computers

MDPI

Article

Digital Archives Relying on Blockchain: Overcoming the Limitations of Data Immutability

Hrvoje Stančić [1,*] and Vladimir Bralić [2,*]

[1] Faculty of Humanities and Social Sciences, University of Zagreb, 10000 Zagreb, Croatia
[2] Department of Information Technology, University of Applied Sciences Velika Gorica, 10410 Velika Gorica, Croatia
* Correspondence: hstancic@ffzg.hr (H.S.); vladimir.bralic@vvg.hr (V.B.)

Abstract: Archives, both analogue and digital, are primarily concerned with preserving records as originals. Because of this, immutable data as used in a blockchain data structure seem a logical choice when designing such systems. At the same time, archives maintain records which may need to change over the long term. It is a requirement of archival preservation to be able to update records' metadata in order not only to guarantee authenticity after digital preservation actions but also to ensure that relationships to other records, which might be created after an original record has entered the archive (and has been registered in a blockchain), can be maintained. The need to maintain an archival bond, which represents a network of relationships between aggregation of records, i.e., the relationship connecting previous and subsequent records belonging to the same activity, is a prime example of this requirement. This paper explores realisation of the archival bond in the context of blockchain-based archival system by proposing a supporting database system which enables metadata to be changed as required but also significantly simplifies searching compared to searching on-chain information, while keeping the immutability characteristic of blockchain.

Keywords: digital archive; blockchain; database; document; archival bond

Citation: Stančić, H.; Bralić, V. Digital Archives Relying on Blockchain: Overcoming the Limitations of Data Immutability. *Computers* **2021**, *10*, 91. https://doi.org/10.3390/computers10080091

Academic Editor: Victoria Lemieux

Received: 11 June 2021
Accepted: 19 July 2021
Published: 21 July 2021

1. Introduction

One of the key questions in modern society is to what extent digital information can be trusted. Archival institutions have the opportunity to participate in the creation of a new architecture of trust. However, as traditionally trusted institutions, archives need to constantly prove that they can preserve digital records as trustworthily and as long as paper records. This is a challenging task because of the metamorphic aspects of information technologies—they are constantly changing and advancing. Therefore, archives need to constantly monitor ongoing development and react when needed. The reaction should be in the form of digital preservation action, i.e., conversion of records from an (almost) obsolete file format to the current one, migration from an older generation of storage media to the new one, emulation of older software solutions in the contemporary software environment, or virtualisation of old systems in the new hardware and software environment, all the time taking care that the records stay trustworthy, i.e., accurate, reliable, and authentic. Batista et al. explore the taxonomy of records' trustworthiness and explain how the concepts of trust and provenance are closely related as well as how they can be realized in the context of blockchain [1]. Preserving records' integrity, or the ability to prove that the record did not change over the preservation period, and preserving identity, or the ability to prove that the record is what it purports to be, represent two prerogatives for confirmation of authenticity of digital records. The concept of the archival bond, explained in detail in Section 2, relates to records' identity preservation.

This paper focuses on a particular type of digital record—the digitally signed and/or sealed records. Digital preservation of those records is challenging because digital signatures and seals rely on certificates usually issued for the period of two to five years,

after which their validity expires. Moreover, during that time certificates may be revoked. Such a situation is on the one hand favourable in terms of digital security, but on the other hand highly unfavourable in terms of long-term (e.g., 70 years) digital preservation. One approach to this challenge is regular (re)timestamping, or addition of archival time stamps before the certificates expire, but this approach requires keeping track of the certificate expiration date of many millions of individual records in a digital archive and invoking a preservation action just before that. A more elegant approach was proposed by Bralić, Kuleš, and Stančić in 2017 when they developed the TrustChain model—a blockchain-based validity information preservation (VIP) model [2]. The model was upgraded by Bralić, Stančić and Stengård in 2020 to encompass digital signature certification chain preservation [3]. This paper continues the research by proposing the TrustChain supporting system model needed for combining an immutable blockchain-based archival system architecture preserving digitally signed and/or sealed records with a concept of incrementally changing the archival bond relying on metadata expansion over time.

The paper is structured into several sections. After this introduction, Section 2 elaborates on the concept of the archival bond with the aim of clarifying how it impacts digital preservation requirements. Next, Section 3 provides a brief introduction to blockchain data structures and presents a critical part of the TrustChain system as defined in the previous research of the authors [2,3]. The presented data structure does not allow fast searching or metadata changes so the following sections investigate a data model which would enable those requirements. A literature review, presented in Section 4, was used to determine the most appropriate underlying technology for such a system, which is followed by Section 5 where the appropriate logical data model is determined by extraction of mandatory metadata elements of several archival industry standards. Finally, Sections 6 and 7 use the proposed logical data model to develop and present database models which can be used for building the supporting system based on the technologies determined as the most appropriate in Section 4. The models proposed here will enable further development of the TrustChain system following archival standards and requirements. The paper concludes with suggestions for future research and development of the TrustChain system.

2. Archival Bond

Duranti states that the concept of the archival bond is at the core of archival science. She explains that the archival bond represents "the network of relationships that each record has with the records belonging in the same aggregation. (. . .) The archival bond first arises when a record is set aside and thereby connected to another in the course of action" [4], (pp. 215–216). Duranti and MacNeil make clear that "this implies that the entity record is in formation during its period of activity, as the aggregations in which the record belongs are constantly accruing and changing, according to the natural dynamism of the records system. When the records become semi-active (i.e., they are 'no longer needed for the purpose of carrying out the action for which they were created, but which are needed by the records creator for reference' [5]) their documentary context is subject to development and change as only some components of each class will be removed from the records system and accumulated together to build up series. When the records become inactive (i.e., they are 'no longer needed to conduct current business but preserved until they meet the end of their retention period' [5]) the archival bond is defined and stabilized and not subject to further changes. This means that the archival bond, which is an essential component of the record, changes and develops until the record reaches the point of stability, that is, until the action to which it relates is concluded" [6], (p. 61). Lemieux and Sporny explain that "the archival bond must be made explicit and interpretable in order to ascertain the unique identity of each document as a record of the procedurally bound facts contained within it" [7], (p. 1438). Further, they point out that in the context of blockchain "it is a common mistake to think that because every block of transactions (and thereby every transaction) in a proof-of-work blockchain is transitively bonded to every previous block, by virtue of the way proof of work functions, that the archival bond is preserved.

However, even though the time-ordered nature of the transactional records is preserved, the link to their procedural context, and relationship to other transactional records relating to the same procedure, is not" [7], (p. 1439). This prevents searching records registered on the blockchain according to their archival bond relationships as well as interlinking the records already on-chain with the new ones belonging to the same activity. Therefore, this paper explores realization of the archival bond in the context of blockchain, further developing the data model presented by Lemieux and Sporny [7], and suggests a solution based on introduction of supporting secure linkages to non-blockchain-relying systems such as those based on SQL or NoSQL databases. Built around a central repository, or a digital archive relying on blockchain. Such a supporting system can not only enable metadata to be changed as required but also significantly simplify searching compared to searching on-chain information.

Lemieux and Sporny [7] addressed the challenge of archival bond realization on blockchain. They proposed a data model which enabled archival bond preservation in a blockchain-based archival system. Their realization suggested using standard distributed ledger fields, such as the Bitcoin's transaction metadata OP_RETURN field for storing archival bond information. Such fields are limited by size constraints and storing a structured data model (as is proposed) might require special encoding. The data model proposed in this paper takes a different approach. It is designed for a completely new system and the mentioned restrictions did not apply. Lemieux and Sporny [7] suggest using special ontologies to create archival bond categories which can then be used to represent the archival bond context. On the other hand, TrustChain stores only the bare minimum metadata information on blockchain and uses a separate information system for indexing records and preserving metadata. Therefore, the approach proposed here is tailored after the database systems which can be used to realize the supporting system. This enables records to be connected individually using one-to-one connections or to form larger groups of archival bond records by using any implementation of one-to-many or many-to-many relationships that is available to the supporting system, such as embedded collections in a MongoDB-based system. Such implementation allows any already existing specialized, formal ontology to be used in formation of the archival bond.

3. Blockchain Data Structure

This section presents a general overview of blockchain data structures and shows their implementation by the TrustChain system. The purpose of this overview is to identify challenges caused by using an immutable data structure in a digital archival system. The section also serves as a brief introduction into the data structure used by the TrustChain blockchain, as detailed in the authors' previous papers [2,3].

While the contemporary blockchain data structures are traditionally connected with cryptocurrencies and the financial sector, they have found application in almost every field of human activity. There is a tendency to incorporate blockchain in every new information system which could benefit from it—and many indeed can. Since blockchain provides an added layer of data security, in theory, it makes a useful addition to any information system which aims to preserve information.

Such a vast scope of application necessitated development of various blockchain architectures. At the top level one can differentiate between two archetypes of blockchain architecture pairs: public and private, and permissioned and permissionless.

Cryptocurrencies mostly fall in the public type. This architecture allows a vast number of (possibly anonymous, such as in the case of Bitcoin) participants with the integrity of the data commonly being ensured by requiring proof-of-work to be provided for each block addition into the chain.

Private blockchains are maintained by a number of trusted people or institutions, i.e., nodes. The number of participants is usually significantly smaller than in public blockchains. This, along with the fact that all participants are clearly identified, means that

the blockchain does not need to rely on mathematical problem solving such as proof-of-work to ensure data integrity.

As opposed to permissionless blockchain, where any user can record data on-chain, in permissioned blockchain only authenticated, trusted users can record data which are then sealed on-chain using normal methods of combining hashes of current and previous data. Because of its intended primary user, a coalition of archival institutions, a private, permissioned blockchain model is the preferred solution for a system such as TrustChain [2]. It should be noted that a small number of authenticated users which can enter new data into a private blockchain structure does not exclude read-only access for a much wider userbase, or even public access.

Without the need for any proof-of-work, the proposed blockchain structure and its use are significantly simpler compared to the blockchain using proof-of-work, such as various cryptocurrency blockchains. This approach merely needs to link data blocks using hashes. Within the blocks, data are best organized into Merkle trees [8]. A Merkle tree is a full binary tree which enables confirmation of a record's data integrity by calculating only the hashes between the tree leaf containing the record and the root of the tree.

The tree itself can be stored into an array which is easily translated into an actual file stream. Each block in this structure is connected to its preceding and following block by a linear chain of hash values. This logical architecture is shown in Figure 1.

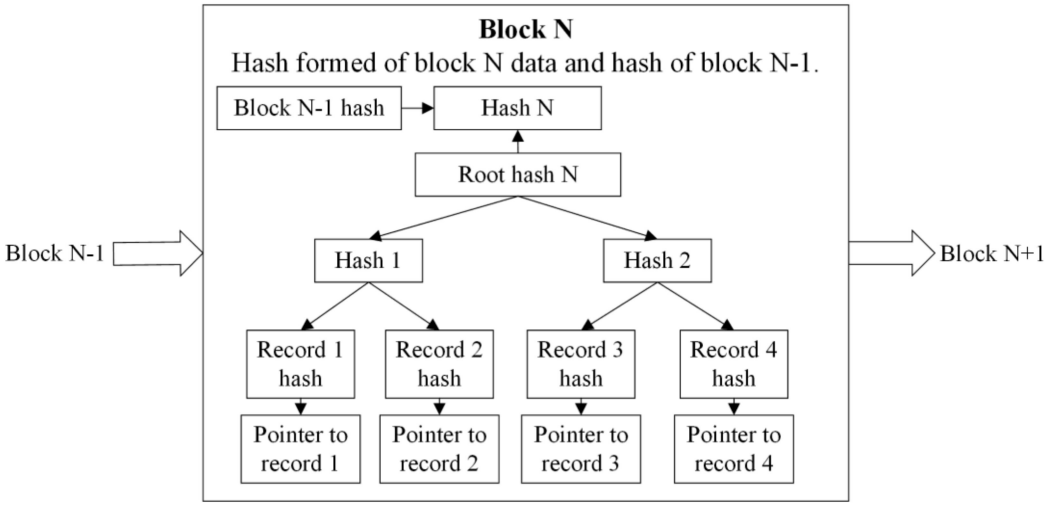

Figure 1. TrustChain logical data structure.

In Figure 1 the "pointer to record X" field is an abstract. This field may contain fixed length information, as suggested in the upgraded TrustChain model [3], or links to actual off-chain files, as suggested in the initial version of the TrustChain model [2].

Such a logical structure is simple to translate into a linear stream using standard mapping of a binary tree to an array. For example, using the upgraded TrustChain model, a block can be written into a file stream as the structure shown in Figure 2.

```
<Block ID>, <Block hash>, <Previous block hash>, <Block Creation Timestamp>,
<ID of creating node>, <Root hash>, <Hash 1>, <Hash 2>, <Record 1 Hash >,
<Record 1 data pointer>, <Record 2 Hash>, <Record 2 data pointer>, …
```

Figure 2. TrustChain file structure.

The upgraded TrustChain model requires revocation and vote data to be added to each block. This is not included in Figure 1 or in Figure 2 in order to maintain compatibility of the solution proposed in this paper with the initial version of the TrustChain model and to show only key elements. However, adding these fields to a block is trivial. Since

this information is fixed length and not numerous (especially the vote fields), they are not included in the hash tree but rather linearly appended to the end of the file.

The presented data structure can be traversed in a very efficient manner. Since the actual digitally signed records are the only variable length data, and they are stored off-chain—most likely as separate files, and only a fixed length pointer to each record needs to be maintained—the blocks can have fixed size. This means that the position of each block can be calculated using their ID (sequential number) in constant time complexity. Reading useful data within the block itself is also conducted in constant time, while confirming the validity of a hash within a block is conducted in logarithmic time. However, all of this assumes the exact record ID (or sequential number) is known. If this is not the case, a linear search of all blocks must be conducted to find the position of an individual record. While this search can be narrowed down by using the block creation date as a filter (if the date is known), we still consider this highly inefficient.

In order to propose a more efficient solution firstly we have comparatively reviewed the existing body of literature discussing blockchain-based data storage systems. Secondly, we have compared four relevant metadata standards and mapped their mandatory and optional elements to make the proposed solution as intercompatible as possible. Based on that we are proposing the TrustChain supporting system model which supports preservation of both digitally signed and/or sealed records and the archival bond between them.

4. Literature Review

The purpose of the paper is to propose a suitable data support system for the TrustChain archival blockchain model. TrustChain itself was conceived by the authors of this paper, with contributions from other researchers, during the InterPARES Trust project (https://interparestrust.org, accessed on 19 July 2021.) and is best described in the original paper [2], which detailed an early concept of the model. The original model assumed that the archival records and their digital signatures would be available to the TrustChain nodes for signature confirmation at the phase of ingest to the digital archive. This requirement proved challenging when dealing with confidential records. Therefore, a new, updated model [3] was created which avoided this issue by requiring only signatures or digital certificates to be examined.

This paper continues the research by proposing a TrustChain supporting data system solving the conflicting requirements of blockchain (immutable), and archival bond (changeable until records become inactive). The supporting data system is an information storage system separate from the blockchain whose purpose is to enable TrustChain users to alter certain metadata information (most notably add archival bond information which might be created after the record has entered the system) and allow the blockchain to be indexed and easily searchable. In order to achieve these goals, the system cannot rely on an immutable data structure, so a dual storage system is needed, consisting of an immutable blockchain core, which guarantees data integrity, and a partially mutable supporting system.

In order to determine the most suitable technology for the proposed supporting system, a literature review was conducted comparing similar systems—both specialized archival systems and general database systems using blockchains to ensure data integrity. Using the literature review approach, we were able to determine which underlying technological solutions are predominantly used and estimate their appropriateness for the proposed system.

Three digital archive models were considered, ARCHAIN [9], Cilegon E-Archive system [10], and Lekana [11]. In addition to these archival systems, several general data storage systems were considered including BigChainDB [12], ChainSQL [13], EthernityDB [14], and Mystiko [15]. A brief overview of these systems is shown in Table 1.

Table 1. An overview of blockchain-based data storage systems.

System	Dedicated Archival System	Centralised	Storage Technology
ARCHAIN	Yes	Yes	Existing state archival system
Cilegon E-Archive	Yes	No	Interplanetary file system (IPFS)
Lekana	Yes	No	Apache Cassandra (noSQL distributed)
BigchainDB	No	No	RethinkDB and MongoDB (noSQL distributed)
ChainSQL	No	No	Any database (SQL or noSQL)
EthernityDB	No	No	Ethereum blockchain
Mystiko	No	No	Apache Cassandra (noSQL distributed)

Several of the examined systems are realised as unique solutions. Thus, they were deemed not applicable to the challenge presented in this paper. They are briefly explained next.

Galiev et al. merely state that ARCHAIN is built upon an existing archival system [9]. While they did provide some insight into the challenges concerning integration with the archival system, the authors did not suggest a solution usable in the context of TrustChain. Still, the paper confirms the hypothesis behind the original TrustChain idea—that blockchains can, and will be, used to ensure data integrity in archival systems.

Cliegon E-Archive system uses Interplanetary file system (IPFS), a distributed file storage system [16] and presents an interesting solution which we, however, do not consider viable for the TrustChain solution. Namely, using IPFS would require TrustChain to develop its own database management system since simply storing files would not suffice for its data support needs because it needs to be easily searchable. However, other researchers have investigated the approach to use IPFS and blockchain in concert to ensure data integrity, e.g., Naz et al. suggested a similar system [17]. The idea has merit and a possible application in the TrustChain solution does exist—such a system could be used to store archived records—but such applications are beyond the scope of this paper.

EthernityDB stores its data on the Ethereum blockchain itself. While such a solution has merit and the volume of data the TrustChain solution stores might not be huge, it will still incur costs for storing data on the Ethereum blockchain. For this reason, using Ethereum was not considered.

The remaining four systems use distributed noSQL databases. ChainSQL is unique in this group as it is equally capable of handling both types of database systems and does not specify any concrete databases used in its creation and testing. This narrows down the choice to either Apache Cassandra [18], used by Lekana and Mystiko, or MongoDB [19], used by BigchainDB, both of which are industry leading distributed noSQL database systems.

From the start, TrustChain was modelled as a distributed system and the use of distributed database was implied, but the use of a noSQL database was not. The literature review has shown a clear prevalence of noSQL databases in similar systems. Since the TrustChain supporting data system does not require a complex relational data model or transactions, a noSQL database makes a preferred choice as a basis for storing metadata, archival bond, and blockchain record location.

While they fall under the same database category, Cassandra Apache and MongoDB are very different systems. Performance wise, Cassandra Apache has shown a slight advantage as data size increases [20]. A more important design decision arises from the different types of data organization between the two systems. Cassandra Apache uses column-based data storage, akin to SQL systems but lacking normalization and foreign key references. MongoDB calls itself a document-based database and stores data in JSON-like format. MongoDB stores JSON data in a binary format—BSON [21]. Because previous

versions of the TrustChain model used JSON to describe data stored on its blockchain, it would make more sense to adopt a MongoDB-based solution than a Cassandra Apache-based one. However, since TrustChain is still a model of an information system, next we will present and compare a data model using both solutions.

5. TrustChain Supporting System Model

In order to model the supporting system, two critical choices had to be made: which technology to use, and which metadata model to use. While indexing is mostly a technical issue, achieved by the supporting system's underlying technology, searchability is not. Archival searches are conducted by exploring metadata collections, and precisely because of this the selection of an appropriate metadata standard is vital for the supporting system.

To determine which metadata set to use, four relevant standards were comparatively analysed, three archival metadata standards—DACS [22], ISAD(G) [23], and PREMIS [24]—and one general digital object description standard—Dublin Core [25] (Table 2).

Table 2. Comparison of metadata standards.

Standard	Archival Standard?	Number of Metadata Elements	Number of Mandatory Metadata Elements
DACS	Yes	25	10
Dublin Core	No	15	0
ISAD(G)	Yes	26	6
PREMIS	Yes	15	2

Since the model anticipates use of an extensible database system (MongoDB or Cassandra Apache), the focus was put on modelling a minimum set of metadata elements required which can later be expanded if needed. The model would benefit from choosing the metadata set which maintains compatibility with as many applicable metadata standards as practical. To achieve this, the selected metadata set should include mandatory metadata elements of all metadata standards supported by the proposed system. Therefore, the metadata standards were comparatively analysed, and the overlapping mandatory metadata elements determined.

Firstly, the metadata standards were examined individually to determine to which degree they are applicable to the TrustChain supporting system.

DACS has the largest number of mandatory elements. It was concluded that many mandatory elements do not favour the envisaged model of storing digital objects complemented with a minimal set of metadata. Elements such as "Name and Location of Repository Element" and "Languages and Scripts of the Material Element" are not vital for the TrustChain supporting system. On the other hand, several basic metadata elements, such as "Title" and "Name of creator", are present in all standards. Deciding to support DACS mandatory metadata elements would require inclusion of fields which would rarely be used. Therefore, it was decided not to fully support it. If required, the system data model can be expanded to support DACS.

Dublin Core has no mandatory elements and can be easily supported by mapping its optional elements to mandatory elements of other standards.

ISAD(G) is very similar to DACS. This was expected since DACS is the United States' implementation of ISAD(G). However, because of the smaller number of mandatory elements, ISAD(G) was found to be much easier to support.

PREMIS has been shown to be the most applicable standard for the use case presented in this paper. This is largely due to the following factors. Firstly, in general, PREMIS has only two mandatory metadata elements. However, when used to describe digital objects, i.e., files and bitstreams, it requires at least one additional element—"objectCharacteristics/format". This element is appropriate to the TrustChain supporting system since it describes the digital format used by the object. The standard is otherwise

very extensive and covers many metadata elements, most of which are optional. Even when they are marked as mandatory, it often means "mandatory if applicable" (such as in the "/format" case), or only applies if certain other fields are used [26]. This allows for a great deal of flexibility. Secondly, PREMIS includes metadata elements which cover digital signature information, vital information for the TrustChain system. Thirdly, PREMIS supports archival bond information via its "relationship" elements. Lastly, PREMIS is designed to be used in XML notation, which is similar to JSON. This simplifies creation of a data model for the MongoDB system.

Table 3 shows mapping of mandatory and optional elements of metadata standards. When a mandatory element is required by one standard but not by another, an appropriate optional element was looked for to fill the gap in the standards comparison table. The table differentiates between mandatory (M), optional (O), and non-existing (X) elements.

Table 3. Mapping of mandatory (M) and optional (O) elements of metadata standards (X—non-existing elements).

DACS	Dublin Core	ISAD(G)	PREMIS
Reference Code Element (M)	Identifier (O)	Reference code (M)	objectIdentifier (M)
X	Type (O)	X	objectCategory (M)
Title element (M)	Title (O)	Title (M)	originalName (O)
Name of creator element (M)	Creator (O)	Name of creator (M)	signatureInformation/signer (O)
Date element (M)	Date (O)	Date of creation (M)	creatingApplication/dateCreatedByApplication (O)
X	X	Level of description (M)	X
Extent element (M)	X	Extent of the unit description (M)	X
Scope and Content Element (M)	X	Scope and content (O)	X
Name and Location of Repository Element (M)	X	Existence and location of originals (O)	storage/contentLocation (O)
Conditions Governing Access Element (M)	X	Conditions governing access (O)	X
Languages and Scripts of the Material Element (M)	X	Language/scripts of material (O)	X
Rights Statements for Archival Description (M)	Rights (O)	X	linkingRightsStatementIdentifier (O)

Following results of the comparative analysis of the mandatory and optional metadata elements in four relevant metadata standards, it was decided to use the PREMIS metadata standard as a basis for the metadata model used in the TrustChain supporting system and to include fields which would enable compatibility with Dublin Core and ISAD(G) standards. The chosen five PREMIS metadata elements to be used as mandatory in the TrustChain supporting system are:

- objectIdentifier—a unique identifier within the TrustChain system.
- objectCategory—PREMIS uses this element to distinguish Entites, Representations, Files, and Bistrams. TrustChain uses this element to differentiate between signed files and digital certificates (both can be recorded independently).
- originalName—name of the record before it enters the TrustChain System.
- dateCreatedByApplication—timestamp of file or signature creation.
- format—used to differentiate specific file formats. In TrustChain this field can also be used to store digital certificate information if the record is a digital certificate.

The above are the mandatory PREMIS metadata elements for a file or bitstream object. Along with those, one non-PREMIS metadata element will be used:

- TrustChainBlockID—this element uniquely identifies the TrustChain data block in which the record is stored. This is a vital element as it enables the metadata set to be used as a search tool for the blockchain.

The chosen 12 PREMIS metadata elements to be used as optional include all signature information elements and relationship elements. The signature elements are:

- signer—name of an individual or an institution, or other identifier, as registered in the digital signature.
- signatureMethod—identifies which algorithms were used for encryption and hashing in the digital signature.
- signatureValidationRules—PREMIS defines this element as "operations to be performed in order to validate the digital signature". This refers to special steps which need to be taken during validation because of special ingest procedures, i.e., the archives-specific process of admitting records into the archive. In the current version of TrustChain this is an optional element intended for possible future use.
- signatureProperties—PREMIS defines this element as "additional information about the generation of the signature". In the current version of TrustChain this is an optional element intended for possible future use.
- signatureValue—the digital signature itself.
- keyInformation—public key used to validate the digital signature.
- signatureEncoding—defines encoding method for signatureValue and keyInformation elements.

Finally, PREMIS relationship elements, which are used to allow archival bond metadata to be recorded, are:

- relationshipType—defined by PREMIS as a "high level categorization of the nature of the relationship". PREMIS vocabulary suggests categories such as dependency, derivation, reference, etc. This element is used to define the relationship recorded in the archival bond but is optional.
- relationshipSubType—same as relationshipType but with a vocabulary composed of narrower terms.
- relatedObjectIdentifierType—defined by PREMIS as "a designation of the domain within which the identifier is unique," this optional element is not needed for regular TrustChain operations since all objects will use the same identifier scheme. However, this element enables the system to maintain archival bond information about records stored in other systems. In that case, this element identifies the digital archive in which the related record is stored.
- relatedObjectIdentifierValue—a unique identifier of the related record.
- relatedObjectSequence—order number of the object relative to other objects in the relationship.

A logical data model, presented in Chen notation, is shown in Figure 3. This model helps better understand the relations between the metadata elements and will assist in the next steps of the TrustChain supporting system data model development.

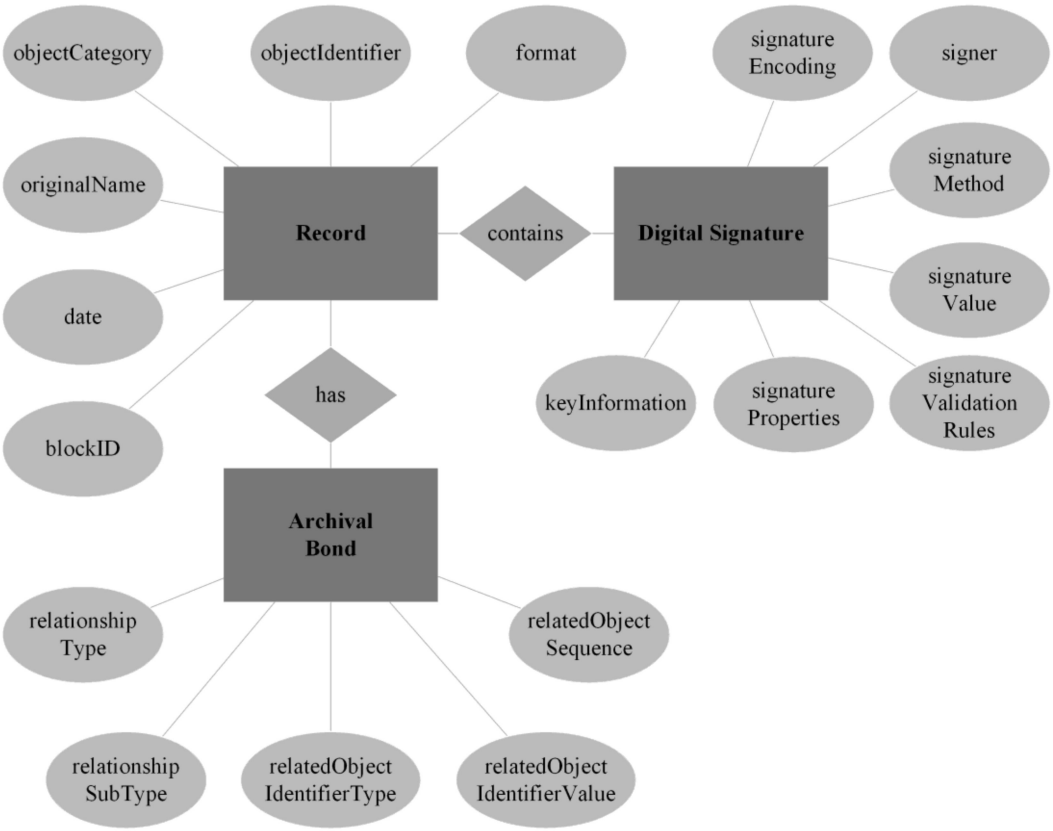

Figure 3. TrustChain supporting system logical data model.

Using the logical model shown in Figure 3, a data model suitable for use in the MongoDB and Cassandra Apache-based systems is formulated next.

6. MongoDB-Based Data Model

MongoDB organizes information in collections and documents which loosely correlate to tables and rows in a relational database management system. MongoDB terminology uses "documents" which should not be mixed with "(archival) records" used throughout the rest of this paper. The MongoDB model presented here uses a single collection with embedded documents to enable relationships. The only relationship which is needed is the archival bond, which is implemented as many one-to-one embedded document relations using PREMIS fields. Figure 4 presents a MongoDB suitable data model.

This data model allows storage of relevant metadata for all the types of records TrustChain is designed to handle, digitally signed files and digital signatures included. By embedding the "digitalSignature" and "arcBond" documents it is possible to store information on multiple digital signatures and the archival bond for each record. An example of such a record is shown in Figure 5. The figure shows a mock-up MongoDB record relating a document whose signature is stored on the TrustChain blockchain with another document by archival bond. The "key" element in the "digitalSignature" document has been shortened to a single line since there is no need to state the full length of non-human readable RSA key.

The TrustChain implementation will include additional elements, i.e., metadata relating to TrustChain specific processes. Information such as date of ingest and the identifier of node responsible for ingesting will be included. However, they are omitted here since definition of the system architecture and its processes represent the next step in the TrustChain development. This paper, and the example shown in Figure 5, focus on the metadata needed to search the blockchain and maintain archival bond information.

```
{
      _id: RECORD_ID,
      category: RECORD_CATEGORY,
      name: RECORD_NAME,
      date: RECORD_CREATION_DATE,
      format: RECORD FORMAT,
      blockID: BC_NUMBER,
      digitalSignature:[
      {
            sigEncode: SIGNATURE_ENCODE,
            signer: SIGNER,
            sigMethod: SIGNATURE_METHOD,
            sigValue: SIGNATURE_VALUE,
            sigValid: SIGNATURE_VALIDATION_RULES,
            sigProp: SIGNATURE_PROPERTIES,
            key: KEY_INFORMATION
      }
      ]
      arcBond: [
      {
            type: RELATIONSHIP_TYPE,
            subtype: RELATIONSHIP_SUB_TYPE,
            id: {
                  idBondType: RELATED_OBJECT_IDENTIFIER_TYPE,
                  idBondValue: RELATED_OBJECT_IDENTIFIER_VALUE
                 }
            bondSeqNumber: RELATED_OBJECT_SEQUENCE
      }
      ]
}
```

Figure 4. MongoDB-based data model.

```
{
      _id: 567321,
      category: "FILE",
      name: "Trade Contract Addendum NA Shipping",
      date: "<2018-04-21>",
      format: "PDF/A",
      blockID: 671,
      digitalSignature:[
      {
            sigEncode: "base64",
            signer: "NA Shipping",
            sigMethod: "RSA-SHA1",
            sigValue: "uooqbWYa5VCqcJCbuymBKqm17vY=",
            sigValid: "",
            sigProp: "",
            key: "MIGfMA0GCSvQGu5 … EURkV1wp/IpIDAQAB"
      }
      ]
      arcBond: [
      {
            type: "replacement",
            subtype: "supersedes",
            id: {
                  idBondType: "TrustChain",
                  idBondValue: 451281
                 }
            bondSeqNumber: 1
      }
      ]
}
```

Figure 5. MongoDB-based data model record example.

In addition to the single collection shown in Figure 4, which is expected to fulfil the needs of most use cases and is expandable at a later date, we propose a second collection which will enable easier searching when the archival bond is the search starting point. The MongoDB-based data model allows the use of many one-to-one connections between records to support the archival bond. The biggest downside to such an approach is the lack of an archival bond object which could be used to easily search for all records connected by archival bond, i.e., participating in the same activity. The presented model of archival bond

relationship data enables creation of multiple archival bond types. One such type can point to an identifier of a different collection, an archival bond object. This would enable the creation of an archival bond catalogue and would allow easy searching of records which contain an identifier corresponding to an archival bond. Such a collection is presented in Figure 6.

```
{
      _id: ARCHIVAL_BOND_ID,
      type: ARCHIVAL_BOND_CATEGORY,
      name: ARCHIVAL_BOND_NAME,
      date: ARCHIVAL_BOND_CREATION_DATE,
      note: ARCHIVAL_BOND_ADDITIONAL_INFORMATION,
      records:[
            RECORD_ID
            ]
}
```

Figure 6. MongoDB-based data model of an archival bond object.

Having both embedded archival bond information and a dedicated collection grants greater flexibility when deciding which type to use. Indeed, since there is no limit to the number of embedded records both methods of storing archival bond information can be used for the same object. Using the two collections presented in Figures 4 and 6 we can achieve easy searchability of the blockchain, no matter the starting point—record or archival bond.

If the example record shown in Figure 5 required an archival bond to multiple objects, which were possibly created after the example record was created, a separate collection for the archival bond in question can be created. An example of such a collection is presented in Figure 7.

```
{
      _id: 2801,
      type: "contract",
      name: "NA Shipping contracts",
      date: "<2021-01-17>",
      note: "contracts relating to NA Shipping",
      records:[
            451281, 567321, 889471, 891253
            ]
}
```

Figure 7. MongoDB-based archival bond object example.

In addition to this, a document can be added to the "arcBond" embedded collection shown in Figure 5 which will reference this new collection. Such an addition would enable an efficient search for all participating records regardless of the search starting point (an individual record or the archival bond object).

7. Apache Cassandra-Based Data Model

The Apache Cassandra-based system uses tables. However, unlike relational database management system (RDBMS) solutions, there are no relationships between tables and each table corresponds to a query. Because of this, modelling such a system without concrete use case requirements is difficult. One use case may require queries encompassing only digital signature information, while the next one may require only general metadata and archival bond information. However, several query tables may be modelled which are guaranteed to be used. Three such tables are proposed (Figure 8), which roughly correspond to the logical model entities with a few additions—most importantly, the addition of "name" (of record) and "signer" to all three query tables.

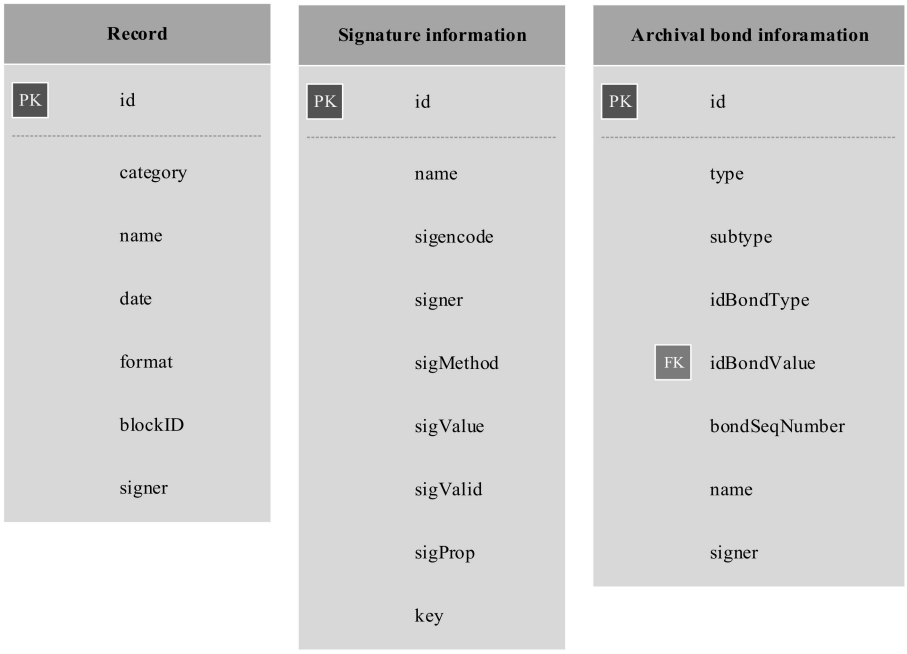

Figure 8. Apache Cassandra-based data model (PK = Primary Key, FK = Foreign Key).

The primary key (PK) and foreign key (FK) values shown in Figure 8 do not represent actual primary and foreign keys, in the meaning of the terms in the context of RDBMS, since Apache Cassandra does not enforce referential integrity. However, they do mark the primary ID of a record (PK) and, in the case of archival bond information, a borrowed ID (FK)—either an ID of a different record or an ID of an archival bond. The query tables shown in Figure 8 directly correspond to the MongoDB record information collection shown in Figure 4. Although an archival bond information query table is shown without additional information, it corresponds to the one-to-one relations as described in the model shown in Figure 3. Therefore, an additional table is designed to store archival bond information which can be used as a starting point in searches. The Apache Cassandra table shown in Figure 9 corresponds to the earlier MongoDB collection shown in Figure 6.

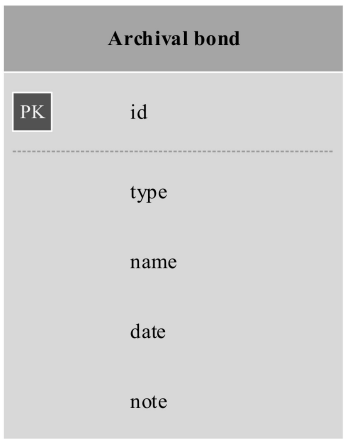

Figure 9. Apache Cassandra-based data model's archival bond query table.

8. Conclusions

Digital archives preserving digitally signed and/or sealed records are facing challenges related to validity expiration of signing certificates used in digital signatures. This threatens the ability of digital archives to confirm trustworthiness of digital records, i.e., their accuracy, reliability, and authenticity. We have, with other co-authors, already ad-

dressed the identified digital preservation challenge, firstly in 2017 when we developed the TrustChain model registering digital signatures' validity in the blockchain and enabling its validation even after expiration of signing certificates [2], and again in 2020 when we upgraded the model with the aspect of certification chain preservation functionality enabling digital signature validation even if the record being ingested in the digital archive has already expired certificates [3]. Until this paper, the challenge of digital certificates' validity information preservation using the immutable and append-only blockchain and distributed ledger technology (DLT) structure was missing the possibility to engage in management and preservation of the archival bond. Here, the concept of the archival bond is explained as a record's relationship network connecting it with other records in the same aggregation. It was pointed out that records' documentary context is changing when the records are active or semi-active, thus resulting with changes in the archival bond which becomes defined and stabilized when records become inactive. This results in the need to manage archival bond information of the records already registered in the blockchain's immutable structure. To that end, in this paper we propose the TrustChain supporting system model.

Firstly, following the results of comparative analysis of metadata standards, we have based the supporting system's metadata model on the PREMIS metadata standard. The metadata model we are proposing uses additional metadata fields to make it compatible with Dublin Core and ISAD(G) standards. Secondly, comparative analysis of blockchain-based data storage systems identified the two most suitable, although very different, systems that might be used. Therefore, the data model was developed as suitable for use in both MongoDB and Apache Cassandra-based systems, demonstrating how the archival bond could be realised and managed in parallel with the records registered in the blockchain. Although it might be argued that the solution we are proposing is similar to various blockchain explorers, it is developed as part of the TrustChain system, and by using it there is no need to explore blockchain data since searchable metadata is entered into the supporting database at the time of block creation. Finally, the proposed system combines the immutable blockchain structure, used for registering digital signatures' validity information, with the realization of the archival bond in the TrustChain supporting system.

To conclude, we consider the proposed approach in digital archive development and realisation of long-term digital preservation as an important step towards embracing the potential of new technologies, such as blockchain and DLT, while still honouring the foundational archival science concepts such as the archival bond.

The TrustChain VIP (validity information preservation) solution, incorporating the aspects of initial and upgraded models as well as the supporting system presented in this paper, is under development. The project is currently in an early stage of development—the system is still being designed. The next step in the TrustChain development will start after the prototype system is finished. An early prototype will allow easy evaluation of the system functionality by archival institutions. This also presents an opportunity for these archival institutions to embrace the idea and participate in the establishing of a network of trusted institutions, becoming trusted nodes of the TrustChain solution. Evaluating performance and scalability is more complicated but the two, at least in the case of the supporting system discussed here, are far from a complete unknown. Both MongoDB and Apache Cassandra are industry standard solutions for large-scale distributed database systems. Nevertheless, performance and scalability will be rigorously tested when TrustChain solution reaches that stage of development.

Author Contributions: Conceptualization, H.S. and V.B.; methodology, H.S. and V.B.; writing—original draft preparation, H.S. and V.B.; writing—review and editing, H.S. and V.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Batista, D.; Kim, H.; Lemieux, V.L.; Stančić, H.; Unnithan, C. Blockchains and Provenance: How a Technical System for Tracing Origins, Ownership and Authenticity Can Transform Social Trust. In *Building Decentralized Trust: Multidisciplinary Perspectives on the Design of Blockchains and Distributed Ledgers*; Lemieux, V.L., Feng, C., Eds.; Springer: Cham, Switzerland, 2021; pp. 111–128. [CrossRef]
2. Bralić, V.; Kuleš, M.; Stančić, H. A model for long-term preservation of digital signature validity: TrustChain. In *Integrating ICT in Society*; Atanassova, I., Zaghouani, W., Kragić, B., Aas, K., Stančić, H., Seljan, S., Eds.; University of Zagreb: Zagreb, Croatia, 2017; pp. 89–113. Available online: https://www.researchgate.net/publication/321171227_A_Model_for_Long-term_Preservation_of_Digital_Signature_Validity_TrustChain (accessed on 2 July 2021).
3. Bralić, V.; Stančić, H.; Stengård, M. A blockchain approach to digital archiving: Digital signature certification chain preservation. *Rec. Manag. J.* **2020**, *30*, 345–362. [CrossRef]
4. Duranti, L. The archival bond. *Arch. Mus. Inform.* **1997**, *11*, 213–218. [CrossRef]
5. ICA. Multilingual Archival Terminology. Available online: http://www.ciscra.org/mat/ (accessed on 2 July 2021).
6. Duranti, L.; MacNeil, H. The protection of the integrity of electronic records: An overview of the UBC-MAS research project. *Archivaria* **1996**, *42*, 46–67.
7. Lemieux, V.L.; Sporny, M. Preserving the Archival Bond in Distributed Ledgers: A Data Model and Syntax. In Proceedings of the 26th International Conference on World Wide Web Companion, Geneva, Switzerland, 3–7 April 2017; pp. 1437–1443. [CrossRef]
8. Merkle, R.C. A certified digital signature. In Proceedings of the Conference on the Theory and Application of Cryptology, New Yok, NY, USA, 20–24 August 1989. Available online: https://www.researchgate.net/publication/221355342_A_Certified_Digital_Signature (accessed on 2 July 2021).
9. Galiev, A.; Prokopyev, N.; Ishmukhametov, S.; Stolov, E.; Latypov, R.; Vlasov, I. Archain: A novel blockchain based archival system. In Proceedings of the 2018 Second World Conference on Smart Trends in Systems, Security and Sustainability, London, UK, 30–31 October 2018. Available online: https://arxiv.org/ftp/arxiv/papers/1901/1901.04225.pdf (accessed on 2 July 2021).
10. Permatasari, I.; Essaid, M.; Kim, H.; Ju, H. Blockchain Implementation to Verify Archives Integrity on Cilegon E-Archive. *Appl. Sci.* **2020**, *10*, 2621. [CrossRef]
11. Bandara, E.; Liang, X.; Shetty, S.; Ng, W.K.; Foytik, P.; Ranasinghe, N.; Zoysa, K.D.; Langöy, B.; Larsson, D. Lekana-Blockchain Based Archive Storage for Large-Scale Cloud Systems. In Proceedings of the International Conference on Blockchain, Honolulu, HI, USA, 18–20 September 2020. Available online: https://www.researchgate.net/publication/344372675_Lekana_-_Blockchain_Based_Archive_Storage_for_Large-Scale_Cloud_Systems (accessed on 2 July 2021).
12. McConaghy, T.; Marques, R.; Müller, A.; Jonghe, D.D.; McConaghy, T.; McMullen, G.; Henderson, R.; Bellemare, S.; Granzotto, A. Bigchaindb: A Scalable Blockchain Database. Available online: https://gamma.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf (accessed on 2 July 2021).
13. Muzammal, M.; Qu, Q.; Nasrulin, B. Renovating blockchain with distributed databases: An open source system. *Future Gener. Comput. Syst.* **2019**, *90*, 105–117. [CrossRef]
14. Helmer, S.; Roggia, M.; el Ioini, N.; Pahl, C. Ethernitydb–integrating database functionality into a blockchain. In Proceedings of the European Conference on Advances in Databases and Information Systems, Budapest, Hungary, 2–5 September 2018; pp. 1–8. Available online: https://www.researchgate.net/publication/327309357_EthernityDB_-_Integrating_Database_Functionality_into_a_Blockchain (accessed on 2 July 2021).
15. Bandara, E.; Ng, W.K.; de Zoysa, K.; Fernando, N.; Tharaka, S.; Maurakirinathan, P.; Jayasuriya, N. Mystiko—blockchain meets big data. In Proceedings of the 2018 IEEE International Conference on Big Data, Seattle, WA, USA, 10–13 December 2018; pp. 3024–3032. [CrossRef]
16. Benet, J. IPFS-Content Addressed, Versioned, P2P File System. 2014. Available online: https://arxiv.org/pdf/1407.3561.pdf (accessed on 2 July 2021).
17. Naz, M.; Al-zahrani, F.A.; Khalid, R.; Javaid, N.; Qamar, A.M.; Afzal, M.K.; Shafiq, M. A secure data sharing platform using blockchain and interplanetary file system. *Sustainability* **2019**, *11*, 7054. [CrossRef]
18. Lakshman, A.; Malik, P. Cassandra: A decentralized structured storage system. *Oper. Syst. Rev.* **2010**, *44*, 35–40. [CrossRef]
19. MongoDB Inc. MongoDB Documentation. Available online: https://docs.mongodb.com/ (accessed on 2 July 2021).
20. Abramova, V.; Bernardino, J. NoSQL databases: MongoDB vs. Cassandra. In Proceedings of the international C* Conference on Computer Science and Software Engineering, New York, NY, USA, 10–12 July 2013. [CrossRef]
21. MongoDB Inc. JSON and BSON. Available online: https://www.mongodb.com/json-and-bson (accessed on 2 July 2021).
22. Society of American Archivists. Describing Archives: A Content Standard (DACS). 2021. Available online: https://www2.archivists.org/groups/technical-subcommittee-on-describing-archives-a-content-standard-dacs/describing-archives-a-content-standard-dacs-second- (accessed on 2 July 2021).
23. Brothman ISAD(G): General International Standard Archival Description. *Archivaria* **1992**, *34*, 17–32.
24. Caplan, P. Understanding PREMIS, Washington DC, USA: Library of Congress. 2009. Available online: https://www.loc.gov/standards/premis/understanding-premis.pdf (accessed on 2 July 2021).

25. Weibel, S.; Kunze, J.; Lagoze, C.; Wolf, M. Dublin Core Metadata for Resource Discovery. 1998. Available online: http://www.hjp.at/doc/rfc/rfc2413.html (accessed on 2 July 2021).
26. PREMIS Editorial Committee. PREMIS Data Dictionary for Preservation Metadata. 2015. Available online: https://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf (accessed on 2 July 2021).