Revoking Records in an Immutable Ledger

A platform for issuing and revoking official documents on public blockchains

Konstantinos Karasavvas
Blockchain Initiative
University of Nicosia
Emails: karasavvas.k@unic.ac.cy, kkarasavvas@gmail.com

***Abstract*—Bitcoin allows anyone to store small amounts of data on its blockchain as part of a transaction. The data stored become part of the history of Bitcoin's blockchain, which after some time, becomes very tamper resistant, an attribute that many applications find of great use, e.g. document notary systems, asset meta-protocols, etc. However, many use cases require the functionality to revoke (cancel the validity) of the stored data. Previous attempts to handle revocation were not adequate. Since tamper resistance is a default attribute one needs a non-systemic approach to deal with revocation, which usually implies additional infrastructure to accomplish it. This paper reviews several revocation methodologies together with advantages and disadvantages for each one, as well as proposes a simple and clean solution for revocation without any additional infrastructure or third-party dependencies.**

I. Introduction

Bitcoin [1], an open decentralized digital currency, introduced technology that allowed for a new way for distributed systems to reach consensus. That innovation invigorated research in several fields of study like peer-to-peer computing, cryptography, machine-to-machine interactions, consensus algorithms, etc., not to mention the potential social aspects and implications of this technology. The technology that Bitcoin introduced is typically referred to as *blockchain* or *distributed ledger* technology and has the following inherent attributes:

- Decentralized: the valid state is determined by a majority consensus of the networks participants. Anyone can attempt to change the state but, for a change to be successful, the majority needs to agree that the new state is valid per the network rules.
- Immutable[1]: transactions can only be added on the structure; they cannot be altered or removed without majority consensus. When transactions are added, they are also timestamped. Every participant has a complete history of all transactions which is synchronized among the participants. The system is extremely fault-tolerant and the data stored are practically permanent during its history (or for as long as there are some participants on the network).
- Transparent: the process of adding new data entries as well as the data themselves are publicly available and anyone can review or access at any point of time.
- Open: the network is open for anyone to participate, i.e. no barriers of entry.
- Secure: strong cryptography ensures that a participant cannot claim funds belonging to other participants.

The above attributes offer the potential for many interesting applications in addition to digital currency. In particular, the Bitcoin protocol allows for including metadata in each transaction that is stored, which allows external applications to store data in an open, transparent and immutable structure that can be accessed at any time. Some applications store metadata that correspond to assets and their ownership [2] [3] [4], while others allow to prove the existence of a document [5] [6] [7].

The proper way[2] to store metadata is by including the `OP_RETURN` [8] [9] instruction as part of the transaction. The maximum amount of data per transaction is 80 bytes. Several transactions can be used to store more than 80 bytes but each transaction incurs a fee. Most applications include a prefix identifier before the actual metadata to make it easier to spot and gather the transactions, e.g. "CNTRPRTY" for Counterparty [3] or "DOCPROOF" for Proof-of-Existence [6].

Sometimes applications require the metadata to be removed or modified but that is not allowed by the very nature of the blockchain. Thus, most applications either not allow revocation of records (most notary services like [6]) or allow revocation with one or more of the following caveats:

1) Centralization: revocation can occur, but there is a central point of failure
2) Complex infrastructure: revocation can occur because there is a second decentralization network on top of Bitcoin's
3) Permissioned: registration and service fees are required

In this paper we will present a method for achieving revocation with a practical approach, while not diverging from the open and decentralized nature of Bitcoin. In the next section, we will describe an application which will be used to apply our revocation approach. Then, in section III, we will examine different ways of revocation used in other systems and elaborate on the benefits and drawbacks of each. Section IV will then describe

[1]Tamper resistant is more precise, but we will use *immutable* as most of the literature uses it.

[2]There are other ways to store data in Bitcoin's blockchain (like using fake addresses in the transaction outputs) but most create a runtime overhead to the network and thus are frowned upon.

in detail the meta-protocol which allows for several actions, including the revocation of the metadata. Finally, in the last sections, we will discuss the solution further and conclude.

II. Example Application

To demonstrate the application of our revocation methodology we will describe our open platform[3] that makes use of it. Our platform allows institutions to issue digital credentials representing certificates, diplomas, driving licenses, etc. The credentials format is a PDF file which was chosen because it is a widely-adopted, portable, human-readable format where viewing software is already available for any platform. It is an intuitive medium that everybody is comfortable using.

The platform issues credentials on Bitcoin's blockchain since it is the oldest and most tested public blockchain. It has the most security in terms of hashrate and community philosophy; arguable the most tamper resistant blockchain. It is already used by most systems that require to store metadata. Note, however, that the platform itself has no real dependency on Bitcoin and it could use another blockchain if required.

The platform's desirable characteristics are:

1) The digital credential, represented as a PDF, should to be self-contained; include everything it needs to be validated independently
2) A hash of the credential should be stored in the blockchain[4]; the blockchain properties described in the introduction ensure that the credentials will be impossible to falsify
3) A single issuance should represent an arbitrary number of credentials
4) It should be permissionless so that anyone can use (no barriers of entry, no registration, open source software)
5) Issuance and validation of credentials should be easy
6) Validation should have no dependency to the issuing institution; the only dependency is the blockchain
7) Revoking credentials should be possible in a decentralized way
8) Identity of issuing institution should be established in a decentralized way

The platform already satisfies characteristics 1-6 and in this section we will describe how it is achieved. The next two sections will describe how we incorporated revocation to the platform while adhering to the decentralized attributes previously discussed.

A. Issuing Credentials

Assuming we have a set of credentials and some metadata that we want to include for each credential, our platform makes it trivial to issue them on the blockchain. The process behind-the-scenes can be briefly described as follows:

[3]https://github.com/UniversityOfNicosia/blockchain-certificates

[4]We are using Bitcoin's blockchain since it is for its security—highest hashrate— its resilience and security through it longevity

1) Adding credentials metadata: In each PDF credential we attach some metadata[5] in machine-readable format. This metadata contain the issuer institution and the Bitcoin address which will be used to publish the certificates on the blockchain. Optionally, additional metadata, like awardee's full name, etc. can be added to provide more useful information on validation. Note, that the metadata are attached to the credential but are not visible and can be retrieved only programmatically.

Fig. 1. Adding credentials metadata

2) Calculating merkle root: The credentials are then hashed together using a merkle tree data structure to produce a merkle root that can potentially represent tens of thousand or more credentials. In essence a merkle root represents all the leaf-hashes (i.e. the credentials) and if a single bit is changed in any certificate the merkle root would be completely different.

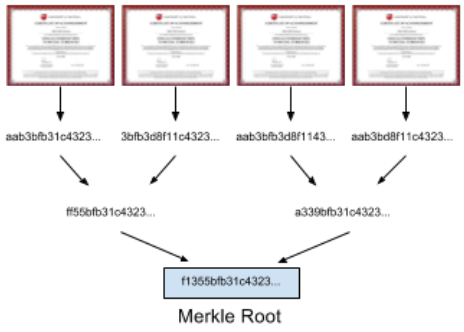

Fig. 2. Calculating merkle root

3) Issuing on the blockchain: We then create a Bitcoin transaction that contains the special OP_RETURN operator followed by a prefix identifier for the institution followed by the merkle root from the previous step. We send the transaction to a Bitcoin node (preferably a local node or one that we fully trust) and we get the transaction identifier (TxID) back. After we wait for a few confirmations we can be certain the transaction (and thus merkle root) is now safely stored in the blockchain.

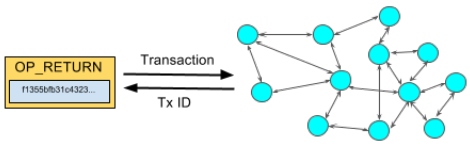

Fig. 3. Issuing on the blockchain

[5]The PDF standard allows for custom metadata [10].

4) Adding blockchain proof metadata: After we issue the merkle root on the blockchain we need to put some extra information into each credential's metadata that proves that it was issued in the blockchain. The blockchain proof contains the merkle proof (all the tree children required to calculate the merkle root starting from the hash of the certificate), the merkle root and the TxID that we got in the previous step. We use the chainpoint standard [11] for the blockchain proof. The additional metadata are added in a deterministic manner, in such a way that removing it will rollback the credential as it was after step 1.

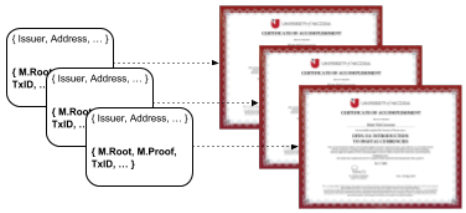

Fig. 4. Adding blockchain proof metadata

The credentials are now ready and contain all that is needed to be independently verified by anyone using the blockchain. They are disseminated to their owners.

B. Validating Credentials

The platform also provides tools with user-friendly interfaces where one can validate a PDF issued by the platform. There are already some independent[6] validators running. A credential holder can send their credentials to a consumer and the latter will have everything needed to verify it; they can use any of the independent validators out there or even get the source code and install a validator themselves. The validating process behind-the-scenes is as follows:

1) Extract blockchain proof metadata: The blockchain proof metadata are extracted and removed from the PDF. The credential is now the same as it was after Step 1 of issuing and it was this hash that was issued on the blockchain. The extracted metadata will be used later on.

Fig. 5. Extract blockchain proof metadata

2) Calculate hash of the remaining certificate: We hash the PDF credential, which as mentioned above, is the one that was issued on the blockchain. If the certificate was modified in any way the hash will differ from the one issued.

3) Get the merkle root stored in the blockchain: We use the TxID from the extracted metadata to get the transaction that contains the OP_RETURN with the merkle root of all the credentials.

[6]Not from the issuing institution.

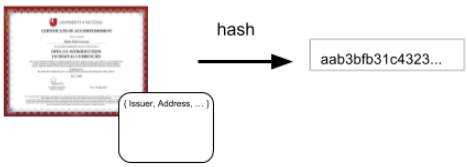

Fig. 6. Calculate hash of the remaining certificate

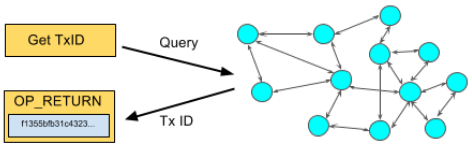

Fig. 7. Get the merkle root stored in the blockchain

4) Calculate merkle root from proof and compare: We then need to validate that the credential's hash is indeed part of the merkle root that we got from the blockchain. We do this by using the credential's hash (Step 1) and using the merkle proof from the extracted metadata to reconstruct the merkle root. If it is the same as the one we got from the OP_RETURN, then we can be certain that this credential is unmodified and the one issued by the institution.

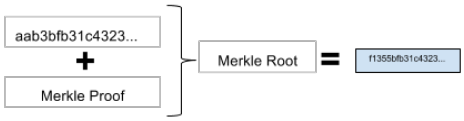

Fig. 8. Calculate merkle root from proof and compare

The authenticity was thus proven[7]. The credential is now part of the Bitcoin's history and no one can alter it or delete it. But what if we wanted to revoke the certificate because it is no longer valid or the name was mistyped, etc.? In the following section we will examine different ways of revocation in blockchain.

III. Revocation

For applications like the one described above revocation is an important aspect of the system, although it might not be used frequently. As is, the platform can only issue a new credential to fix typographical errors but both the new and old credential would be valid. More importantly, it can not revoke erroneous credentials. In this section we will explore potential solutions to revocation.

A. Centralized revocation

This approach is straight-forward. The credential itself contains a URL endpoint that is used to display and/or validate the credential. The URL points to the issuer's[8] website where an internal database is consulted regarding validity of the credential, etc. This approach is being used by Gradbase [12] and revocation is trivial to manage but it is centralized.

[7]But see section V for more on identity issues.

[8]Or to a service provider that manages issuing for clients.

We strongly believe that having a centralized validator defeats the purpose of using Bitcoin's blockchain[9] in the first place.

B. Re-issue all credentials

This approach re-issues all credentials every time a new batch of credentials needs to be issued. The new batch could contain corrections to previously issued credentials or even not include a past certificate, thus revoking it. Since a single transaction can be used for tens or hundreds of thousands credentials, it does not make a difference on the issuing itself. For this to work, the validator will need to take the last issued transaction and check that the PDF's hash is in the merkle root as usual. Note that the transaction in the blockchain proof from the PDF metadata is not used in this case.

The benefits of this approach is that it is very easy to implement; we just need to slightly change the credential valitation rules. However, there are several drawbacks:

- It requires very good management of credentials. This is important for all approaches but re-issuing everything every time introduces the possibility of making errors with past credentials as well, which is not the case with other approaches
- Bitcoin's timestamp is meaningless since after re-issuing the previous certificates *lose* their timestamp
- If the private key of the issuing address is compromised, all certificates (past and future) would be invalidated
- We *break* chainpoint semantics; chainpoint tries to become a standard on how to put and validate data into the blockchain
- Most importantly, all certificates need to be disseminated again to all the awardees; this is infeasible in practice

C. Extra transaction outputs per credential

This approach was used by the first version of Blockcerts [13] [14], a platform to publish academic certificates on the blockchain. The process is similar to the one described in the previous section but the certificate is just a JSON file with no other representation. All users/consumers need specific software to visualize the JSON file and see the certificate.

The transaction that issues the OP_RETURN with the merkle root includes several extra outputs sending some satoshis to each. Specifically, for each credential two extra outputs are created one owned by the institution[10] and one by the awardee. Upon issuing, 2750 satoshis[11] are sent to each of the extra addresses. The validator rules state that the hash has to be valid and that both of the extra addresses require to have funds. If either of the amounts has been spent (by the institution or by the awardee) then the Blockcerts certificate is considered invalid.

This approach has the benefit that there is a lot of granularity and flexibility on revocation, especially that both the issuer and the awardee can revoke the certificate[12]. One can think of simple variations, like having only one extra address allowing only the institution to revoke, etc. but overall, this approach also has significant drawbacks:

- Makes issuing expensive; for 1,000 certificates there will be 2,000 extra outputs with 2,750 satoshis each which means 0.055 bitcoins. Moreover, a transaction with that many outputs will incur significant fees
- That approach is limited to the number of certificates that can be issued since there is a limit of 100k bytes per transaction
- Revocation is price-dependent
- Depending on Bitcoin's price the amount of satoshis could be considered *dust*[13] and not be easy to spend
- Issuers require infrastructure to keep track of all the extra information per certificate: the awardees' addresses and the institution's addresses and private keys for all certificates ever issued. It also has the overhead that each user/graduate to provide an address they control to the issuer.
- Even awardees with more than one certificate need to remember which address corresponds to which certificate
- The Unspent Transaction Output set (UTXO) is unnecessarily bloated increasing the runtime requirements of nodes

D. Revocation is handled by an issuer-hosted CRL

This approach is used by the second version of Blockcerts [14]. Effectively, the JSON certificate contains a URL which resolves to a file that contains a certificate revocation list (CRL); effectively all the revoked certificates of that issuer.

This introduces a central point of failure and has similar disadvantages with the centralized revocation approach that we described above.

E. Use an additional decentralization layer for revocation

Another approach is to make use of another blockchain like Factom [7], Alexandria [15], Lbry [16] or an Ethereum [17] smart contract to store the revocation information. Entries on these blockchains would represent a credential in some way and it would allow revocation at that level; pointing at the appropriate transaction at the Bitcoin network.

This approach is promising but it does add an additional dependency (blockchain layer), which introduces unwarranted complexity both design-wise and implementation-wise. It is worth noting however, that if the additional overhead is accepted, this approach could offer a complete alternative to how our example application would issue and validate credentials, i.e. it could handle more than the revocation part.

[9] Or any open blockchain.

[10] Sent to an address which private key is controlled by the institution.

[11] At the time that was cheap enough without the output being considered dust, i.e. the transaction fee cost of spending the amount is close to its value.

[12] Arguably, the only reason for the awardee to revoke a certificate is because a fake issuer created their certificate. But if that is the case they would not be able to revoke anyway since the fake institution would not have used an address that the awardee controls!

[13] The transaction fee cost of spending the amount is close to its value.

F. Blockchain specializing in credential issuing

Similarly, and for completeness, one other approach would be to design a specialized blockchain platform to deal with credentials; their issuance, dissemination, validation, revocation, etc. Private initiatives have been announced already but there are no real results yet. From our perspective, a private blockchain platform suffers from all the caveats mentioned in the introduction. On the other hand, a public blockchain that incentivizes institutions to use it would be of particular interest but we are not aware of any such attempts.

It is worth noting that there are attempts to design blockchains that allow modifying past transactions [18] but in practice they do introduce centralization points invalidating several of the advantages that blockchain offers.

G. Credentialing meta-protocol

The final approach and the one that we will describe in great detail in the next section is to encode meta-protocol information in the 80 bytes that we have available in the OP_RETURN. The meta-protocol will contain operations specific for credentials and will refer to previous transactions as appropriate. The validator rules will be such that other transactions could be consulted before the final resolution for credential validity.

This will allow to implement revocation as well as additional functionality. Some of the encoded operations are: issuing credentials, revoking a credential batch or specific credentials and even revoke issuing addresses. This approach has all the required attributes, i.e. it is a simple and intuitive decentralized solution that is open and has no other dependency than a single blockchain. The only disadvantage of this approach is that because we have a limit of 80 bytes we can only revoke two credentials at a time, which means a different transaction per two revocations. However, since revocations are not very frequent, it is a minor inconvenience.

IV. BDIP META-PROTOCOL

This section describes a meta-protocol, named *Blockchain Document Issuing Protocol (BDIP)*, that specifies how to issue, revoke and validate credentials on a blockchain. The protocol defines the structure of the messages as well as their semantic meaning. The messages are limited to the 80 bytes that we can store in an OP_RETURN. Each operation is a separate transaction which consists of an input from a Bitcoin address that the institution controls[14], the OP_RETURN output and, optionally, a change output.

A. Message Operations and Structure

Every message consists of a header and the payload. We will represent messages in their hexadecimal value, i.e. maximum of 160 digits. The header is 8 bytes and has three parts:

1) An identifier for the protocol; the ASCII of 'BDIP', which is 4 bytes and corresponds to '42444950' in hex

[14]And is used for the transaction fees.

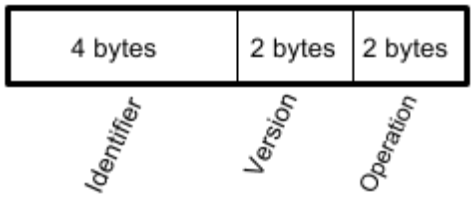

Fig. 9. Header — structure / bytemap

2) Protocol version; 2 bytes for a total of 65536 versions, starting with '0001' in hex
3) Protocol operation; 2 bytes for a total of 65536 operations

Every operation starts with the above header being differentiated only by the last 2 bytes. Currently, BDIP version 1, defines five operations.

1) Issue operation: Issues a batch of credentials. The operation code is '0004'. The issuer identifier is there just

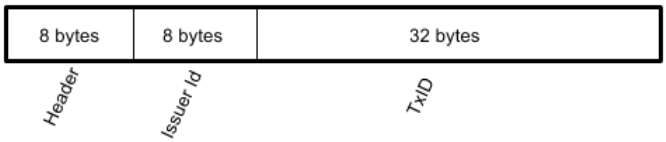

Fig. 10. Issue credentials — structure / bytemap

to allow for an institution to *brand* the issuing making it easy to identify when looking at an OP_RETURN message. It is the equivalent of the prefix that was described in our example application in section II. It is optional and does not play a role in validation.

2) Issue with expiry operation: Issues a batch of credentials that will expire some time in the future. The operation code is '0006'. This is similar to the previous operation with the ad-

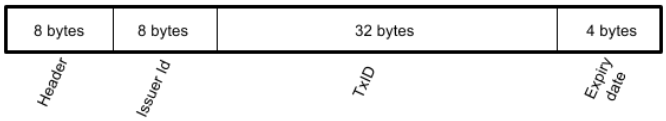

Fig. 11. Issue credentials with expiry data — structure / bytemap

dition of an expiry date. The expiry date could be represented either as Unix time[15] or a future block height. If time is used it will be resolved as a Median Time-Past (MTP) [19]. During validation expiry is checked before anything else.

3) Revoke batch operation: Revokes a whole batch of credentials. The operation code is '0008'. This simple operation

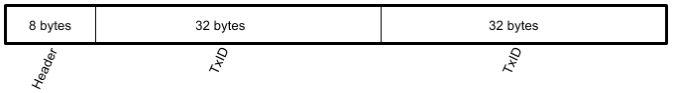

Fig. 12. Revoke a batch of credentials — structure / bytemap

consists of just a transaction id (or two) of the issuing batch

[15]https://en.wikipedia.org/wiki/Unix_time

that we want to revoke. The transaction needs to be from a previous issuing operation that has been send from the same address as the revoke operation. During validation revoke operations are taken into consideration.

4) Revoke credentials operation: Revokes one or two specific credentials. The operation code is '000c'. A single

Fig. 13. Revoke specific credentials — structure / bytemap

transaction id is specified from an existing issuing operation; both transactions (issuing and the revocation) have to originate from the same Bitcoin address. Then after the transaction id, one or two file hashes can be used for revocation. These are the PDF credential hashes as produced if we hash the file after step one of the issuing process (see subsection II-A). To save space we apply RIPEMD-160 hash function on the hashes to reduce them to 20 bytes each fitting them neatly in the message. During validation the credential's hash is hashed again with RIPEMD-160 and compared to each of the hashes in revoke operations.

5) Revoke address operation: Revokes an issuing address. The operation code is 'ff00'. The address is specified in the

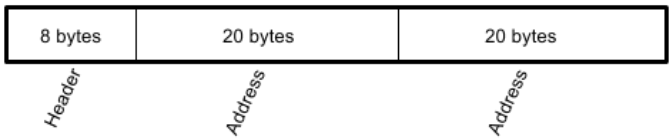

Fig. 14. Revoke address — structure / bytemap

public key hash format and not as an address[16]. Two addresses can be specified for the compressed and uncompressed version of the public key.

After a revoke address operation has been issued the current Bitcoin address cannot be used again to issue credentials or for anything else. The operations until the revocation are still valid but nothing after it. This can be useful when the private key of the institution has been compromised and we want to prohibit others issuing on the institution's behalf.

B. Validation Rules

When a credential is presented for validation, we have to follow specific rules to capture the semantics defined in the previous section. Several steps need to be followed and for version 1 of the protocol these are:

1) Validate the credential as described in section II-B; and ensure that the issuing operation has at least one confirmation
2) If the operation was issued with expiry, check that it has not expired first.
    a) If the expiry is by block height compare with the current block height and fail validation if the current block is higher than the one in the issuing transaction
    b) If the expiry is by Unix time calculate the MTP, compare them and fail validation if the MTP is higher than the expiry operation in the issuing transaction
3) Get all transactions for the issuing address before the issuance of the credential in question. If a revoke address operation contains the issuing address, the validation fails; i.e. the address was revoked, so no new credentials should be issued
4) Get all transactions for the issuing address after the issuance of the credential in question. Check all revoke operations until the last transaction or until a revoke address with that issuing address is found
    a) If a revoke batch operation exists check whether it revokes the issuing transaction and if yes, fail validation
    b) If a revoke credentials operation exists check whether it revokes the RIPEMD-160 of the credential's hash and, if yes, fail validation

Of course, the implementation is much more optimized than the above description. For example, a single call[17] can be used to retrieve all transactions related to an address and the rest of the checks/calculations are trivial.

V. Discussion and Future Work

The system and protocol described in this paper is a working system already in production use for several years publishing certifications on the blockchain since 2014. From 2017 all diplomas issued by the University of Nicosia are issued on Bitcoin's blockchain and other universities (e.g. The British University in Dubai) already make use of it. The revocation functionality was added a year ago but it can be used to revoke past qualifications as well.

Some operations, like *issue* and *issue with expiry*, could be one operation but given the huge space of potential operations we opted to have simple operations for more straight-forward implementation of the validation rules.

It is important to emphasize, that in contrast to some of the other revocation approaches, anyone can run a validator to verify credentials. However, it is much easier to use online validators operated by institutions that used this solution[18]. Any credential issued by any institution can be verified to any available validator assuming that the same open source repository is used.

Note that, when issuing, there is an optional institution identifier. If used, it could make the validator produce extra information on certain institutions. For example, a validator from institution A would apply to credentials from all institutions but if it finds credentials that they issued it can consult internal records to provide with even more information for those credentials. Although this is possible, it is not recommended since

[16]This is the same way that addresses are specified in scriptPubKey scripts.

[17]Several services provide such an interface and more than one can be used to avoid a centralized call. With a bit of more extra work a local node can also be consulted instead of several third party services.

[18]At least two official validators existed when this was written: https://verify.unic.ac.cy/verify and http://94.200.87.28:5000/verify

we would not like to encourage differentiation between the validators' functionality—even if that functionality is above the fundamentals. This is because end-users might be incentivized to use the validators with the extra information. Having said that, the system is completely open.

Performance-wise, the only potential overhead could be in the number of transactions created from an issuing address. We expect institutions to use those addresses with care and when necessary but, even if they do not, it will hardly influence the system as a whole. Only the credentials from that institution will take longer than usual[19] so they would only hurt themselves.

We do intend to extend this work in several ways. First, we will enable contacting a local Bitcoin node for validation purposes to eliminate the dependency on third-party APIs—even though we are using several and thus is not exactly centralized even now. More importantly, though, we are working on credentials schemata and an identity solution.

Institutions are already allowed to issue credentials with a wide range of metadata. However, different metadata will put a lot of strain on keeping the validator up-to-date with different metadata. By introducing schemas to define credentials we can formalize what is supported by each validator and even automate user interface creation depending on the specific schema—allowing validators to display information from a credential's metadata without any a-priori knowledge of the structure or content of the metadata.

Finally, a very important aspect for the security of such systems—i.e. it applies for all of the other approaches mentioned—is how to resolve and verify the identity of each institution. This is an unsolved problem although there are several approaches trying to tackle with it. We are exploring our options on identity and are leaning towards a web-of-trust approach but it is still work in progress and outside the scope of this paper.

VI. Conclusion

While other solutions for revocation in blockchains have been suggested they do have one or more caveats; others are centralized, others require complex infrastructure and resources, while others are permissioned services.

We have described our solution for decentralized digital credentials using blockchain technology where an institution can issue self-contained digital credentials which can be validated immediately in a trustless manner with no single point of failure.

While we focused on the domain of digital credentials, the revocation methodology and the meta-protocol solution described can apply to other domains—any kind of official document can be timestamped on the blockchain—that require immutability but still need a way to revoke entries.

Acknowledgment

The authors would like to thank the Blockchain Initiative team of the University of Nicosia and especially George Giaglis and Antonis Polemitis for their feedback throughout design and implementation.

[19]Since it will need to traverse the address's transactions for potential revocations.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] J. R. Willett, M. Hidskes, D. Johnston, R. Gross, and M. Schneider, "Omni protocol specification." [Online]. Available: https://github.com/OmniLayer/spec

[3] "Counterparty specification." [Online]. Available: https://counterparty.io/docs/protocol_specification/

[4] "Open assets protocol." [Online]. Available: https://github.com/OpenAssets/open-assets-protocol

[5] "Stampery website." [Online]. Available: https://stampery.com/

[6] "Proof-of-existence website." [Online]. Available: http://proofofexistence.org/

[7] P. Snow, B. Deery, J. Lu, D. Johnston, and P. Kirby, "Factom white paper." [Online]. Available: https://www.factom.com/devs/docs/guide/factom-white-paper-1-0

[8] "Bitcoin OP_RETURN community wiki." [Online]. Available: https://en.bitcoin.it/wiki/OP_RETURN

[9] M. Bartoletti and L. Pompianu, "An analysis of bitcoin OP_RETURN metadata," *arXiv:1702.01024v2 [cs.CR]*, 2017.

[10] "Pdf standard custom metadata." [Online]. Available: https://helpx.adobe.com/acrobat/using/pdf-properties-metadata.html

[11] W. Vaughan, J. Bukowski, and S. Wilkinson, "Chainpoint paper." [Online]. Available: https://chainpoint.org/

[12] "Gradbase website." [Online]. Available: https://www.gradba.se/en/

[13] "Blockcerts website." [Online]. Available: https://www.blockcerts.org/

[14] "Blockcerts revocation v1 v2." [Online]. Available: https://github.com/blockchain-certificates/cert-schema/issues/24

[15] "Alexandria website." [Online]. Available: http://www.alexandria.io/

[16] "Lbry website." [Online]. Available: https://lbry.io/

[17] "Ethereum white paper." [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[18] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain - or - rewriting history in bitcoin and friends," *IEEE Xplore 10.1109/EuroSP.2017.37*, 2017.

[19] "Median time-past as endpoint for lock-time calculations." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0113.mediawiki