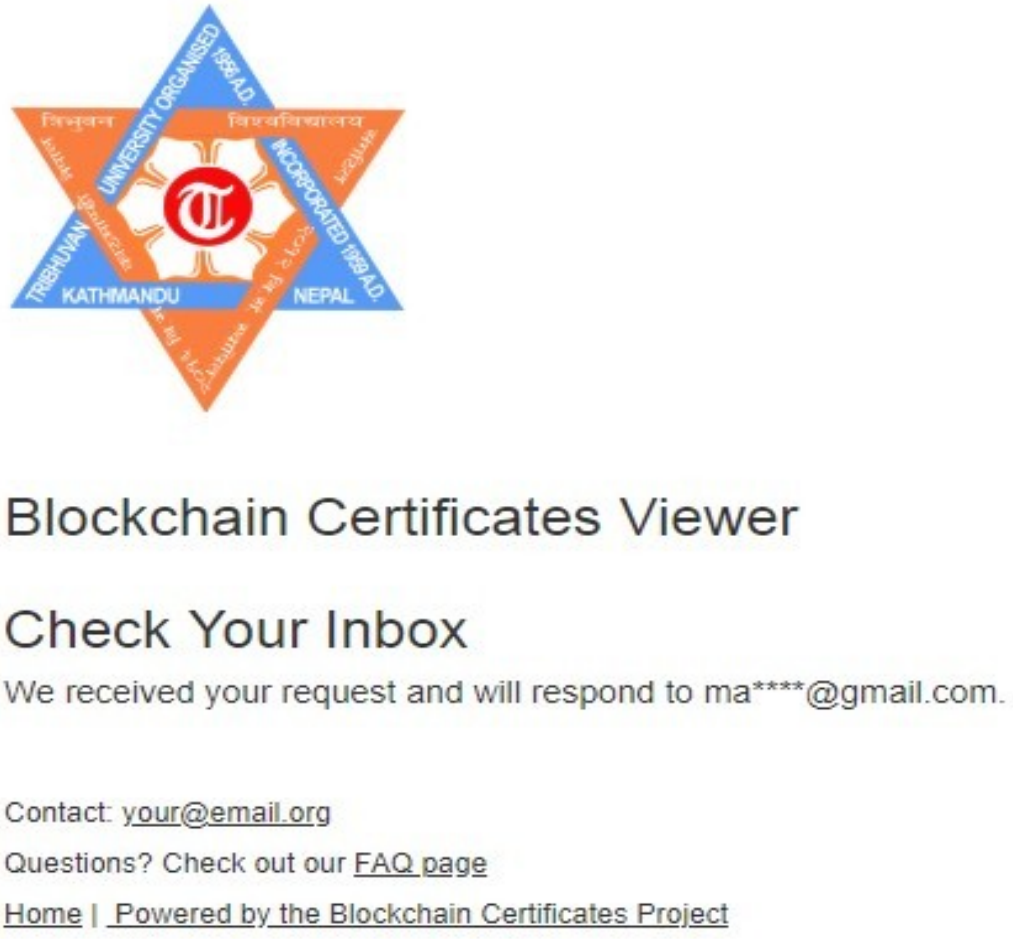

Figure 19: Request submitted

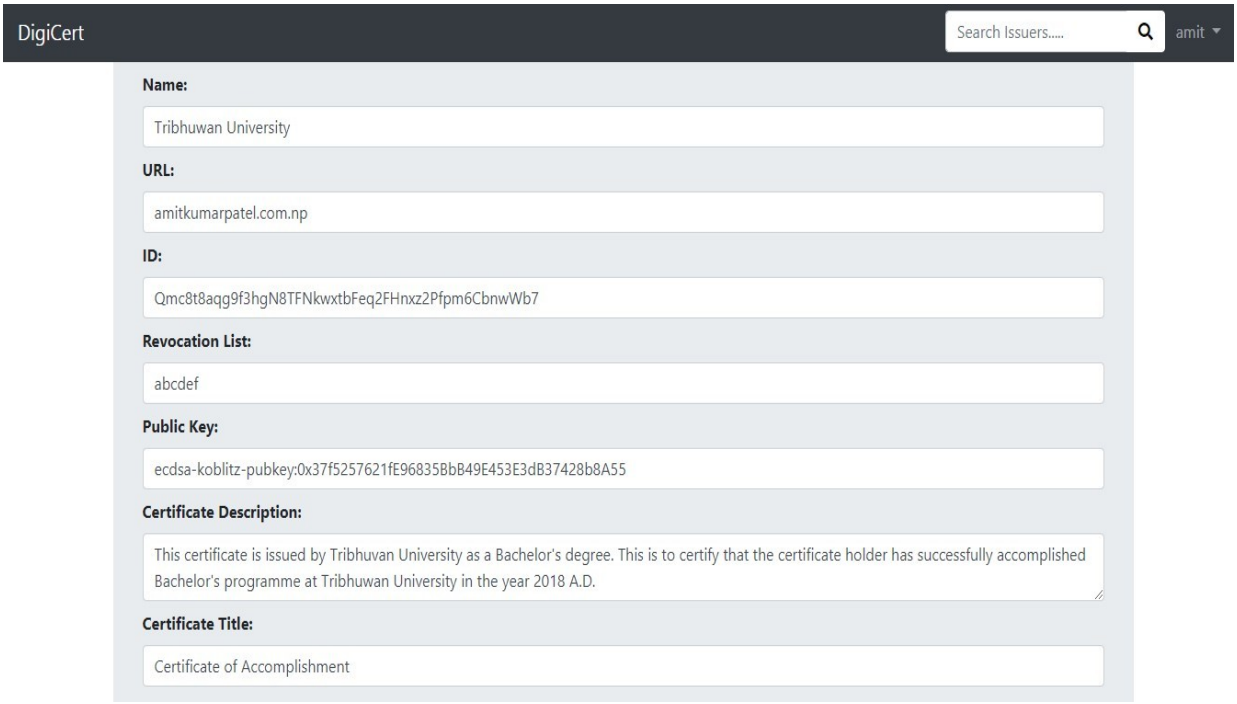

Figure 20: Issuer profile update

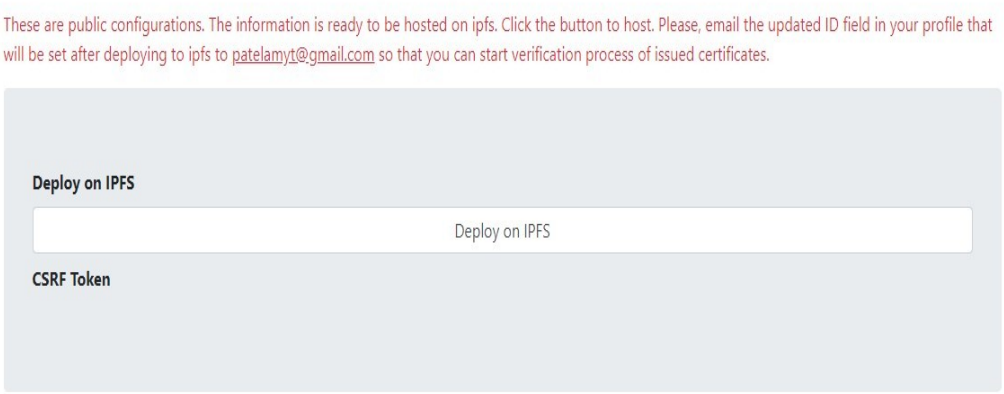

Figure 21: Deploying issuer info to ipfs

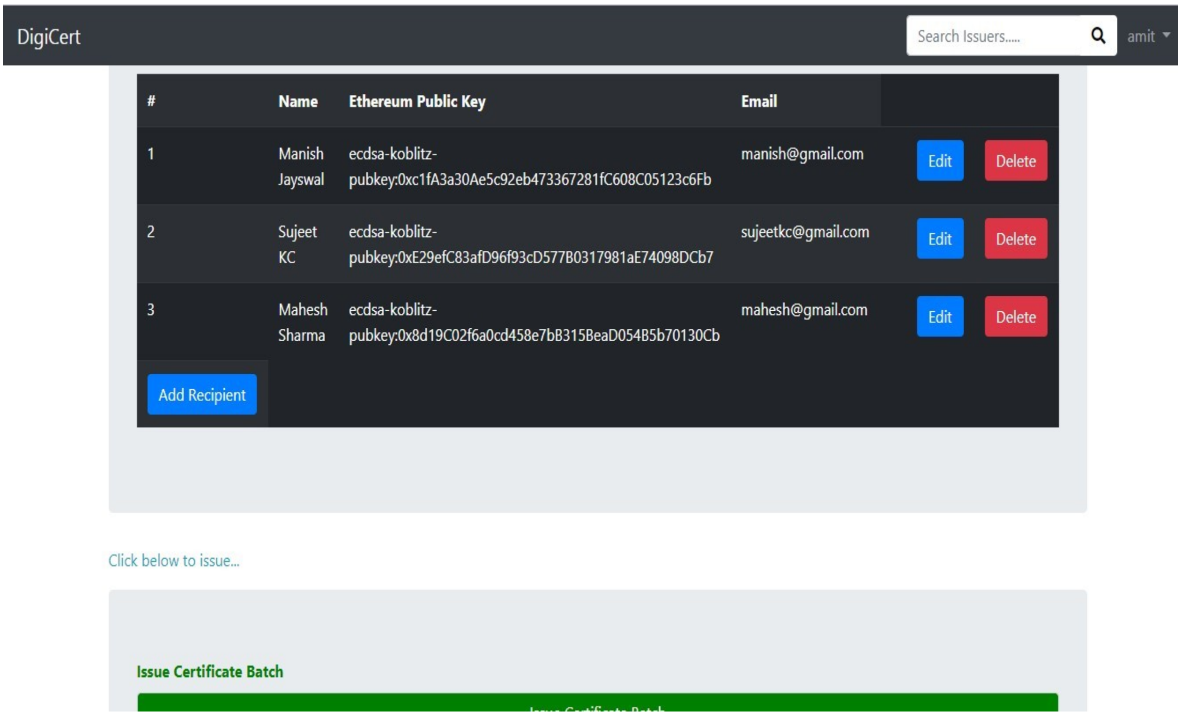

Figure 22: Requests arrived to issuer

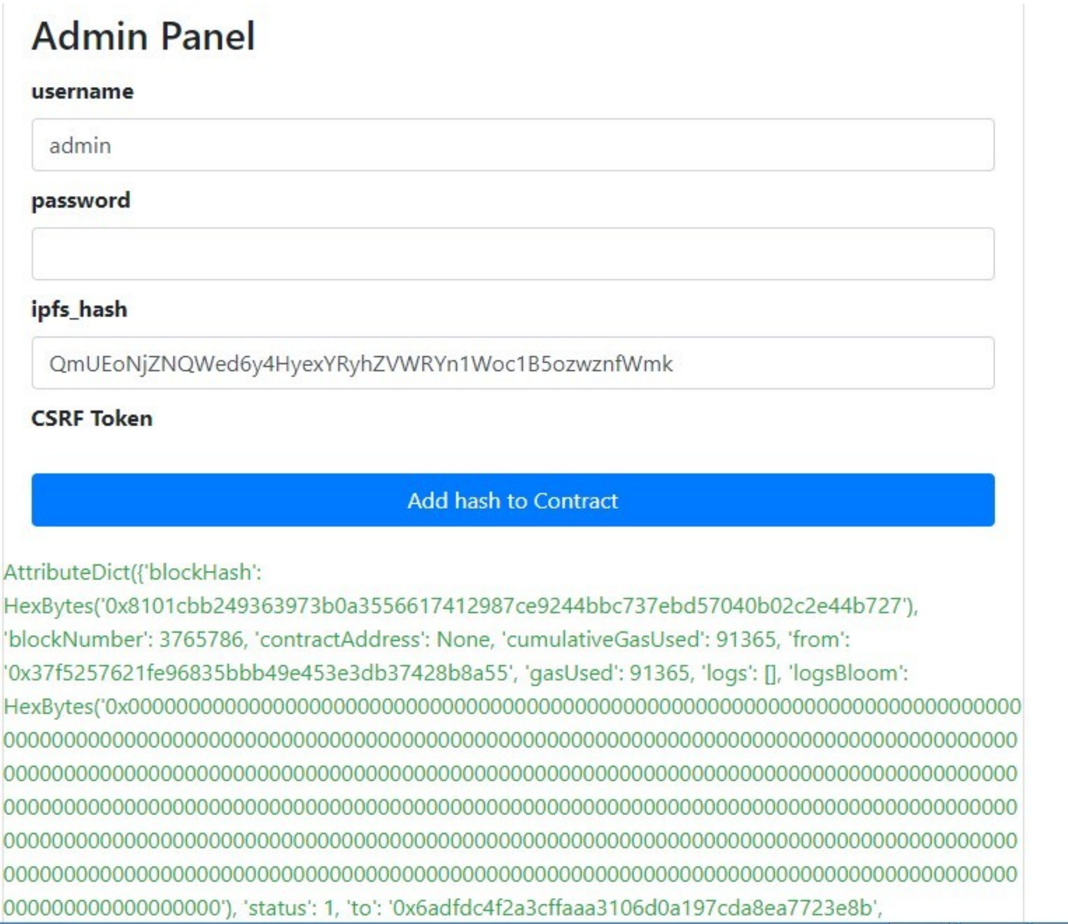

Figure 23: Issuer id added to smart contract

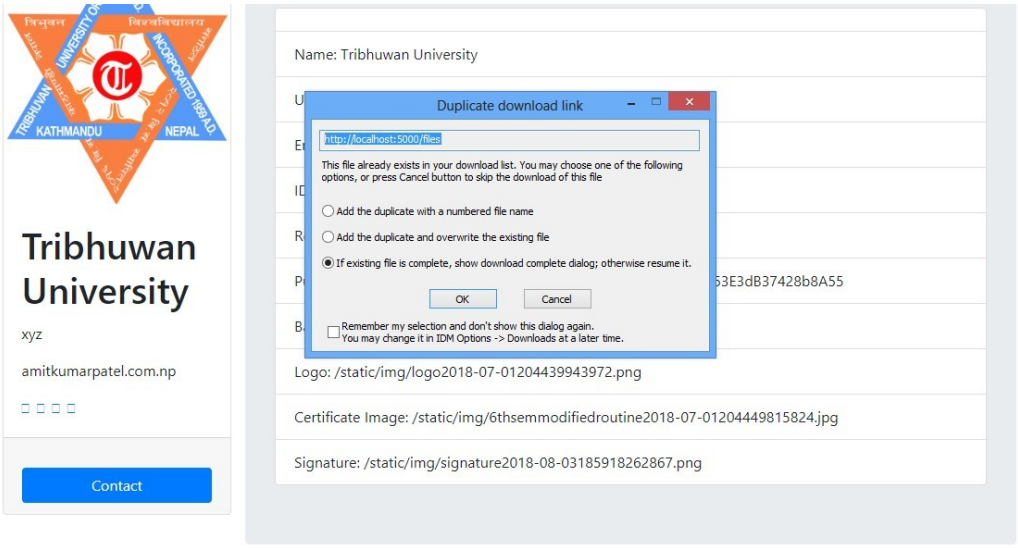

Figure 24: Certificates' zip downloaded after issue

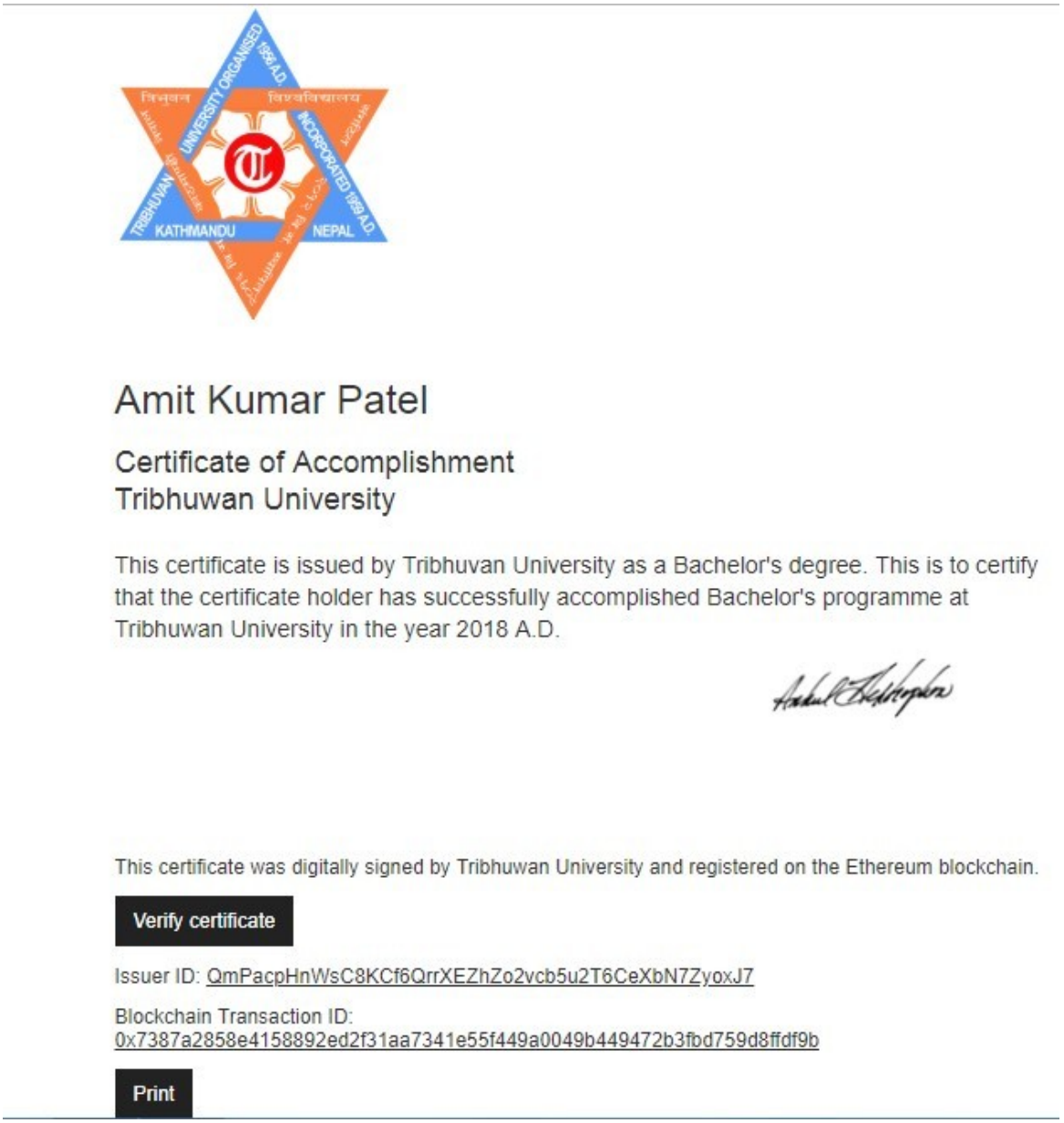

Figure 25: Verification page for issued certificate

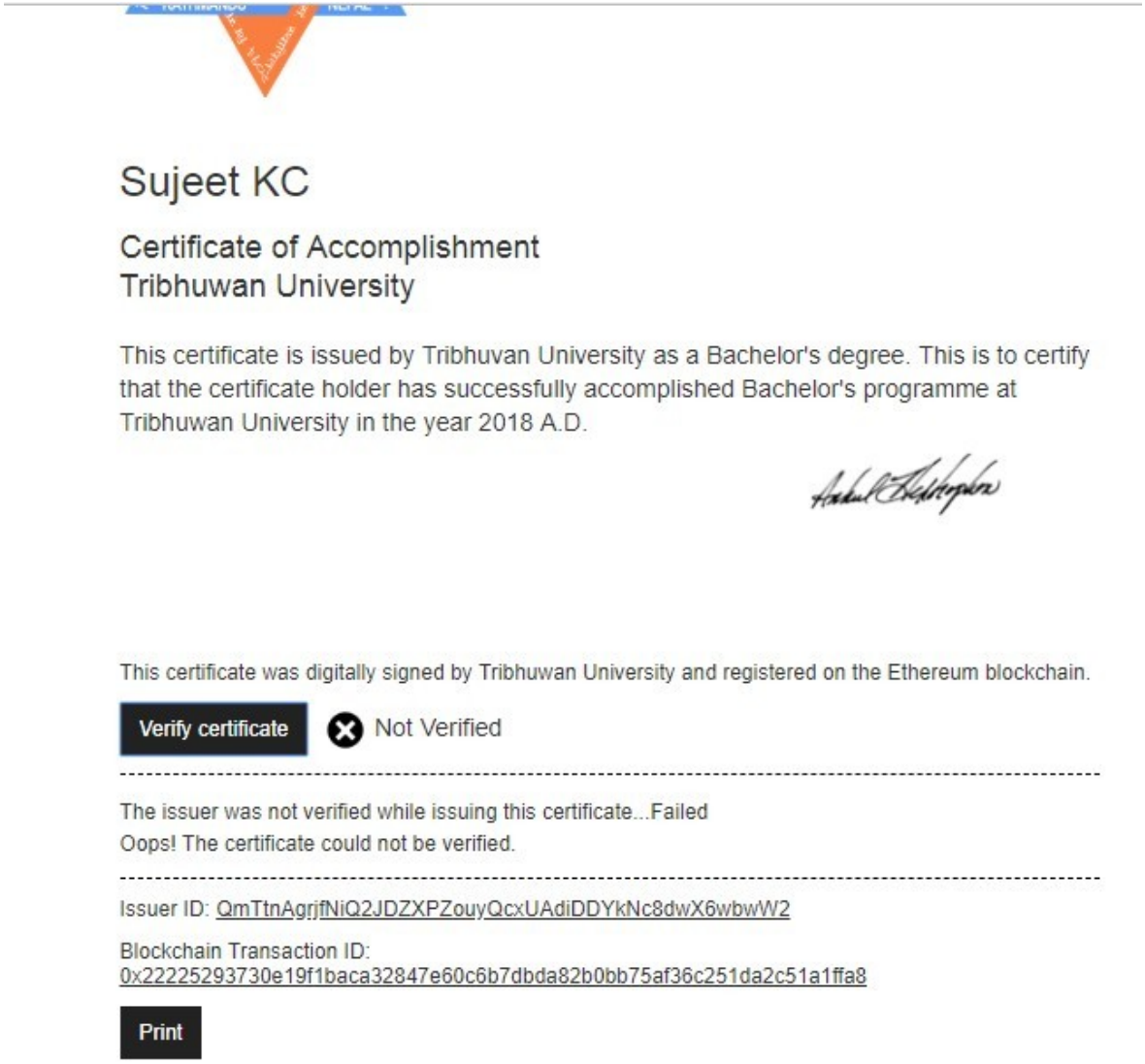

Figure 26: Issuer not verified error

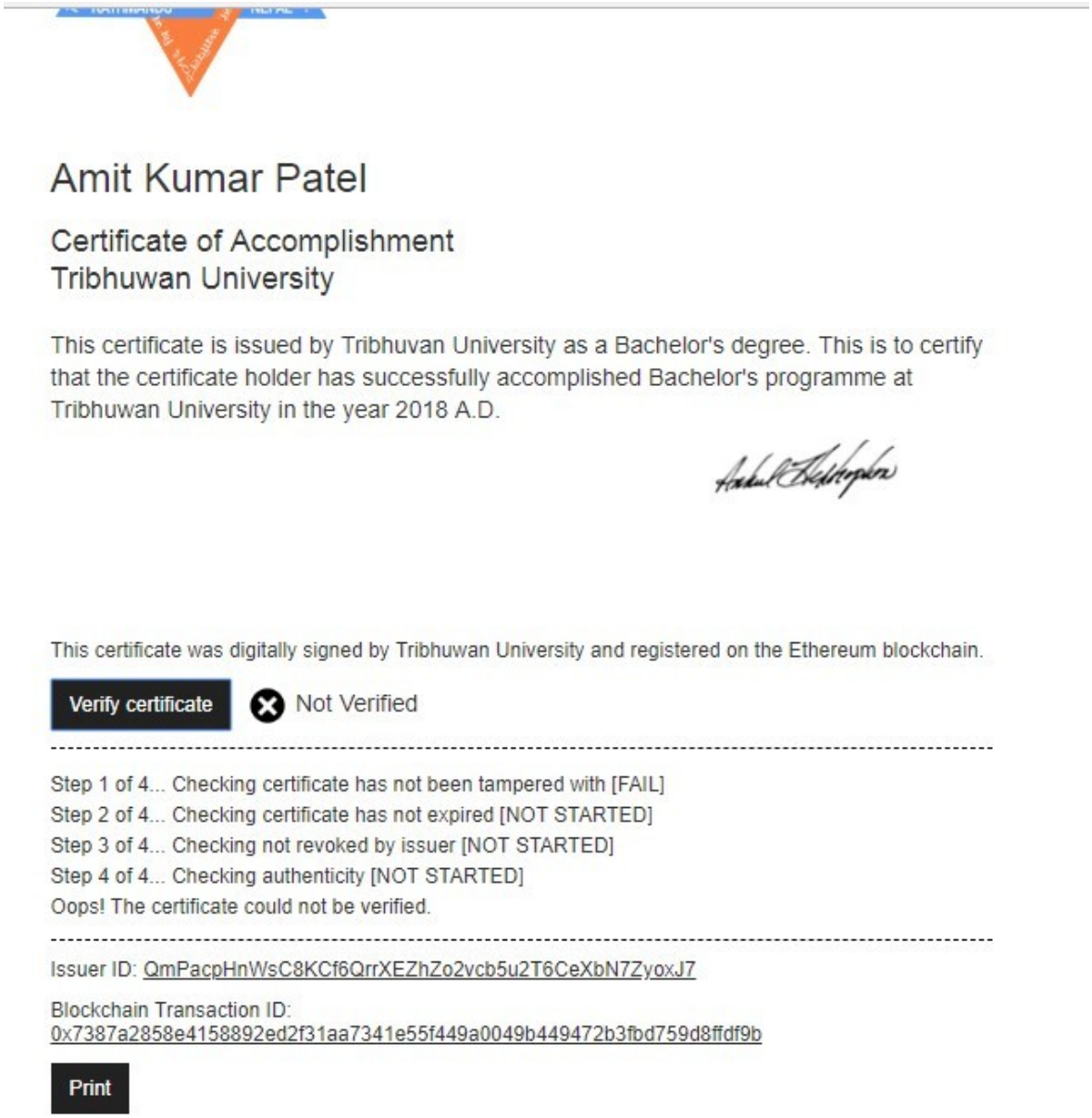

Figure 27: Certificate Tampered Error

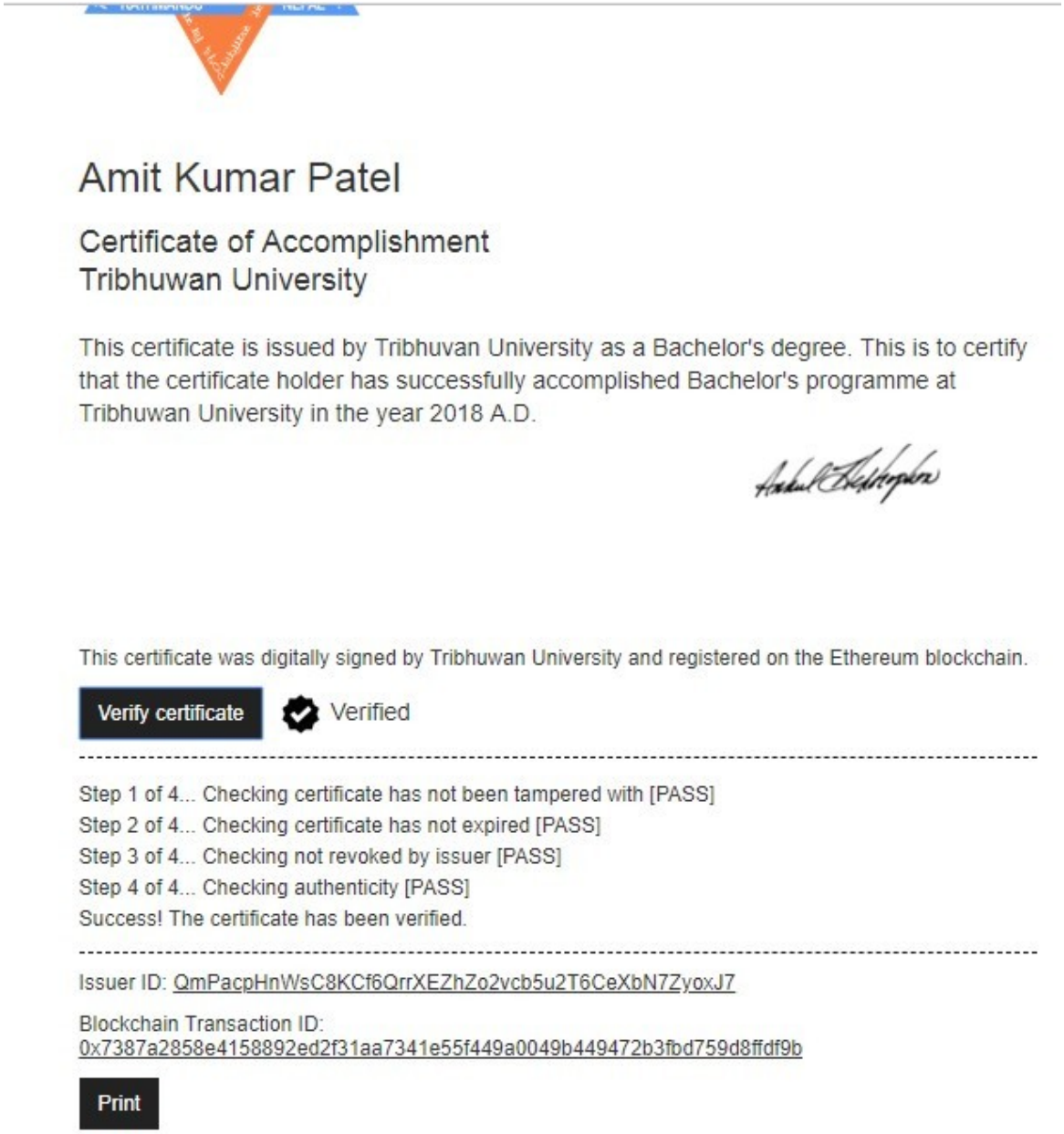

Figure 28: Verified status for certificate