A model for long-term preservation of digital signature validity: TrustChain

Vladimir Bralić, Magdalena Kuleš, Hrvoje Stančić
Department of Information and Communication Sciences
Faculty of Humanities and Social Sciences, University of Zagreb
Ivana Lučića 3, 10000 Zagreb, Croatia
bralic.vladimir@gmail.com, magdalenakules@gmail.com,
hstancic@ffzg.hr

Summary

When archiving a digitally signed document an issue arises once the certificate used in the signature expires (or possibly the certificate authority stops functioning). Once this happens, the signature can no longer be confirmed and tampering with the document is possible. This paper presents a model for long-term preservation of digitally signed documents using blockchain technology. The authors propose a semi-open system in which only certain institutions can create new entries but any interested party can view the records and confirm their authenticity.

Key words: digital signature, blockchain, archive, certificate authority, long-term preservation, TrustChain, TRUSTER

1. Introduction

Digitally signed documents are now seeing widespread use in almost any online, digital document storage, management and preservation systems. They are replacing the traditionally sealed and signed documents. While the digital signature system is now well-documented, safe, and easy to implement, an issue arises with the outdated certificates. For better understanding, it is necessary to define what a digital signature is. According to one definition, digital signature is "a code, generally created using a public key infrastructure (PKI) associated with a digital object that can verify the object has not been altered and, in some contexts, may be used to authenticate the identity of the sender"[1], while the other defines digital signature as "cryptographic transformation of data which, when associated with a data unit, provides the services of origin authentication and data integrity and may support signer non-repudiation"[2]. Digital signatures

[1] A Glossary of Archival and Records Terminology, Society of American Archivists, https://www2.archivists.org/glossary/terms/d/digital-signature (Accessed 30.05.2017).
[2] InterPARES Trust Terminology Database, http://arstweb.clayton.edu/interlex/en/term.php?term=digital%20signature (Accessed 30.05.2017).

have two aspects. Firstly, they guarantee the integrity of a document. This means they guarantee the document contents match that at the time of digital signing. Secondly, they guarantee authenticity[3]. A digital signature can be traced back to a specific person or institution using a certificate authority. In this manner, one can be sure that the document was created by the stated author, and that the document could be used as a legally binding contract. The first aspect is time independent (one can always confirm the document in question has the appropriate signature by recalculating the hash of the document and comparing it to the one in the digital signature. However, the second aspect is time dependent. Most digital certificates expire, and unless renewed by their owners, i.e. document creators, cannot be confirmed. They are also reliant on certificate authority institutions still being in operation. It is certainly conceivable that some certificate authorities might go out of business or close down at some point in the future. Once this happens, it will be impossible to confirm that their certificates are genuine. Currently, most archives depend on trust to confirm outdated digital signatures. One has to trust the archive (or another institution) which preserves the document that the signature was valid at the time of archiving and that the document has not been tampered with. Another common solution is to use a time stamp service, which significantly extends the lifetime of a signature but, much like the digital signature itself, is not a permanent solution. To improve this situation we propose a system based on the blockchain technology[4] that might eliminate the need to trust archiving institutions by storing control hashes of digital signatures in an immutable and publicly readable blockchain. By using such a system, any interested party could confirm that a digitally signed and archived document has indeed remained unchanged and that its signature was valid at the time of the blockchain record creation. We call this system TrustChain. TrustChain is being developed as part of the TRUSTER Preservation Model (EU31) research study at the InterPARES Trust international project and is one of several solutions to the problem of long-term preservation of digitally signed documents being considered by the research group.

2. The TrustChain concept

The system we propose is based on cooperation between multiple archival (or other interested) institutions. While there is no technical reason why a single

[3] ISO 15489 defines that "an authentic record is one that can be proven: a) to be what it purports to be, b) to have been created or sent by the person purported to have created or sent it, and c) to have been created or sent at the time purported", ISO 15489-1 Information and Documentation – Part 1: General, p. 7.

[4] Blockchain – an open-source technology that supports trusted, immutable records of transactions stored in publicly accessible, decentralized, distributed, automated ledgers. InterPARES Trust Terminology Database, http://arstweb.clayton.edu/interlex/index.php (Accessed 20.04.2017).

institution could not run the needed software and hardware components, the trust in the envisioned system is in direct relation to the number of independent participating institutions. If a single institution runs the whole system, that institution is capable of manipulating records and would need to be trusted implicitly. This is the situation we have today. We are bypassing this need to trust a single institution by requiring multiple institutions to confirm the validity of a digitally signed document before writing it into an immutable blockchain. In principle, our approach uses a blockchain as described in the Bitcoin Whitepaper (Nakamoto, 2008) but we do not include the proof-of-work concept so our solution is perhaps more similar to the original Haber and Stornetta timestamp linking and random-witness solutions (Haber and Stornetta, 1991). We have merged both approaches and designed the system for use specifically for preserving (timestamping) digital signatures by a trusted union of (archival) institutions.

The core of the system is a blockchain containing hashes of digital signatures. Any interested individual or institution can request a record to be added to the blockchain but only the authorised nodes are allowed to write the new record into the blockchain (after confirming validity of digital signature(s)).

TrustChain nodes are servers maintained by institutions participating in the TrustChain project. These servers accept new record requests, process them, write them into the chain and keep the blockchain stored and available to be read by interested parties. Communication between a party requesting a new record to be added and nodes can be achieved via a specialized TrustChain client software or a web interface provided by the nodes themselves. Similarly, a party interested in confirming the validity of a document with an expired signature would contact a node, read the blockchain, find the relevant entry and compare it to the document that needs signature conformation. Finding the relevant block in the blockchain would be achieved by an indexing system that relies on the document metadata stored in the blockchain. This indexing system might be part of the TrustChain nodes or it might be outside of the system (since the blockchain is freely readable). The basic architecture of the TrustChain system is shown in Figure 1.

While TrustChain cannot extend the life span of a digital certificate, it would provide a guarantee that the document and its signature have remained unchanged since the TrustChain entry was created. Since the digital signature contains the name of its owner, this can be used to confirm the creator of the document at a later date.

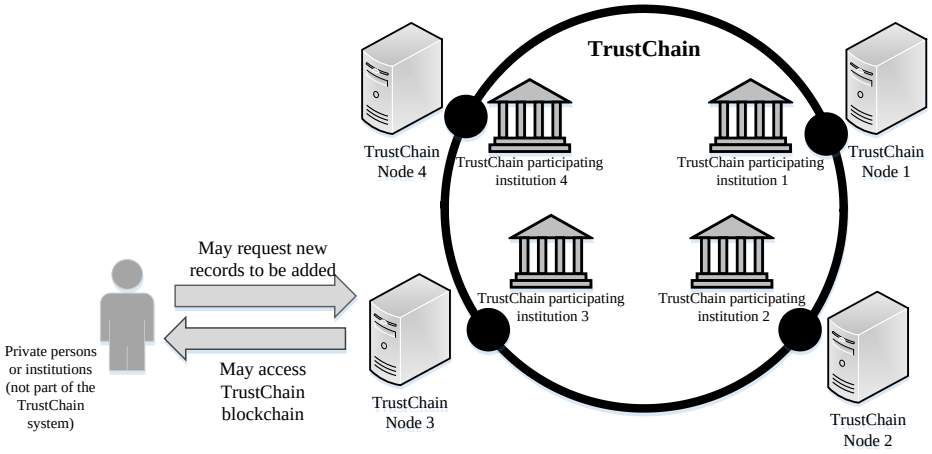

Figure 1. Basic concept of the TrustChain model

3. The TrustChain model processes

The process of adding a document to the TrustChain begins with the interested party selecting a digitally signed document that needs to be preserved. The digital signature of this document needs to be validated by the relevant certification authority. This check is performed at this point as well as later (by multiple TrustChain nodes). The document itself is to be stored outside of the TrustChain system, as the TrustChain stores only a control hash of the digitally signed document. If the full documents were to be stored in the blockchain, it would increase the blockchain to an unmanageable size quickly. It would be possible to build the TrustChain on top of a dedicated database system, similar to BigChainDB (McConaghy et al, 2017), but this is not our intention at this time. As it is, TrustChain is a system that complements other digital document and records management systems, digital archives, or repository systems and does not replace them.

The software preparing the TrustChain record calculates the hash, which is stored in the TrustChain system. As stated earlier, this can be a standalone application that communicates with the TrustChain nodes' API or a web service provided by the nodes. A link to the document (stored in an outside service), a timestamp and any relevant metadata (entered by the user) are added to the hash and a TrustChain record is formed. The record can then be forwarded to a TrustChain node for inclusion into the blockchain. This process is shown in Diagram 1.

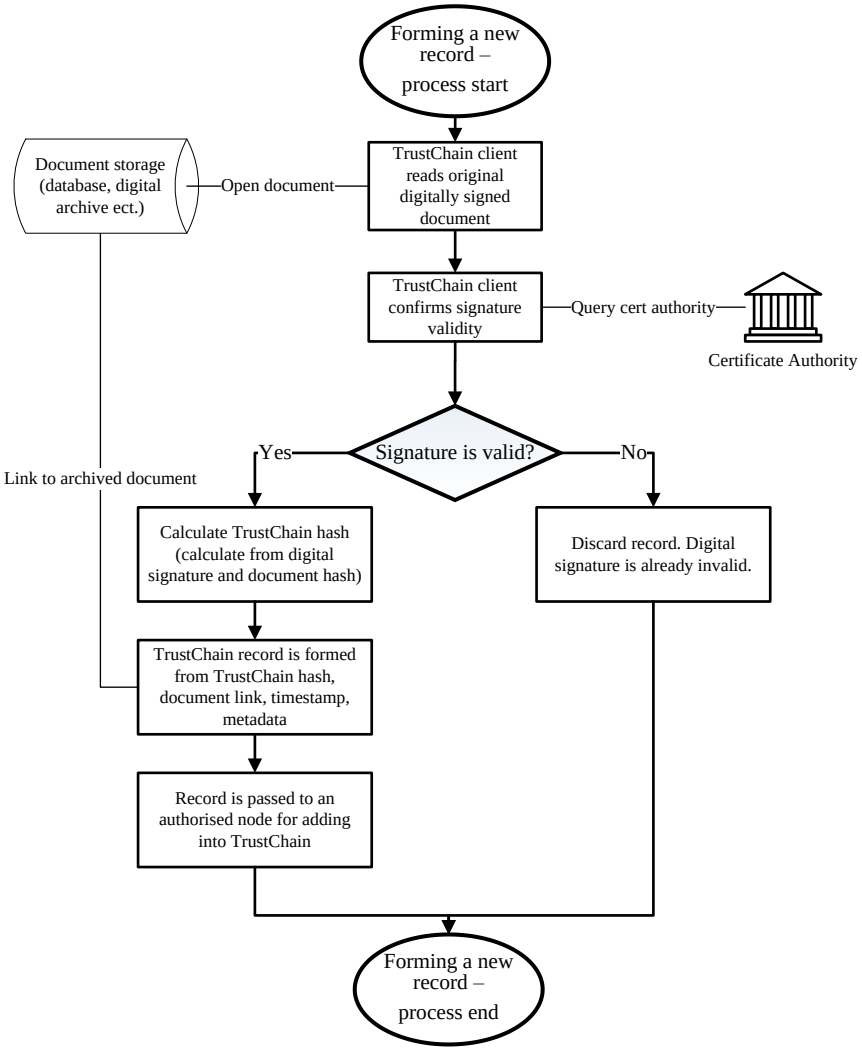

Diagram 1. Adding a document record to the TrustChain

The node will check signature validity and create a new record to be added to a joint record que (which will later form a new TrustChain block). Signature validity is to be confirmed automatically by the node accepting the block. The exact process by which this is achieved will depend on the format of the document (file) and its digital signature. Which documents and signatures can be checked will depend largely on the participating institutions and their requirements. Generally, this step involves checking that the document has not been tampered with after signing (by recalculating the document hash and decrypting the signature hash) and sending the certificate to the certificate

authority for validation. Describing this process in detail is beyond the scope of this paper (this will be addressed at a later stage in the development) but there are many existing industry solutions to this problem. For example, a very common digital signature is the x.509[5] and checking the validity of such a signature is easily achieved by using .NET System.Security.Cryptography.X509Certificates class[6] or Java java.security.cert class[7]. Since the proposed blockchain solution doesn't depend on or store the document or the signature and only interacts with them while checking signature validity this part of the system is independent and we except it to grow and change to be able to accommodate future file formats or digital signature types.

It should be noted that the system makes no effort to eliminate documents signed by the compromised certificates. This is out of scope for TrustChain. The security of a certificate is obviously a responsibility of the certificate owner and his or her certificate authority. The best the TrustChain can do is to make use of protocols such as the Online Certificate Status Protocol[8] to identify the revoked certificates.

The process of adding records to a block and writing that block into the blockchain is left exclusively to TrustChain nodes (ran by trusted providers, most likely archival institutions that are members of the TrustChain system). At this stage of development, the nodes act in a round robin system. Once a node comes to its turn it collects new (candidate) records from a queue and attempts to validate all signatures. If a signature fails, the record is discarded as invalid and new records are collected. Once a sufficient amount of valid records is found, they are added to a block. We still do not add this block to the blockchain. Before this happens, we also require a certain number of other nodes to confirm signature validity of all records. The required number depends on the total number of available TrustChain nodes and the required level of reliability (the more nodes rechecking the records, the more reliable the vote will be). Since the number of participating institutions is not known, at this, early stage we will assume that all participating institutions maintain a node and they all vote on every block. Should the number of institutions rise to a number where having everyone vote becomes a performance issue, a smaller, randomly selected, subset of nodes can vote for each block. This subset should change for every block. If the majority of voting nodes agree that the block is valid it can be added to the blockchain (after having its hash calculated from its contents

[5] IETF RFC 5280, https://tools.ietf.org/html/rfc5280 (Accessed 29.9.2017).

[6] MSDN Library, .NET Development, Framework Class Library, System.Security.Cryptography.X509Certificates Namespace (Accessed 29.9.2017).

[7] Java™ Platform Standard Ed. 7 Online documentation, Package java.security.cert. (Accessed 29.9.2017).

[8] ITEF RFC 6960, https://tools.ietf.org/html/rfc6960 (Accessed 29.9.2017).

and the previous block's hash). Otherwise, the block is discarded and the records that formed it are returned to the new records queue (Diagram 2).

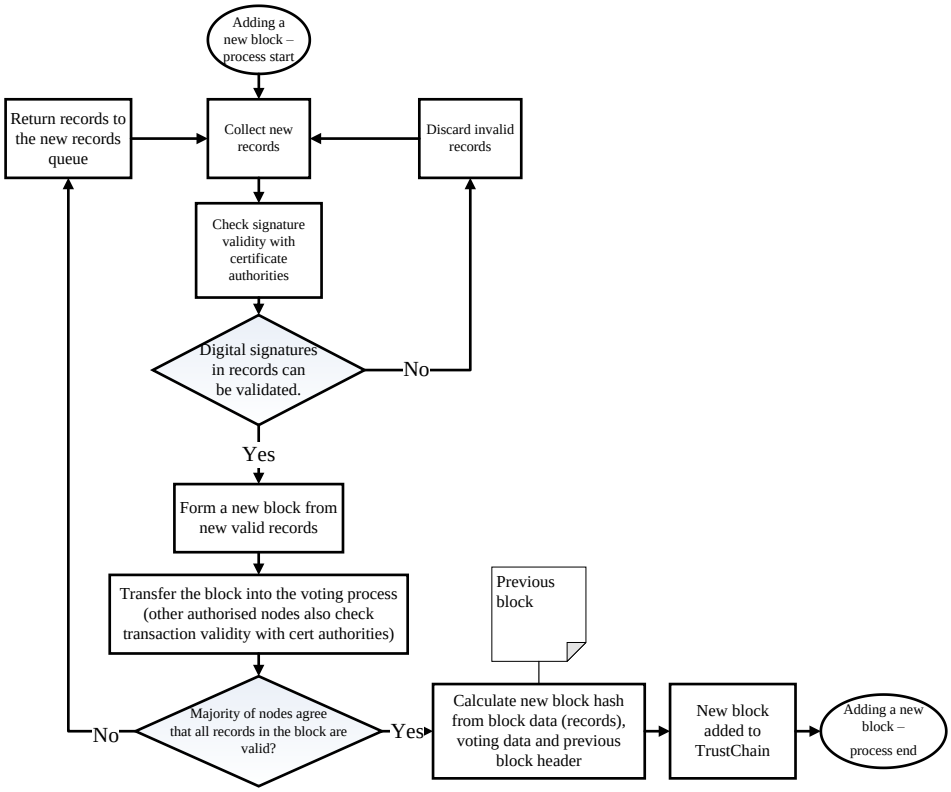

Diagram 2. Adding a new block to the TrustChain

The confirmation process of the (expired) digital signatures begins with finding the relevant records in the TrustChain blockchain. For this, the TrustChain relies on the recorded document metadata. Special services that allow searching the blockchain will need to be built as part of a standalone TrustChain clients or web services. Since the blockchain is written in pure text form, it is also possible to download it and search it without a specialized tool but this might prove troublesome for some users.

Once the relevant record is identified, all that needs to be done is to recalculate the hash from the original document and compare it to the one written in the TrustChain. If these hashes match, one can reliably claim that the document and its signature have remained unchanged since the date indicated by the blockchain record timestamp (Diagram 3).

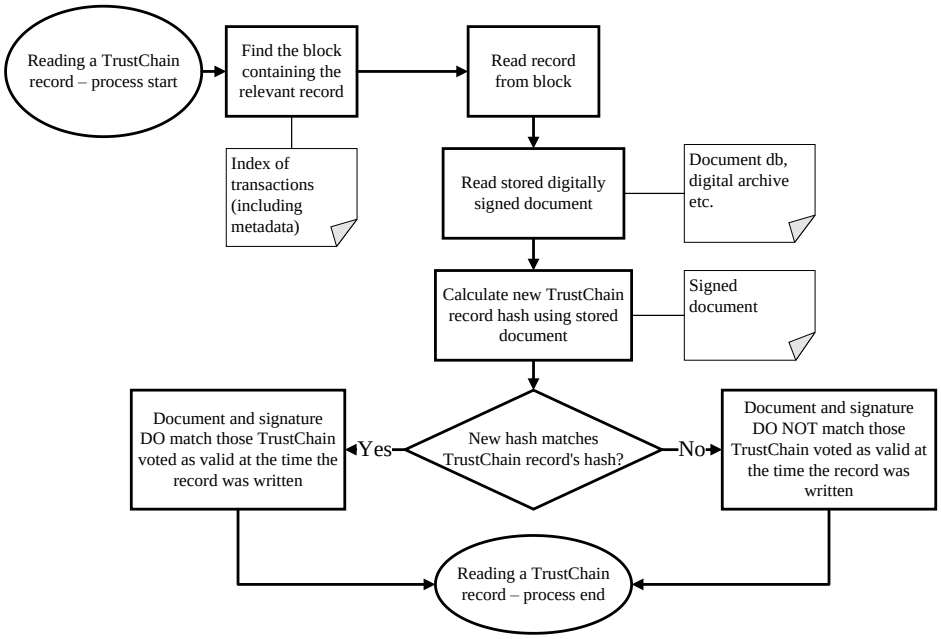

Diagram 3. Reading a record from the TrustChain

4. The TrustChain model's record and blockchain format

This chapter describes the core data structure of the TrustChain system – its blockchain. At this time, the TrustChain blockchain is designed as a plain text file with its contents written in JSON format. We decided on the JSON format because of its widespread use, Internet service-oriented design and human readability (IETF RFC 7159, 2014). Like all blockchains, the solution proposed here is comprised of multiple blocks that form an immutable chain by including a hash which is calculated from the current and the preceding block. These blocks can be further broken down into three sections: header, records and votes. These sections are further described in order of their creation during the process of writing new data into the blockchain.

The record section is at the beginning. Each record contains information about a single digitally signed document and its hash. Document hashing algorithm to be used is most likely SHA256, but it could change over time. This hash, along with a document link and any relevant metadata, are the most important parts of a TrustChain record. Record structure is presented in the following code.

```
{
"id": "<record hash to be used as uID>",
"version": "<model version>",
"record": {
      "timestamp": "<transaction timestamp>",
      "certAuthName": "<cert auth name>",
      "certAuthID": "<cert auth id, if available>",
      "certAuthApiLink": "<cert service link>",
      "data": {
            "TrustChainHash": "<document cert hash>",
            "docLnk": "<document link>" },
      "metadata": {
            "docRefCode": "<document reference code>",
            "docTitle": "<title of issued document>",
            "docCreator": "<name of document creator>",
            "docCreationDate":"<document creation date>" }
         }
}
```

Most of the fields are self-explanatory but few needs to be clarified. The "version" key refers to the version of the data model used to create this record. It is certainly conceivable that the format of the record will change over time and it is also possible to have standalone clients which might not have been updated regularly. Because of this, different versions of the records might appear in the same block and that is why the record data model version needs to be recorded on a record level.

The metadata subsection, which relies on the ISAD(G)'s essential set of elements[9], needs to contain all the information necessary to index and later find the document record in the blockchain. On the other hand, it would not be wise to overburden the blockchain with unnecessary information or fields that will often be left blank. This section requires fine balance and is still under review by the group.

The metadata section might also contain information pertaining to the archival bond (Lemieux and Sporny, 2017). While the primary purpose of the TrustChain is not to store complete documents (or records) and their metadata, since this is clearly a task for an external storage or archival solution, it would certainly add to their functionality. It could be used to add archival bond information to storage systems which otherwise might lack such features. In their paper, Lemieux and Sporny propose the use the OP_RETURN field of the Bitcoin transactions to store archival bond metadata. Since we propose a specialized system with its own blockchain, an implementation of the archival bond syntax could be significantly simpler. Aligning to the original proposal, a

[9] ISAD(G), General International Standard Archival Description, Second Edition, ICA, Ottawa 2000, p. 9, http://www.ica.org/sites/default/files/CBPS_2000_Guidelines_ISAD%28G%29_Second-edition_EN.pdf (Accessed: 30.06.2017).

subfield of the metadata section could be implemented which can contain the needed information and the additional signatures and links to the original transaction which a Bitcoin OP_RETURN field implementation requires can be omitted. The only data needed to be stored would be the ontology used and whatever specific fields it requires to insure the archival bond remains unbroken. An archival bond subfield can be added to the metadata information and any document that requires an archival bond can make use of it (implementing an appropriate ontology).
Multiple record subsections form the records section of a block. Following this section is the votes section. This section is again comprised of multiple vote subsections. The whole section is formed at the time of block creation by the originating node, but nodes that are indicated as voting nodes enter their vote in the "is_block_valid" field. The structure of this section is presented in the following code.

```
{
"nodeID": "<unique id of the node voting>",
"nodeSig": "<signature of voting node, hash of vote data>",
"vote": {
      "blockCandidate": "<id of the voted block>",
      "is_block_valid": "<true | false >",
      "timestamp": "<voting timestamp>"}
}
```

The voting system is based on a simplified version of the system used in the BigChainDB system (McConaghy et al, 2017). Once the originating node receives the responses from voting nodes (filled out vote fields in its block), it confirms the votes as valid by checking the voting node signatures. This method is under review by the group as it adds a public-private key element into TrustChain whose purpose is to avoid reliance on such systems. However, since the signing occurs in the voting section it is not needed to reconfirm it at a later date to validate document record included in the block. It is also easy to maintain certificates for the participating and previously participating nodes using the TrustChain infrastructure itself (in this case the TrustChain acts as a certificate authority for identification and authentication of the voting nodes). This might be an acceptable compromise. Since adding blocks to the TrustChain is a closed system, another possible method of insuring vote safety would be to skip this field altogether and implicitly trust the voting node responses. This would require security measures at the network and system levels to guarantee that the votes originated from the nodes in the TrustChain network, and have not been changed. Describing such a solution is beyond the scope of this paper but remains as a possibility during further development of the system.

Once the votes subsection has been filled in with sufficient votes confirming the block's validity, the final hash of the block is calculated and the block is written into the blockchain. The hash of the block is calculated from the entirety of the current block and the header of the previous block (its own hash and id). This is presented in the following code.

```
{
"blockHash": "<hash of block>",
"blockID": "<block order number>",
"block": {
   "previousBlockHash": "<hash of previous block>",
   "timestamp": "<timestamp of block creation>",
   "nodeID": "<unique id of the node creating the block>",
   "records": [
      { <record1> }
      { <record2> }],
   "votes": [
      { <vote1> }
      { <vote2> }],
         }
}
```

Finally, the complete architecture of the TurstChain blockchain is illustrated in the Figure 2.
As stated, this blockchain would be freely available for downloading to any interested party but only the TrustChain members (authorised nodes) would be allowed to write new data. They may do so to fulfil their own archiving requirements or at the request of any person or institution which needs such a (trusted) service.

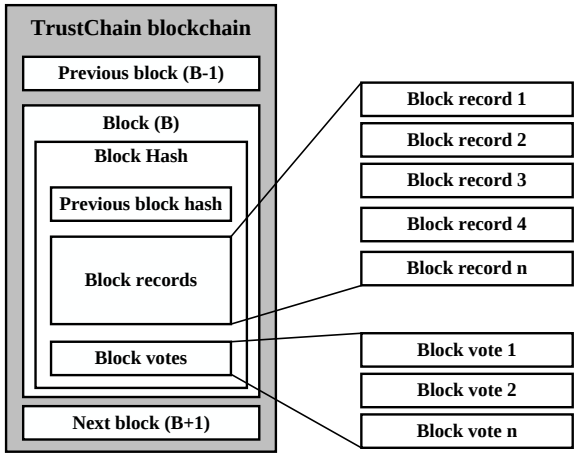

Figure 2. The TrustChain blockchain architecture

5. Existing standards for long-term preservation of digitally signed documents – time stamping

Long-term preservation of digitally signed documents is not a new concern. The first commercial application of an RSA digital signature has made an appearance in Lotus Notes[10], almost three decades ago. One way to address out-of-date certificates is to digitally time stamp the signed documents. According to Volarević and Stančić[11] this process is standardised by the international ISO/IEC 18014 and the ETSI 319 422 standard and, while it does provide a solution, it requires the document to be periodically restamped. On the other hand, the model proposed here attempts to eliminate this requirement. The technical aspect of the ETSI standard is based on the RFC 3161 – Time-Stamp Protocol (TSP). This RFC describes the process of time stamping a document by adding a trusted date and time to the document that are protected by a public key of the time stamp issuer. Since the time stamping process is reliant on the trusted time stamp providers, which use their own private and public keys to prove the document was indeed untouched since the time of stamping, they suffer from problems similar to the digital signature itself. They can be used to extend the lifetime of a digital signature but are not a permanent solution. The RFC itself makes note of this in chapter 4: Security Considerations[12].

Point 2 of chapter 4 states: "When the TSA[13] private key has been compromised, then the corresponding certificate SHALL be revoked". A revoked certificate invalidates all the existing time stamps (or tokens). In this event the RFC does not provide a reliable way of distinguishing between the time stamp tokens which are valid from the ones which are compromised and suggests that an audit trail of all tokens (stamps) should be kept in attempt to distinguish between the two. In any situation after such an event, all the documents (even valid ones) would have to be time stamped again, which will be a problem for those whose original digital signature certificates have expired. TrustChain nodes also use the public-private key system to sign their votes, but since every entry into the TrustChain block requires multiple nodes to confirm the entry validity, multiple nodes would have to have their private keys compromised at the same time for an attacker to be able to write an invalid entry into the blockchain. Depending on the number TrustChain nodes this makes such an attack on the system highly impractical.

Point 3 of chapter 4 raises another limitation that does not affect TrustChain. This point states: "The TSA signing key MUST be of a sufficient length to allow for a sufficiently long lifetime. Even if this is done, the key will have a

[10] IBM developerWorks. The History of Notes and Domino.

[11] Volarveić, Ira; Stančić, Hrvoje. Standards for electronic time stamps and the possibilities for their application in archival practice.

[12] IETF RFC 3161 - Time-Stamp Protocol (TSP).

[13] Time stamp authority.

finite lifetime". This limitation comes from the fact that keys are expected to become vulnerable after a certain period, even if no vulnerabilities inherent to the cryptographic algorithm are present, because of increased processing power. The relevant RFC suggest that as a key length or algorithm reaches the end of its lifetime the documents should be time stamped again with new keys. ETSI 319 422 refers to ETSI TS 119 312[14] to define how long the lifetime of a certain key length (and cryptographic algorithm) is. This document only attempts to predict key length/algorithm durability for up to 10 years or up to the year 2030 (and most combinations do not last even that long). This is insufficient for archiving needs in the context of long-term preservation since many records maintain relevance for much longer and are legally required to be preserved (e.g. 70 years or permanently). As in the previous point, the TrustChain node private keys will also, inevitably, become obsolete and invalid and will need to be changed after a long period but, in the case of TrustChain, this will not affect existing records. This is because TrustChain stores its records in an immutable blockchain and the key is only relevant at the moment of block addition, the fact that certain key length/algorithm combinations will become obsolete will not require "restamping" of old entries as it does in the case of time stamping described by the relevant ETSI standard and RFC document.

In contrast to ETSI 319 422 and RFC 3161, which require periodical restamping, TrustChain provides a (more) permanent solution by writing its entries into a blockchain. Considering this, it is obvious that while the goal is similar, TrustChain is more than a time stamping service – it provides a way to securely store its entries without the need to restamp them. This might be an improvement but it comes at a price. In its current form, TrustChain assumes the existence of a network of trusted (archival) institutions. Not every party might be willing to trust these institutions, or an insufficient number of them might be willing to participate in such a system. It should be noted that the time stamping standards also require existence of institutions that will provide the service but because they require a single institution per service they can be considered easier to implement (and indeed are since many time stamping services already exist). One of the additions to the system currently being considered by the authors and the InterPARES Trust's research group is the addition of a third-party time stamp (instead of simply using TrustChain node clocks) to TrustChain records. This would insure the system encompasses the advantages of both solutions but would further complicate TrustChain as regards of the number of participating institutions and required third party services.

[14] ETSI TS 119 312 V1.1.1 (2014-11) – Electronic Signatures and Infrastructures (ESI); Cryptographic Suites.

6. Conclusion and further work

We have presented a possible solution relevant for the long-term preservation of digitally signed documents. The proposed system is not an archival or digital preservation system for the documents themselves, but rather a standalone system which works in concert with such systems in order to provide the ability to reliably store information on the expiring digital signature validity (or the validity of signatures whose certification authorities no longer exist), without having to trust any single institution. To achieve this goal, the blockchain technology can be relied upon. The proposed system relies on the involvement of a group of trusted institutions that are interested in implementing such a system. Once such a group is identified and the system is implemented, it can be made available to any interested party. The single largest downside of the model is that it only solves the problem for the documents with valid digital signatures. It does not directly provide a solution for the existing documents whose certificates have already expired. These would need to be resigned before having their records written into TrustChain, or a separate blockchain-based solution for storing the validated signatures should be developed and connected to the TrustChain solution.

The proposed model is a prototype and is one of a few possible solutions to the problem of long-term preservation of digitally signed documents being currently pursued by the authors. Further research of this model will include consultation with archival institutions about their willingness to participate in such a project, refining model details with regards to archival institutions' needs, publishing a full project whitepaper, development of a working prototype and testing it in various recordkeeping and archival preservation situations.

Acknowledgements

The research presented here is a part of a much broader research study "Model for Preservation of Trustworthiness of the Digitally Signed, Timestamped and/or Sealed Digital Records (TRUSTER Preservation Model)" which is part of the international multidisciplinary research project InterPARES Trust, http://www.interparestrust.org.

References

A Glossary of Archival and Records Terminology, Society of American Archivists, https://www2.archivists.org/glossary/ (Accessed 30.05.2017)

ETSI EN 319 421 V1.0.0 (2015-06) - Electronic Signatures and Infrastructures (ESI); Policy and Security Requirements for Trust Service Providers issuing Time-Stamps (draft). European Telecommunications Standards Institute, 2016, (Accessed: 4.9.2017)

ETSI EN 319 422 V1.1.1 (2016-03) - Electronic Signatures and Infrastructures (ESI); Time-stamping protocol and time-stamp profiles. European Telecommunications Standards Institute, 2016, (Accessed: 4.9.2017)

ETSI TS 119 312 V1.1.1 (2014-11) - Electronic Signatures and Infrastructures (ESI);

Cryptographic Suites. European Telecommunications Standards Institute, 2014, (Accessed: 4.9.2017)

Haber, Stuart; Stornetta, W. Scott. How to Time-Stamp a Digital Document. *Journal of Cryptology*. 3(1991), 2, pp. 99-111.

IETF RFC 3161 - Time-Stamp Protocol (TSP), Internet Engineering Task Force (IETF), 2001, https://www.ietf.org/rfc/rfc3161.txt (Accessed: 4.9.2017)

IETF RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Internet Engineering Task Force (IETF), 2008, https://www.ietf.org/rfc/rfc5280.txt (Accessed: 29.09.2017)

IETF RFC 6960 - X.509 Internet Public Key Infrastructure Online Certificate Status Protocol, Internet Engineering Task Force (IETF), 2013, https://tools.ietf.org/html/rfc6960 (Accessed: 29.9.2017)

IETF RFC 7159 - The JavaScript Object Notation (JSON) Data Interchange Format, Internet Engineering Task Force (IETF), 2014, https://tools.ietf.org/html/rfc7159 (Accessed: 30.6.2017)

InterPARES Trust Terminology Database, http://arstweb.clayton.edu/interlex/en/index.php (Accessed 30.05.2017)

ISAD(G), General International Standard Archival Description, Second Edition, ICA, Ottawa 2000, http://www.ica.org/sites/default/files/CBPS_2000_Guidelines_ISAD%28G%29_Second-edition_EN.pdf (Accessed: 30.06.2017)

ISO 15489-1 Information and Documentation – Part 1: General, ISO, 2016, https://www.iso.org/standard/62542.html (Accessed: 6.9.2017)

ISO/IEC 18014-1:2008: Information technology – security techniques – time-stamping services – part 1: Framework. ISO/IEC, 2008, https://www.iso.org/standard/50678.html (Accessed: 6.9.2017)

Lemieux, Victoria L.; Sporny, Manu. Preserving the Archival Bond in Distributed Ledgers: A Data Model and Syntax. *Proceedings of the 26th International Conference on World Wide Web Companion*. Perth: WWW '17 26th International World Wide Web Conference, 2017, pp. 1437-1443.

McConaghy, Trent; Rodolphe, Marques; Muller, Andreas; De Jonghe, Dimitri; McConaghy, T. Troy; McMullen, Greg; Henderson, Ryan; Bellemare, Sylvain; Granzotto, Alberto. BigchainDB: A Scalable Blockchain Database. Berlin: BigchainDB GmbH, 2016. https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf (Accessed: 20.04.2017).

Microsoft: MSDN Library, .NET Development, Framework Class Library, System.Security.Cryptography.X509Certificates Namespace. https://msdn.microsoft.com/en-us/library/system.security.cryptography.x509certificates(v=vs.110).aspx, (Accessed: 29.9.2017)

Nakamoto, Satoshi. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf (Accessed: 15.04.2017).

Oracle: Java™ Platform Standard Ed. 7 Online documentation, Package java.security.cert. https://docs.oracle.com/javase/7/docs/api/java/security/cert/package-summary.html (Accessed: 29.9.2017)

The History of Notes and Domino. IBM developerWorks. International business machines, 2007, https://www.ibm.com/developerworks/lotus/library/ls-NDHistory/ls-NDHistory-pdf.pdf (Accessed: 6.9.2017)

Volarević, Ira; Stančić, Hrvoje. Standards for electronic time stamps and the possibilities for their application in archival practice. Arhivi i domovinski rat. Zagreb: Hrvatsko arhivističko društvo, 2016, pp. 425-435.