Hypercerts: A Non-Siloed Blockchain-Based Certification Service

João Santos
joao.marques.santos@tecnico.ulisboa.pt

Instituto Superior Técnico
(Advisors: Nuno Santos and David Dias)

Abstract. Educational organisations, such as schools, universities, and enterprises, are known for certifying individuals who partake in their courses. Certification platforms, such as edX or MOOC, offer digital certification services, but are built upon a silo architecture. As a result, the certification data and meta-data tends to be valuable only within that silo and offers no guarantees of permanent storage. We propose to build Hypercerts, a non-siloed certificate service that does not depend on any single fully trusted entity for maintaining certificate data while supporting the typical certification operations, including revocation. Our solution is based on the adoption of a blockchain infrastructure. We then aim at eliminating the dependency on the certificate issuers with respect to the validation of the revocation status of certificates and to the storage of issued certificates. To overcome these challenges, we propose to leverage Ethereum's smart contracts and the storage capabilities of the IPFS distributed file system, respectively. Hypercerts will be built on top of Blockcerts, an open source blockchain-based certification infrastructure.

1 Introduction

Certificates are credentials that serve as proofs a subject's proficiency in a given subject. Formal learning entities, like schools, universities and enterprises are known for certifying individuals who partake in their courses. Those certificates are usually physical goods, printed in paper, stamped by the certifying entity, and that stamp is what confirms its authenticity.

Unfortunately in recent years several cases of certificate forgery have come to light[1] [2]. Poor certificate verification mechanisms would not constitute a serious problem if said certificates had no real importance, but that is not the case. For many professional careers, certification is a decisive factor on whether an individual should be hired, or what his salary should be. For that reason, certification is something that should be taken seriously.

In most cases it is enough to simply list a certificate on our CV and no one will confirm its authenticity since that would require the hiring party to

[1] http://www.thecrimson.com/article/2007/4/26/mit-admissions-dean-resigns-after-fake/
[2] https://www.vice.com/en_uk/article/just-how-easy-is-it-to-get-a-fake-degree

contact the entity who emitted the certificate. That is a laborious and expensive process (there are companies who specialise in professional credentials checking[3], especially if we consider places with a high hiring rate, and companies simply can not afford to do it. What is needed is a better way to issue and verify certificates.

Ideally, certificates should have a set of desirable properties. In particular, they should be easily verifiable, revocable, self-sufficient, have a high degree of interoperability, be informative, shareable, and permanent. Popular certification platforms, such as edX or LinkedIn, manage digital certificates which satisfy some of these properties. However, these platforms are built according to siloed architectures. As a result, the data contained in them has a very low degree of interoperability (the certificates generated by silos are only valuable inside the ecosystem they were created at) and offer no guarantees of permanent storage.

As for potential alternatives to overcome the shortcomings siloed platforms, we find the Open Badges specification. Open badges are certificates that offer well-defined mechanisms for verification, high granularity, and interoperability. The Mozilla Backpack is an application that offers issuing, storage and sharing of Open Badges. However, Mozilla Backpack still falls short of overcoming all the problems that silos have in the sense that it offers interoperability, but it still has control over certificate storage. Blockcerts is a blockchain-based solution compliant with Open Badges specification. However, this system still relies on third parties for verifying the revocation status of certificates and for storing them persistently.

We present a solution that solves the aforementioned shortcomings of Blockcerts. In particular, we propose to build a certification service named Hypercerts. The design of the system includes a novel mechanism for certificate revocation status validation, which leverages Ethereum's smart-contracts, that removes the dependency on the issuer. Additionally, we propose an alternate mechanism for storing certificates that not only removes the dependency on the issuer to maintain said certificates but also ensures their permanence on the network. We do this by taking advantage of IPFS's content-addressing of data.

The rest of this document is structured as follows. Section 2 defines the main goals of this work, and Section 3 presents the related work. Section 4 is dedicated to the presentation of our solution, the Hypercerts certification service. Section 5 is dedicated to the system's evaluation, Section 6 to the planning of future work, and Section 7 to the main conclusions of this proposal.

2 Goals

The goal of this project is to design, implement, and evaluate the Hypercerts system, which is a *non-siloed blockchain-based certificate service* based on Blockcerts. Our aim is to contribute to the Blockcerts community project by solving problems of hosted data dependence on third parties. To this end, we plan to:

– (1) improve the certificate revocation verification process,

[3] `https://www.equifax.com.au/employmentverification/products/academic-qualifications`

– (2) introduce a new mechanism for certificate storage.

In short, the expected results of this dissertation are as follows:

– A complete design of the Hypercerts system.
– An implementation of the system for the Blockcerts blockchain.
– An extensive experimental evaluation of the system.

3 Related Work

This section describes the related work and is organised in three main parts. First, we provide an overview of the state of the art in certification systems (Section 3.1). In particular, we highlight the dependency of these systems on some fully trusted entity and discuss the drawbacks of such dependency to the end-users. Then, in Section 3.2, we introduce the emerging blockchain technology, which has played a pivotal role in decentralising trust for several applications. We introduce the main technical contributions of this technology, and present a brief roadmap of relevant applications built around blockchain. Finally, in Section 3.3, we describe how blockchain has been leveraged in the design of a distributed certification system, named Blockcerts, aimed at achieving complete independence from fully trusted entities, and discuss why this goal has not yet been attained, which motivates our current proposal.

3.1 The Status Quo of Certification Services

In this section, we present the current status of certification services. We start by presenting a general model for certification services and relating it to reputation systems (Section 3.1.1). Next, in Section 3.1.2, we introduce the most common certification services architecture, which is based on information silos, and discuss the inherent problems with this design. Finally, in Section 3.1.3, we present some relevant alternatives to the siloed architecture that have been proposed to overcome these limitations.

3.1.1 General Model of Certification Services

Figure 1 illustrates the workflow of a certification service in its simplest form. Essentially, this diagram represents the process through which an entity (*issuer*) ascertains a set of accomplishments or skills of another entity (*receiver*). An *accomplishment* is achieved if the receiver meets a set of predefined *goals* as determined by the issuer (step 1). The issuer oversees the process through which the receiver attained those goals (step 2), and issues a document, a *certificate* (step 3), which certifies that achievement. The certificate is then made available for the receiver, who can freely share it with other entities (step 4).

To better clarify this model and understand the role of each participant, consider a simple example. Alice and Bob are both Computer Science engineers

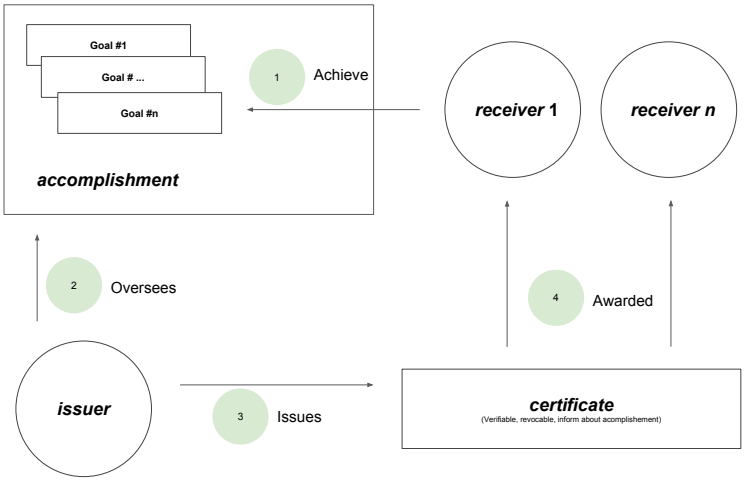

Fig. 1. Basic model of a certification service.

who attended to the same University. During her academic years, Alice was an active member of a student group. She attended many workshops and organised multiple events. Bob, on the other hand, dedicates his time towards building up his programming skills. He participated in and won several coding competitions and was an active member of online programming forums, participating in discussions and helping other programmers. In this situation, Alice and Bob would be considered *receivers*, the *accomplishment* would be finishing their degrees by having a positive score on their subjects (*goals*) and the University would be the *issuer* who oversaw the achievements and issued diplomas, the *certificates*.

Properties of certification services: Ideally, certificates should have a set of desired properties, which are enforced differently depending on the specific certification service implementation. The main of such properties are the following:

- **Revocability:** Revoking a certificate consists on invalidating a valid certificate, meaning that from the point that a certificate is revoked it should no longer be successfully verified. The issuer should be able to revoke an issued certificate. The receivers should also be able to revoke their certificates (if for some reason they no longer wish to be associated with that issuer). There should also be the option for a third party, for instance a government authority, to revoke a certificate.
- **Verifiability:** There should be a well-defined mechanism for certificates to be validated. That is, for any participant in the certification workflow or third party to be able to check the authenticity, integrity and correctness (non-revoked) of any given certificate.
- **Self-sufficiency:** Ideally, none of the steps of certification (issuing, verification, storage of the certificate) should rely on a third party.

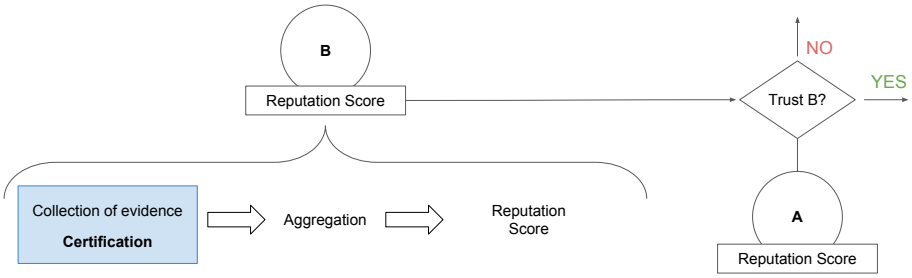

Fig. 2. The role of reputation in a certification system.

- **Compatibility:** Certificates should obey a certain standardised schema and be able to link with each other, even to those that are issued by different entities.
- **Informativity:** Certificates should provide relevant information about the achievement they refer to. Information such as a description of the achievement and the process through which the Receiver had to go through in order to be certified.
- **Shareability:** Certificate *receivers* should be able to share their certificates with any party they wish.
- **Permanent:** Once created, Certificates should not disappear and should be retrievable at all times.

How certification is performed: Usually people associate the issuing of a certificate with the conclusion of a formal learning process, like a University degree. It would be good to be able to issue certificates with a higher granularity. Taking the case of Alice and Bob, they attended the same degree and therefore they will receive identical University diplomas even though they have very different skill sets, considering the extra-curricular activities they carried out during their formative years. There should be a way to generate certificates that deal with the individual skills that they have.

Certification and reputation: Certification and reputation systems are intertwined. This relation is represented by the diagram in Figure 2, which depicts the function of a reputation system. Each entity has a reputation score, which is the metric used to decide whether to establish trust. The reputation score is computed in two steps, collection of evidence and aggregation of that evidence. The relation between certification systems and reputation systems lies on the collection of evidence, since on skills based reputation systems, certificates are one of the types of evidence collected.

3.1.2 Siloed Certification Services

The certification services model presented above can be instantiated in different architectures. Now we present the architecture most commonly adopted in real

world systems: the centralised *siloed architecture*. To do that we analyse some of the most successful certification platforms: edX, a MOOC platform, LinkedIn, and StackOverflow. We list their positive features and then enumerate the desirable functions that are lacking. This latter point will serve as motivation for establishing the need for moving towards a decentralised non-siloed architecture.

The edX platform. In recent years, Massive Open Online Courses (MOOCs)[1] have received widespread adoption. MOOCs are free online learning programs usually composed of videos and interactive platforms to teach and test students. There a number of providers and they tend to be associated with universities or companies, such as Coursera (Stanford), Khan Academy, Udacity (Georgia Institute of Technology, Google, Facebook, Nvidia), Lynda.com. One of the main MOOC providers is edX[4], which is associated with MIT, Harvard University, and more recently, Instituto Superior Técnico. The reason why edX is relevant in this subject is because it is one the MOOC providers that give its participants the ability to receive certificates upon completing a course.

Since 2015, for a small fee, edX offers its participants two types of certificates: Verified Track Certificates, which certify a participant on a particular course, and XSeries Certificates, which confirm a user has received a Verified Track Certificate for each course in a Series of courses. Essentially, a XSeries Certificate is no more than a set of Verified Track Certificates. A Verified Track Certificate contains information about the participant (whose identity is automatically verified using a webcam and a copy of a government issued ID) name of the course, the edX partner institution responsible for the course (usually a university), names and signatures of members of the course's team, an issuing date and more importantly a URL that anyone can visit that verifies the authenticity of the certificate. Coursera and Udacity offer similar verified certificates. edX's certificates are an important and welcome improvement over the traditional paper certificate as they are revocable, easily verifiable and shareable.

The LinkedIn platform. Another platform that offers a good solution for certification is LinkedIn's skill endorsement feature[5]. Users can endorse other users on arbitrary skills, we can look at endorsements as certificates. Those endorsements are displayed on the user's page with the name of the skill and number and names of users who have endorsed. Another important feature of this endorsement system is that gives users a lot of control over the endorsements they receive and give. A user has the option to retract endorsements he made and to hide endorsements he received from someone. There is also the option to disable endorsements all together.

LinkedIn's endorsement mechanism is an improvement over other digital certification mechanisms such as edX. In particular, its certificates (named *endorsements*) have improved revocation functionality, and are more informative because they have much higher granularity. In contrast, in edX only the issuer can revoke

[4] `https://www.edx.org`

[5] `https://www.linkedin.com/help/linkedin/answer/31888/skill-endorsements-overview?lang=en`

the certificate, but in LinkedIn any of the parties can revoke it. Moreover, edX's certificates provide information about the completion of a set of courses while LinkedIn's system provides information about particular skills. LinkedIn even allows for edX certificates to be embedded in a user's profile.

The StackOverflow platform. In platforms such as StackOverflow, users are awarded badges for having desirable behaviours (such as answering forum questions or solving programming challenges). Even though these systems are more focused on reputation building (the reputation of a user on the website is measured by the amount of badges he has) the process through which the badges are awarded is a certification process. The badges certification approach has positive features. First, badges are a great form of certificates with high information granularity. They record information about very particular achievements, and are way of embedding gamification techniques in reputation systems, which has been proven to have positive outcomes, [2,3,4]. Furthermore, badges are revocable, verifiable and very informative.

Main weaknesses of these platforms. So far in this section, we have talked about a few popular siloed reputation systems, namely edX, LinkedIn and StackOverflow. However, in spite of their benefits, these platforms still fall short on attaining all desirable certification properties listed in Section 3.1.1, more precisely:

- **The certificates are not self-sufficient:** In order to view or verify certificates (or badges) issued by these platforms it is always necessary to contact the issuer since the certificates are stored by the *issuer*.
- **Lack of interoperability between certificates:** Certificates issued in one platform can not be recognised or referenced to by certificates issued on a different platform since they have different schemas.
- **The receiver has no control over its credentials:** This is the biggest problem with siloed systems. The certificates issued by platforms such as LinkedIn StackOverflow are only relevant in the ecosystem they were created on and issuers have total control over the information. If LinkedIn or StackOverflow cease to exist, all the users' certificates they Issued disappear and receivers are no longer able to share their credentials. Furthermore, having user data spread across so many siloed systems results in each one of them holding only a subset of information about a given user which results in non of them having a holistic view of its users.

We argue that a truly decentralised system is needed in order to give the user full control over his credentials. Systems such as LinkedIn or StackOverflow only give the illusion of control. It is true that the user has some control over the visibility of his certificates and achievements, but the platform has the last say in the sense that the platform operators can delete a user's entire history. Moreover, all the information contained inside these silos has no meaning in any platform other than the one it was created at, so a user has no choice to transfer the data from one platform to another due to the lack of compatibility between

these platforms. Siloed systems are characterised for having total control over the information they hold and not making that information compatible with any other platform. Next, we are going to introduce relevant platforms that aim at overcoming the limitations of siloed architectures.

3.1.3 Toward Breaking Down Siloed Certification Services

In this section we are introducing two certification systems, Mozilla Backpack[6] and Blockcerts[7], which aim at taking the best features of siloed systems, described in Section 3.1.2, while overcoming some of their shortcomings. Both systems are implementations a standardised certification specification called Open Badges[8]. We will start by introducing the Open Badges standard, followed by an overview of Mozilla's Backpack. We conclude that while Mozilla's Backpack is an improvement over siloed systems in the sense that the certificates it holds are standardised and can interact with each other, it still has the problem of relying on third parties for certificate verification. Finally, we present Blockcerts which is an OpenBadges compliant system unique in its implementation as it relies on the Bitcoin blockchain for certificate validation.

The Open Badges specification. Open Badges inform about an individual's achievements (and skills) and consist of encoded relevant information about each achievement in a digital file. An Open Badge is just like a certificate, only instead of being a PDF file, piece of paper, or a digital token that lives on a website (and is only relevant in the context of that website) is a file that contains relevant information about a certain achievement or skill, has the capability of being easily verifiable and can be linked to other Open Badges issued by different entities. For a file to be considered an Open Badge, it has to obey the Open Badges specification[9], which defines mechanisms for how information about accomplishments and skills should be wrapped (a schema) and packed into portable image files, badges, and the infrastructure for the validation of those badges.

An Open Badge is a JSON-LD[10] file, which allows for files to link to reference each other file. It comprises the following information about who issued the badge (Issuer), who received it (Receiver) and about its meaning (Assertion):

- **Issuer:** An *issuer* is identified by a Profile class object that contains the *issuer*'s ID, the name, email and URL. It can also contain a Public Key if the *issuer* wishes to sign its certificates.
- **Receiver:** The Receiver is identified by an Identity object that contains an identity field, that can be an Email for instance. This field can be plain text string, however that is not advised due to privacy issues. Instead, the identity field can be a hash of the identity.

[6] `https://backpack.openbadges.org`
[7] `http://www.blockcerts.org`
[8] `https://openbadges.org`
[9] `https://www.imsglobal.org/sites/default/files/Badges/OBv2p0/`
[10] JSON for Linked Data

– **Assertion:** This is the object that provides context about the achievement or skill. It contains an ID, a Badge class object (that describes the type of badge being attributed), a Verification Object (that has instructions for verification) and the issuing date. It can also contain an expiration date and an Evidence object that gives context to the tasks the receiver completed in order to be awarded the badge.

Since the badges are JSON-LD files they can reference other badges' fields, for instance an Evidence field in a badge can be a reference (URL) to the Evidence object that lives in another badge.

The Mozilla Backpack system. Mozilla Backpack was a great improvement as far as certification goes. It implements a standard, Open Badges, that has well-defined mechanisms of verification, that has a high semantic granularity, that makes use of the benefits of gamification, that allows for a high degree of interoperability.

Open Badges allow for future extensions, meaning that each issuer is free to customise its badges in order to fulfil its needs. Badges can be Issued by anyone by using what is called a Bakery, which is essentially a program that receives the information listed above and outputs a badge. This output can come in one of two forms. It can be a file, JSON-LD, that the issuer hosts (on a web server controlled by him and shares a link, URL, to it with the receiver) or it can be a PNG image file that contains the JSON-LD embedded in it. In the first approach the issuer has control over the badge and the verification and authenticity of the badge is proven because the issuer is the one hosting the badge (so anyone who visits the URL and sees that the badge exists has an automatic confirmation of its existence). This verification approach is similar to the one used by edX. In the second approach the receiver is the one who controls the badge and the way to store it is to use a Wallet, an application designed to keep Open Badges. In order to verify the authenticity of the badge, the badge must have been signed by the issuer. The most popular Wallet application is the Mozilla Backpack, which allows not only for users to store their badges (so far it has more than 1 million badges stored) but also to share them across the web with any applications that have Open Badges support.

The Mozilla Backpack is a centralised system that stores badges. It solves the interoperability problem that occurs in information silos, however a user is still dependent on Mozilla's goodwill to continue storing their badges. Granted that users can always keep local copies of their badges, but if they don't and the Backpack platform ceases to exist there will be no record of those badges. Up until recent years this problem would not have an apparent solution. However, that is no longer the case, as Blockchain technology has paved the way for truly decentralised applications.

Blockchain-based certification infrastructures: Decentralisation is the key for certification, which is the path being taken by Blockcerts[11] by legeraging blochchain

[11] `https://www.blockcerts.org/`

	Revocability	Easily Verifiability	Self-sufficiency	High Compatibility	Informativity	Permanency	Decentralisation
Paper	No	No	No	No	No	No	No
edX	Yes	Yes	No	No	No	No	No
LinkedIn	Yes	Yes	No	No	Yes	No	No
StackOverflow	Yes	Yes	No	No	Yes	No	No
Mozilla Backpack	Yes	Yes	No	Yes	Yes	No	No
Blockcerts	Yes	Yes	No	Yes	Yes	No	Yes
Hypercerts	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 1. Comparison between the several certification systems presented

technology. Specifically, Blockcerts is defining the way blockchain technology can be used with Open Badges and aims to provide "the open standard for blockchain certificates". However, in spite of its potential, it currently cannot provide for self-sufficiency and permanency support.

Table 1 summarises a comparison between the several presented certification systems. We see that siloed certification systems fall short on a lot of desirable properties. Non-siloed reputation systems, namely, Mozilla Backpack and Blockcerts, both based on Open Badges, who are an improvement but still have a high degree of reliance on the issuer for storage and verification processes. What is needed is a system that takes the best features of all the ones presented so far, ease of validation, high granularity, interoperability, gamification, decentralisation and combines them into one cohesive ecosystem. This is exactly what our solution (Hypercerts) aims to achieve.

Since both blockchain and Blockcerts are particularly relevant in this proposal, we focus about both these technologies in detail in the next sections before introducing our propose solution in Section 4.

3.2 Overview of the Blockchain Technology

In this section we present the emerging blockchain technology, which is a building block of both Blockcerts and Hypercerts, our proposed system. We start by providing a brief historical context of blockchain (Section 3.2.1). Then, we discuss the main features of blockchain as it appeared firstly in the Bitcoin cryptocurrency (Section 3.2.2), and how it has been improved in alternative cryptocurrency systems (Section 3.2.3). Lastly, we introduce two relevant systems of the blockchain ecosystem: Ethereum (Section 3.2.4) and IPFS (Section 3.2.5), both of which will be used to build our proposed non-siloed certification service.

3.2.1 Historical Context of Blockchain

Blockchain technology was made popular with the creation of Bitcoin [5], in 2009. Since the 1980s that there had been proposals for crypto-currency systems and in 2005 for decentralised consensus protocols. The proposed crypto-currency systems were never widely adopted due to the need of a centralised platform, as for the decentralised consensus protocols, they never became a reality as it

was unclear how to carry out the implementation. Bitcoin's white paper,[5], was published in 2009 by Satoshi Nakamoto (a pseudonym) and it was the first successful implementation of a decentralised crypto-currency system.

Bitcoin was the first truly decentralised crypto-currency, it uses cryptographic properties to solve the double-spending problem [12] and allows for parties to engage in transactions without having to trust each other. In the backbone of Bitcoin lies a blockchain, which is a distributed ledger that keeps track of every valid transaction that ever took place in the network. Transactions are validated and packed into blocks that are then added to the ledger by nodes (miners) that run a consensus protocol, proof-of-work in the case of Bitcoin. Bitcoin's blockchain has full auditability, meaning that any node can verify the validity of any transaction and correctness, meaning that all the transactions on the longer chain are valid.

The introduction of a blockchain in the Bitcoin design constituted a true technical breakthrough. Although the blockchain concept is mostly associated with Bitcoin, since it played a pivotal role in the large scale adoption of blockchain technology, it has been applied to multiple uses, which range from Internet of Things applications [6], to new Byzantine Fault Tolerant protocols [7], to alternative crypto-currency systems. With this regard, Bonneau et al. [8] offer a technical overview on Bitcoin's architecture and protocol, and Barber et al. [9] identify possible security vulnerabilities and solutions.

3.2.2 Technical Overview of the Bitcoin's Blockchain

Bitcoin proposed an alternative mode of establishing trust between transacting parties. On traditional financial transaction systems, such as banks, two transacting entities have trust on a centralised element, the bank. Let us look at the example of a normal wire-transfer. User A sends money to user B. Both A and B are confident that the bank will withdraw the right amount of money from A's account and place that same amount in B's account. In this traditional trust model the bank is an essential entity in the process as it constitutes the only source of trust between the two transacting parties. Bitcoin's approach is different. Instead of asking the two parties to trust each other, it uses a proof-of-work system whose cryptographic properties assure transactional correctness as long as honest nodes control the majority of the CPU power of the network.

An electronic currency is nothing more than a piece of digital data and one of the features of digital data is their easy replication. For that reason, an issue that an electronic currency has to tackle is the double spending problem. That is, ensuring that a given node is not able to spend the same token (coin) more than once. The way to solve this problem without resorting to a centralised system is to create a ledger of transactions and get all the nodes to agree on the content of that ledger. To that end, a consensus protocol is used, the Bitcoin consensus protocol, also known as the Nakamoto Consensus if based on a proof-of-work.

[12] `https://en.bitcoin.it/wiki/Double-spending`

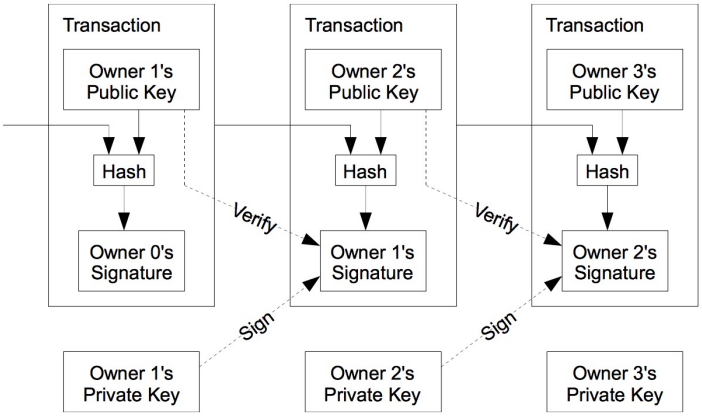

Fig. 3. Bitcoin's transactions [5].

Transactions and proof-of-work: A Bitcoin transaction consists on nodes exchanging unspent coins and is identified by chain of digital signatures. To complete a transaction the sending node digitally a hash that identifies the past transaction and the public key of the next owner. This ensures that each transaction is linked to the ones preceding it, creating a chain. The next key challenge is getting all the nodes to agree on the set of past transactions, in order to prevent double-spending. To that end transactions are packed into blocks, and added to an append-only data structure, the blockchain. Figure 3 depicts the format of Bitcoins transactions. Each transaction is linked to the one before it and is signed by the owner of the UTXOs used as input in that transaction.

The mining process: To add a block to the blockchain, nodes have to mine the block and the way to do that is to solve a mathematical challenge with a pre-defined difficulty level, this is called a proof-of-work [10]. The first node to solve that challenge mines the block, permanently adding it to the Blockchain. Each time a node mines a block, a new Bitcoin (coin) is created and given to that node. This constitutes an incentive for nodes to mine. The algorithm is designed to automatically adjust the challenger's difficulty in order to regulate the currency issuing rate. All the mined blocks are made of transactions that are linked to each other, which means that if a dishonest node was to change a past transaction that change would be eventually be detected by an honest node.

Sometimes a node will mine a block that has already been mined by another node but due to network latency the first is unaware of this. In this situation a fork occurs. Both of these blocks are propagated through the network and eventually a new block will be mined and appended to one of those forks. The network will choose the longest fork as the standing one and the blocks on the branch that is not chosen are discarded. Those blocks are called stale blocks. The probability of the occurrence of a fork of depth n is $O(2^{-n})$ which gives miners

a high degree of confidence that the transactions they process will eventually be included in the ledger.

Bitcoin's consensus protocol: Bonneau et al. [8] provide an overview of Bitcoin's consensus protocol stability features. It ensures consensus, eventually all correct nodes will agree upon the set of blocks that exist on the blockchain, it has exponential convergence, giving confidence to miners that the blocks they mine will be eventually be added to the blockchain, it has liveness as new blocks will continue to be added, motivating miners by enabling them to collect fees, it has correctness since the longest chain (the active one) will not include stale blocks and it has fairness in the sense that the probability of a miner mine the next block is directly proportional to the amount of computing power that miner has.

3.2.3 After Bitcoin: Alt-coins and Other Consensus Protocols

Following Bitcoin's success, a large and active community has emerged and worked on modifications and extensions to the original protocol to improve some of Bitcoin's shortcomings like performance, introduce new consensus protocols, add new features and even new banking systems [11]. These modifications and extensions, which are usually forks of Bitcoin or abstractions that sit on top of existing blockchains, are called "alt-coins" [8]. This section provides an overview of systems that emerged since Bitcoin's inception.

Evening out the distribution of mining power: One of the criticisms of Bitcoin's proof-of-work mechanism is that is motivates an uneven distribution of mining power. This happens because the time it takes to solve the computational challenge can be decreased by using more CPU cores concurrently. To address this problem, Litecoin[13] introduces a proof-of-work mechanism that is not solved faster by adding CPU cores. Ethereum also tackles this issue (see Section 3.2.4). Other systems propose completely different consensus mechanisms such as proof-of-space [12] where nodes are rewarded by storing data, rather than performing computations; this is the case with Permacoin [13] and FileCoin[14] crypto-currency whose goal is to leverage the blockchain to store data. Blockstack [14] uses the Bitcoin blockchain to provide a global naming service (like DNS). Peercoin [15] proposes a Proof-of-Stake mechanism where nodes' stake in the network is calculated based on the age of the coins they own and is based on the premise that a node will act in its own best interest, so if has a high stake in the network he will be well behaved and thus contribute to a good operation of the network. DDOS-Coin [16] proposes a malicious consensus mechanism, proof-of-denial of service, where nodes are rewarded by proving they participated in a DDOS attack. Another consensus mechanism include proof-of-bandwidth [17], proof-of-luck [18]. Some crypto-currencies proposals have focused on improving Bitcoin's transaction rates and scalability, which is the case

[13] `https://litecoin.org`
[14] `https://filecoin.io`

of Bitcoin-NG [19,20], and others, such as ZCash [21] and HAWK [22], have focused on transactional privacy [23].

Enabling contract customisation: One of the shortcomings of Bitcoin was the lack of contract customisation, as while it did allow for parties to create some rules regarding a given transaction (such as requiring a third party to approve the transaction), doing so required learning a not very intuitive scripting language and the functionality was extremely limited. In particular, it only allowed for a very simple form of smart contracts; smart-contracts are programs that run on the blockchain and allow for the creation of special transaction rules. Ethereum [24,25] is a crypto-currency blockchain, similar to Bitcoin in many ways, that consists of a smart contract and decentralised application platform. It has a Turing complete programming language which allows for users to create distributed applications while making use of the features of the blockchain.

3.2.4 Ethereum and Smart-Contracts

The alt-coins introduced in the section above aim at solving some of Bitcoin's shortcomings such as scalability and lack of functionality. However, they are highly optimised for one use case and lack flexibility to serve other purposes. Ethereum aims at solving this limitation by offering smart-contracts. A smart-contract is a script that runs in the blockchain. It has Turing completeness, thus effectively offering a state-machine that runs on the blockchain.

Regarding its architecture, Ethereum [24,25] is built on many of the same principles as Bitcoin. It is a blockchain based system that can also serve as a crypto-currency. It also uses a proof-of-work mechanism as consensus protocol, albeit with a difference, while Bitcoin's consensus protocol challenge lies on CPU power, Ethereum's lies on memory requirement. The goal of this different implementation is to make the network more democratic. One of the consequences of Bitcoin's consensus protocol was the creation of mining farms, i.e., infrastructures provided with Bitcoin mining specialised hardware, which resulted on an uneven distribution of the mining power among active nodes. Memory is something that is extremely optimised on pretty much every electronic device nowadays, much more than CPU power, so this measure should empower lighter miners to the detriment of more powerful ones.

Ethereum provides an API for developers to apply arbitrary logic to smart-contracts. Its transactions can be of one of two types.

- **Private**: Are controlled by an Elliptic Curve Cryptography (ECC) private key and allow for currency transaction.
- **Smart-contract**: These are transactions controlled by the logic programmed into a smart contract.

Both types of transactions can interact with one another (a private transaction can interact with a smart contract and the latter can also engage with a

private transaction). From a programming stand point, one can look at smart-contracts as objects in the Ethereum universe, these objects can interact with each other to fulfil any business rules required by an application.

A smart-contract has to define an upper limit of computing power it will use. This limit of computing power is knows as Gas. The amount of Gas a program has is what defines for how long it can be ran by Ethereum nodes. Gas costs money and miners are rewarded with that Gas. This condition is extremely important to stop programs that halted and malicious programs designed to enter an infinite loop. If a program runs and terminates successfully and does not spend all the Gas, the remainder is returned to the smart-contract, and can be used in future executions. On the other hand if a smart-contract runs out of Gas before ending its execution, all the changes made by the smart-contract are rolled back (to prevent inconsistent states) but the Gas is not returned to the smart-contract, it is given to the miners.

Some of Ethereum's properties can be very useful to build a non-siloed certification service. As described in Section 3.3.2, Blockcerts' current method for verifying if a certificate has been revoked depends on a third party in the sense that it relies on consulting a certificate revocation list which is owned and maintained by the issuer. Ethereum can be utilised to solve this problem. For every certificate, or batch of certificates issued, a smart-contract can be created that keeps track of those certificates' revocation status. The issuer and the receiver would have permissions to change the revocation status of certificates (Section 4.2 details the rationale behind who can revoke certificates) and from there anyone who wanted to verify the revocation status of a given certificate would do so by checking the state of the corresponding smart-contract.

3.2.5 The IPFS File System

Storing data on a blockchain is inefficient and expensive. IPFS is useful for a wide range of applications, one of them being blockchain. IPFS (Interplanetary File System) [26] is a peer-to-peer distributed file system. The motivations behind its creation are a better use of bandwidth, the ability to permanently addressing content and moving the web towards a distributed architecture. As the web grows the amount of bandwidth required to make it work also increases and the HTTP file transfer model that is widely used today does not optimise bandwidth usage. If 50 people in a classroom all download the same file from Dropbox at the same time, 50 independent requests are made to backbone of the Internet. By addressing that same file with IPFS nodes can download the file from other nodes that have the same file and if there is a node in your network who has the requested file you do not even need an Internet connection. This is a very important feature that can change the way applications work.

Rather than being addressed by their location (IP based link), IPFS files are addressed by their content in the form of a self describing cryptographic hash (a multihash[15]) that is immutable and permanent. This is a desirable feature for

[15] `https://github.com/multiformats/multihash`

Hypercerts as it removes the dependency on an issuer to maintain a link to a file. Thus, to address the storage limitation of blockchains, a user can store large amounts of data on IPFS and reference that data in blockchain transactions, by using the multihash.

IPFS is very efficient in the way it handles data. Each node only needs to store the blocks it is interested in and when a node is searching for a file it queries the network for nodes who have that file. Files are kept track of by using a Merkle DAG [27], a binary tree where each node is created by hashing its two child nodes, that allows for reliable verification of integrity in large data sets. This is the same data structure behind Git, Bitcoin, and BitTorrent. IPFS is built on top of IPLD, a data format that works as an abstraction layer on top of all the data it represents, which allows for IPFS clients to access data transparently, regardless of the type. This allows for extreme flexibility in the way content is linked, allowing for instance an IPFS client to traverse into an Ethereum smart-contract and check the value of its variables – in a string based link, just like a file path in a Unix based system – without the need to run an Ethereum node. Next, we present Blockcerts in more detail.

3.3 Blockcerts: Overview and Weaknesses

Blockcerts is an open-source community-based project that provides a standard for a decentralised implementation of an Open Badges Specification compliant solution for issuing, distributing, and validating certificates. Next, we provide an overview of Blockcert, and then discuss some of its shortcomings.

3.3.1 Overview of Blockcerts

Blockcerts is currently being used to issue certificates by the MIT Media Lab[16], MIT Global Entrepreneurship[17] and Laboratorio para la Ciudad[18]. Just like in Open Badges, the certificates issued by Blockcerts can serve arbitrary purposes such as academic credentials, professional achievements and soft-skills acquisition. Certificates can be issued as singles or in batches, the latter making more financial sense as we will see later.

Figure 4 presents Blockcerts' architecture and workflow (it represents steps 3 and 4 of Figure 1, steps 1 and 2 are out of the scope of the specification). Its workflow is divided into two main steps: issuing and distribution. The decentralised nature of Blockcerts comes from its reliability on blockchain technology. At the time of this document's writing, Blockcerts uses the Bitcoin blockchain for verification purposes, but the goal for future improvements is to make the platform blockchain agnostic, meaning that it would be able to work with any blockchain.

[16] `https://coins.media.mit.edu`
[17] `http://certificates-bootcamp.mit.edu`
[18] `http://certs.labcd.mx`

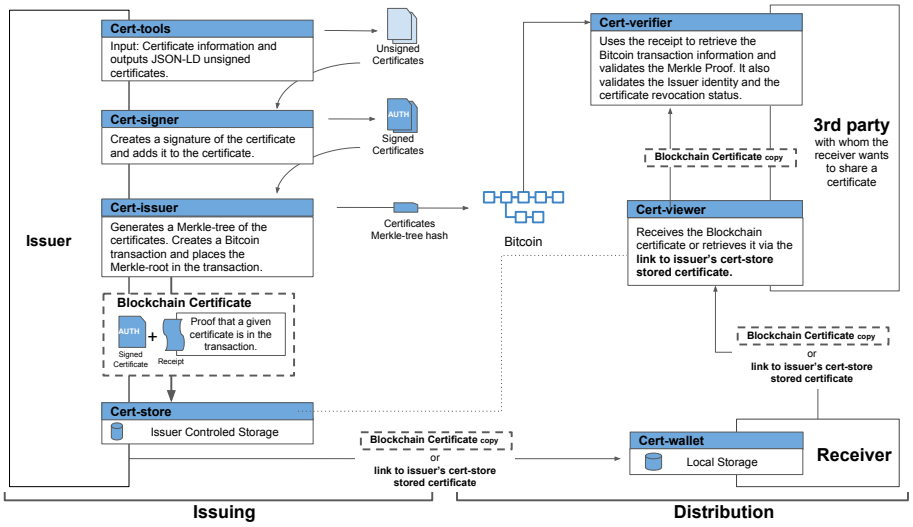

Fig. 4. Blockcerts' certificate issuing architecture

The issuing step: The first step of issuing certificates is having the issuer collect the necessary information (about the achievement and receivers) to create Blockcerts compliant[19] certificates. Receivers are identified by a Unique Identifier (UID), which ideally is a public key that the receiver previously sent the issuer, but it can also be an email address. That information is then fed by *Cert-tools*, which produces, per certificate, a JSON-LD file, which is a Blockcerts compliant *unsigned certificate*.

Next, the issuer signs the receivers' UIDs and adds the signatures to the certificates, which are now *signed certificates*, using *Cert-signer*. The next step is to use the *Cert-issuer* to log the certificates on the blockchain. A merkle-tree is created from the batch of certificates and a hash of the merkle root is placed in a Bitcoin transaction's OP_RETURN[20] field (a field that can be filled with an arbitraty value). The benefit of using a merkle-tree is that by only using the merkle-root it is possible to prove that a given certificate was a part of that tree, this is called merkle-proof [21]. What this means is that, since only the merkle-root is added to the blockchain, the cost of issuing one certificate or issuing multiple is the same (which is the cost of a transaction).

After the issuing has been logged in the blockchain, Cert-issuer outputs, for each certificate, a *receipt* which contains the necessary information for anyone to verify the validity of that certificate, using the merkle-proof. A *blockchain certificate* is a signed certificate accompanied by a receipt. Finally, the Blockchain

[19] https://github.com/blockchain-certificates/cert-schema/

[20] https://en.bitcoin.it/wiki/OP_RETURN

[21] https://github.com/chainpoint/whitepaper/raw/master/chainpoint_white_paper.pdf

certificate is stored by the *Cert-store* which can have several implementations, the most common being a UID lookup store, such as MongoDB.

The distribution step: After a certificate has been issued, the issuer sends it to the receiver. The specification does not go into detail as to how this should be made. For instance, the issuer can send the certificate via email or just supply a link. Receivers can store their certificates locally in the *Cert-wallet*. From that point they can share their certificate with anyone they choose to (*third party*).

For sharing a certificate, just like issuers, receivers can either send the certificate file or a link. The third party uses the *cert-Viewer* to retrieve the certificate from a link (if that is the case) and then to convert the certificate (a JSON-LD file) into a human readable format. Guidelines about displaying a certificate are available at Blockcerts' website[22]. Cert-viewer is responsible for calling *Cert-verifier* in order to verify the validity of the certificate. The certificate is checked for authenticity (by verifying the key pair used to sign the certificate was active at the time of the issuing), followed by authenticity (by performing a merkle proof check using the receipt information) and expiration date (ensuring the current date is not beyond the certificate's expiration date, if that field is filled).

Finally, the certificate's revocation status is verified. Blockcerts follows Open-Badges' specification for certificates revocation method. As of now, the way this is done is by having the *issuer* maintain a list of revoked certificates. An address to that list is provided on each issued certificate so that any party who wants to check the state of the certificate can do so by cross-checking the certificate's ID with the list of revoked certificates.

3.3.2 Blockcerts Weaknesses

Despite all the strengths of Blockcerts with respect to the decentralisation of certification, there are some important weaknesses. An important weakness of Blockcerts regards the revocation mechanism and the fact it requires the maintenance of a list of revoked certificates. In fact, this approach has been discussed in Blockcerts' forum and several problems have been highlighted:

- **The issuer has total control over the certificate:** The certificate can only be revoked by the *issuer* (as he is the one in control of the revoked certificates list) and in the case of there being a need to revoke a batch of certificates, it has to be done one by one.
- **Verification relies on a third party:** Since the list is kept by the issuer, each time a party wants to verify a certificate it will have to communicate with the issuer. This creates a perpetual dependency between each certificate and its issuer which in turn defeats some of the purposes of using a blockchain, namely the capability of having a decentralised perpetual correct ledger. If the issuer ceases to exist there is no longer a way to revoke the certificates it once issued. This issue also ties up **NS: broken sentence!**

[22] `http://www.blockcerts.org/guide/display-guidelines.html`

– **Raises privacy concerns:** Each time the certificate revocation list is consulted on can assume the issuer can know about it (since it is the one hosting the list). This informs the issuer about the way the certificates are being used. Moreover, for each revoked certificate in that list there is a field in which the issuer can clarify the reason for the revocation, since the revoked certificates list is public, anyone can crawl those lists gathering information about revoked certificates.

A second important problem with Blockcerts is that certificates are stored and maintained by the issuer. This is an especially important problem because most of the certificates' distribution is done through the sharing of links (HTTP URLs) which can be easily broken simply by changing the file's location (to another server or another directory). What is needed to solve this is a way to address the certificates with a permanent link that does not depend upon the certificate's location. Hypercerts solves this issue by using IPFS to address certificates.

4 The Hypercerts System

This section presents Hypercerts, our proposed solution for a non-siloed certificate service, i.e., one that does not depend on any single fully trusted entity for maintaining certificate data while supporting the typical certificate service operations, including revocation. First, we present an overview of our system (Section 4.1). Then we describe in more detail the key functionality to be offered to end-users (Section 4.2), and the technical insights to remove trust bottlenecks regarding certificate revocation (Section 4.3) and storage (Section 4.4).

4.1 System Overview

Figure 5 depicts the architecture of Hypercerts. Hypercerts is built upon the Blockcerts system and comprises a few additional software modules on a number of components which aim to overcome important limitations of Blockcerts. (The role of each of these modules is explained in Sections 4.3 and 4.4.)

To design a distributed, non-siloed system that does not rely on third parties to maintain and verify certificates, in Hypercerts, we need to overcome two problems: (1) eliminate the dependency on a certificate issuer in order to check the revocation status of certificates, and (2) remove the dependency on an issuer to store issued certificates. To address (1) we propose to leverage Ethereum's smart contracts to handle certificate revocations. This would allow us to add a lot of functionality and flexibility to the system. To address (2) we propose a new method to store certificates that relies on the IPFS distributed file system by referencing each certificate with a permanent immutable link.

It is important to mention that this would not automatically solve the issue of dependence on a specific blockchain. That is outside of the scope of this work. There is a proposal to how that should be addressed and it points towards there

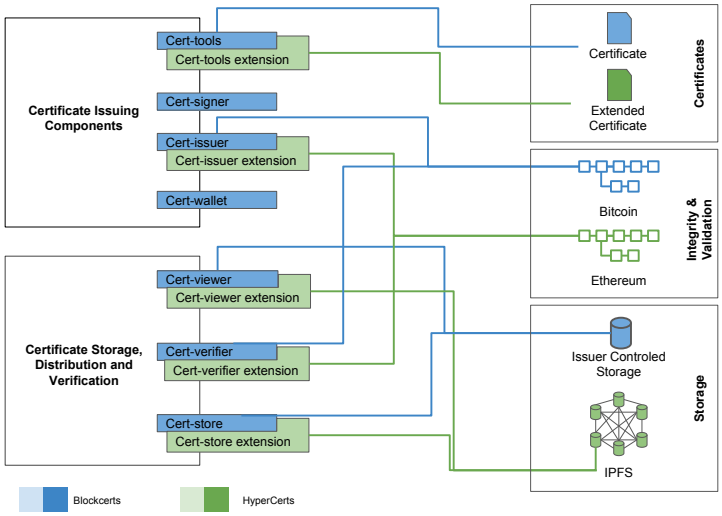

Fig. 5. Components of Hypercerts: native Blockcerts components are depicted in blue; in green, the components to be implemented from scratch.

being an implementation per blockchain (one for Bitcoin, one for Ethereum) and then having Blockcerts interface with them. So even though this does not make Blockcerts blockchain agnostic, it contributes to the effort of having an implementation per blockchain.

Hypercerts is designed under several assumptions. First, we assume the Bitcoin network will continue to work and are not going to model the system to resist a Bitcoin blockchain failure. Second, we assume that there will always be one copy of each certificate in the network. The model Blockcerts uses relies on an issuer having a copy of each certificates it issues and that constitutes a single point of failure. By implementing the certificate storage using IPFS we can ensure that the immutable permanent IPFS link linking to an object will always work as long as there is at least one node in the network with the copy of that certificate. However, the action of storing a certificate is voluntary, we do not implement any rules to ensure that there will always be a node with a copy of a certain certificate. That is requiring for someone to commit to storing a piece of data should have a price associated with it. A way to implement that would be to use a blockchain with a proof-of-space mechanism, such as FileCoin [28].

Currently, Blockcerts does not have an infrastructure in place for receivers to share public keys with issuers, which is a desirable feature to enable proofs of strong ownership (a way for a receiver to unequivocally claim ownership over a certificate). This is an area that is being actively investigated by the community and is outside the scope of Hypercerts.

Regarding privacy, the only publicly available information on the Bitcoin blockchain is the hash of issued certificates. This is a way to guarantee a rea-

sonable level of privacy, since for a party to verify the validity of a certificate is if the certificate receiver discloses both the certificate and the Bitcoin transaction ID where the certificate was Issued to the verifying party. We assume that both the issuers and the verifying parties are well-behaved and will not disclose information that was provided to them by the receivers.

4.2 Functionality Provided to End-Users

Our system will implement several operations, in respect to Blockerts, some are new an improvement of existing ones and others are new. Hypercerts will allow for several revocation mechanisms in order for the system to be as flexible as possible. It will also allow for issuers and receivers to make their certificates available on IPFS.

- **Revocation by the issuer:** The issuer of the certificate should be allowed to revoke a certificate at any given point if it sees a fitting reason to do so.
- **Revocation by the Receiver:** A new functionality, not supported by revoked certificates list, which gives the receiver full control over the certificates it may hold. It is legitimate to imagine a scenario where a receiver no longer wants to be associated with a given issuer.
- **Revocation by a third-party:** Another new feature which will allow a third-party, authorised by the issuer and receiver, to revoke a certificate. This can also be applied to cases where the validity of a given certificate depends on the validity of another certificate.
- **Revocation by a combination of issuer, receiver and third-party:** It will be possible for any combination of the three revocation methods described above to be used.
- **Temporary revocation:** A unique feature of a smart-contract based approach to this is that the revocation status can be changed over time. This will allow for temporary revocation status that can be triggered by a programmable action (for instance, a certificate can be revoked if a receiver fails to pay a pre-accorded monthly fee).
- **Batch revocation:** Some certificates can be issued in batches (as described in 3.3, it makes more financial sense to issue certificates in a batch whenever possible) and that can be troublesome for revocation since having to individually revoke each certificate in a batch would require a lot of operations, which would make it an expensive process (see 3.2.4 that explains the cost associated with computation in Ethereum). For that reason it is important to ensure that the cost of revoking a batch of certificates is either the same or close to the price of revoking just one certificate.
- **Store certificate on IPFS:** Issuers will have the ability to directly store the certificate on IPFS. They can then use the IPFS link to share the file with the recipients.
- **Retrieve certificates from IPFS:** Receivers will be able to retrieve their certificates directly from IPFS.

4.3 Certificate Revocation in Hypercerts

A naive approach to address certificate revocation problem of Blockcerts is to leverage Bitcoin's unspent transaction outputs (UTXOs). By assigning and UTXO to a certificate the way to revoke said certificate would be to perform a transaction that utilised that UTXO, which would render the unspent transaction output, spent. This approach solves all the problems that come with the Revoked Certificates List. The certificate can now be revoked by more than one party (by implementing a Bitcoin smart-contract with multi-signature), the verification no longer requires a third party, as it can be made entirely relying on the Bitcoin blockchain and privacy concerns are no longer an issue (certificate revocation reasons are still public but are recorded i individual files, rather than being all together on a list).

Even though this mechanism would solve the problems with the existing system, it also adds problems of its own:

- **Provides little functionality:** Once revoked the certificate can no longer the valid. A desirable feature on a system such as this one would be the ability to freeze certificates, that same way bank accounts can temporarily be frozen.
- **It is Bitcoin specific:** Even though Blockcerts' current implementation is made on top of the Bitcoin blockchain, the long-term goal is for it to be blockchain agnostic. For that reason, creating a dependency on the Bitcoin blockchain is not a desirable feature.
- **The cost of revoking batches if very high:** It is legitimate to assume that sometimes there will be the need to revoke a batch of certificates. With this approach this would require one transaction per revoked certificate. Since each transaction has a cost associated with it, revoking a whole batch of certificates would make the revoking cost grow linearly with the number of certificates in the batch.

To overcome this problem, Hypercerts uses Ethereum's smart-contracts, which are Turing complete and have support for familiar programming languages thus can be programmed almost like normal applications. The revocation status of a given certificate can be maintained as a variable in a smart-contract. Certificates revocation status can be accessed individually or as a batch. A simple architecture to support this would be to have a smart-contract per batch of certificates. Implementing an Ethereum smart-contracts solution for certificate revocation will require changes to some of Blockcerts' components (refer to Figure 4):

- **Blockcerts Schema:** The Blockcerts schema defines the fields and schema a certificate must have in order to be Blockcerts compliant. Currently one of the required fields is the Revocation List, which contains either an HTTP URI pointing towards the issuer's revoked certificates list or an embedded array of revoked certificates. This schema will be extended in order to contain a URI pointing towards that specific certificate's status, that URI can point to an Ethereum smart-contract, to a Bitcoin UTXO, depending on the

implementation (this because even though we will be procuring an Ethereum implementation, the goal is to make Blockcerts certificate revocation function with any blockchain that can support the functionality).
- **Cert-Tools:** This component receives inputs (data relevant to the issuing of the certificate) from the issuer and outputs a Blockcerts-compliant certificate. It will be required to extend this component to comply with the schema changes proposed above.
- **Cert-Verifier:** The Cert-verifier is responsible for checking the integrity, authenticity and revocation status of a given certificate. This component will also be extended to account for the new revocation status verification mechanism. In this case it will check the revocation status in the Ethereum smart-contract.

4.4 Certificate Storage in Hypercerts

IPFS offers an efficient distributed storage solution for certificates in Hypercerts. Contents are addressed by unique links that are permanent and immutable. Blockcerts provides a reference mechanism for issuers to store their issued certificates, Cert-store, which is implemented in MongoDB. They do make clear, however, that this implementation is not part of the standard and that any issuer can have different approaches to how they store their certificates. The only required aspect is that the certificates can be retrieved via a URL, and IPFS complies with that. In order to accomplish certificate storage on IPFS we propose to change the Cert-store and the Cert-viewer of Blockcerts.

- **Cert-Store:** This is the application that issuers can use to store their issued certificates. We want to implement this application in a way that stores certificates on IPFS.
- **Cert-viewer:** This application is used to retrieve certificates when provided with a URL. We need to extend this application to support IPFS' URLs.

5 Evaluation

The value of this work is directly related to the importance it will have as an improvement Blockcerts' current system. To that end we intend to evaluate the system in three components:

Revocation cost: There is an associated cost with every transaction and it is important to optimise the system in order to make the system as cheap to operate as possible. To that end, we are going to create a private Ethereum blockchain and review the transaction outputs, which will give us information about the cost of operation.

Revocation performance: Blockchains are known for its low throughput and Ethereum is no exception. While revoking individual certificates should not be an issue, revoking batches might by. To that end we will deploy our system in the Ethereum blockchain and test the performance.

Functionality: The goal of this project is to have this system being used by as many people as possible. For that to occur we must assure that the functionality we suggested are well implemented and correct. An important part of the system's evaluation is going to be testing the applications with each other to ensure that we have a cohesive ecosystem that works seamlessly.

6 Planning

We propose the following planning to pursue this work:

- **September 15 – December 5:**
  - Detailed design specifications, and implementation of the system.
  - Submit the software to Blockcerts' Github repository as a pull request.
  - Receive community feedback and improve the solution.
- **December 5 – January 5:**
  - Perform system testing and evaluation.
  - Start writing a technical paper describing the solution.
- **January 5 – February 20:**
  - Conclude technical paper.
  - Finalise the writing the dissertation.
- **February 20:**
  - Deliver the MSc dissertation.

7 Conclusions

Current certification mechanisms fall short of a lot of desirable features such as interoperability and self-sufficiency. Although there have been steps in the right direction there are still problems to solve. Blockcerts, which follows the OpenBadges specification, is the system that currently offers the best set of desirable properties, due to its use of the blockchain for certificate verification purposes, but still fails at completely removing dependency on issuer hosted data. We present Hypercerts that leverages Ethereum's smart contracts and IPFS to solve the aforementioned problems.

References

1. Ch, S.K., Popuri, S.: Impact of online education: A study on online learning platforms and edX. Innovation and Technology in Education MITE, IEEE International Conference in MOOC (2013)
2. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining gamification. ACM (2011)
3. Hamari, J., Koivisto, J., Sarsa, H.: Does Gamification Work? – A Literature Review of Empirical Studies on Gamification. In: 2014 47th Hawaii International Conference on System Sciences (HICSS). (2013)

4. Ibanez, M.B., Di-Serio, A., Delgado-Kloos, C.: Gamification for Engaging Computer Science Students in Learning Activities: A Case Study. IEEE Transactions on Learning Technologies (2014)
5. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. self-published paper (2009)
6. Christidis, K., Devetsikiotis, M.: Blockchains and Smart Contracts for the Internet of Things. IEEE Access (2016)
7. Miller, A., 0005, Y.X., Croman, K., Shi, E., Song, D.: The Honey Badger of BFT Protocols. ACM Conference on Computer and Communications Security (2016)
8. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In: 2015 IEEE Symposium on Security and Privacy (SP). (2015)
9. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to Better - How to Make Bitcoin a Better Currency. Financial Cryptography (2012)
10. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the Security and Performance of Proof of Work Blockchains. In: the 2016 ACM SIGSAC Conference. (2016)
11. Danezis, G., Meiklejohn, S.: Centrally Banked Cryptocurrencies. In: Network and Distributed System Security Symposium. (2015)
12. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of Space. In: Advances in Cryptology – CRYPTO 2015. (2015)
13. Miller, A., Juels, A., Shi, E., Parno, B., Katz, J.: Permacoin: Repurposing Bitcoin Work for Data Preservation. In: 2014 IEEE Symposium on Security and Privacy (SP). (2014)
14. Ali, M., Nelson, J., Shea, R., Freedman, M.J.: Blockstack: A global naming and storage system secured by blockchains. 2016 USENIX Annual Technical (2016)
15. King, S., Nadal, S.: Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper (2012)
16. Wustrow, E., VanderSloot, B.: DDoSCoin: cryptocurrency with a malicious proof-of-work. of the 10th USENIX Conference on (2016)
17. Ghosh, M., Richardson, M., Ford, B., Jansen, R.: A TorPath to TorCoin: Proof-of-Bandwidth Altcoins for Compensating Relays. Yale University (2014)
18. Milutinovic, M., He, W., Wu, H., Kanwal, M.: Proof of Luck. In: the 1st Workshop. (2016)
19. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A Secure Sharding Protocol For Open Blockchains. ACM Conference on Computer and Communications Security (2016)
20. Eyal, I., Gencer, A.E., Sirer, E.G., van Renesse, R.: Bitcoin-NG - A Scalable Blockchain Protocol. NSDI (2016)
21. Bowe, S., et al.: Zcash Protocol Specification. self-published paper (2017)
22. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In: 2016 IEEE Symposium on Security and Privacy (SP). (2016)
23. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In: 2015 IEEE Symposium on Security and Privacy (SP). (2015)
24. Wood, G.: Ethereum Yellow Paper. self-published paper (2014)
25. Buterin, V.: Ethereum white paper. self-published paper (2013)
26. Benet, J.: IPFS-content addressed, versioned, P2P file system. arXiv.org (2014)
27. Merkle, R.C.: A Digital Signature Based on a Conventional Encryption Function. In: Advances in Cryptology — CRYPTO '87. (1987)

28. FileCoin.io: Filecoin: A Cryptocurrency Operated File Storage Network. self-published paper (2014)