https://dspace.org/

A Supplementary Tool for Web-archiving Using Blockchain Technology

De Villiers, John E.; Calitz, André P.

2020-06-30
LINK Centre, University of the Witwatersrand (Wits), Johannesburg
https://hdl.handle.net/10539/29194
De Villiers, J. E., & Calitz, A. (2020). A supplementary tool for web-archiving using blockchain technology. The African Journal of Information and Communication (AJIC), 25, 1-14. https://doi.org/10.23962/10539/29194

Downloaded from a DSpace repository. https://dspace.org/

A Supplementary Tool for Web-archiving Using Blockchain Technology

John E. de Villiers
Master's Student, Department of Computing Sciences, Nelson Mandela University, Port Elizabeth, South Africa
https://orcid.org/0000-0002-9146-8180

André P. Calitz
Professor, Department of Computing Sciences, Nelson Mandela University, Port Elizabeth, South Africa
https://orcid.org/0000-0002-2555-9041

Abstract
The usefulness of a uniform resource locator (URL) on the World Wide Web is reliant on the resource being hosted at the same URL in perpetuity. When URLs are altered or removed, this results in the resource, such as an image or document, being inaccessible. While web-archiving projects seek to prevent such a loss of online resources, providing complete backups of the web remains a formidable challenge. This article outlines the initial development and testing of a decentralised application (DApp), provisionally named Repudiation Chain, as a potential tool to help address these challenges presented by shifting URLs and uncertain web-archiving. Repudiation Chain seeks to make use of a blockchain smart contract mechanism in order to allow individual users to contribute to web-archiving. Repudiation Chain aims to offer unalterable assurance that a specific file and its URL existed at a given point in time—by generating a compact, non-reversible representation of the file at the time of its non-repudiation. If widely adopted, such a tool could contribute to decentralisation and democratisation of web-archiving.

Keywords
web-archiving, blockchain, trusted timestamping, non-repudiation, cryptographic hash functions, Merkle trees, DApps, smart contracts

DOI: https://doi.org/10.23962/10539/29194

Recommended citation
De Villiers, J. E., & Calitz, A. (2020). A supplementary tool for web-archiving using blockchain technology. *The African Journal of Information and Communication (AJIC), 25*, 1-14. https://doi.org/10.23962/10539/29194

This article is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence: https://creativecommons.org/licenses/by/4.0

1. Introduction

The ubiquity of the internet-enabled World Wide Web has allowed for its use as a global storage platform for data and information (Berners-Lee & Fischetti, 2001). The preservation of web content is of immeasurable value, and tools and services that can aid in this preservation need to be continually researched and explored. Web resources are linked to uniform resource locators (URLs), but the permanence of URL citations is uncertain and dependent on several factors, including websites' management policies and their archiving and hosting agreements. When a URL changes or is deleted, the associated resource, such as a web page, a document, an image, a photo or a video, will become either difficult to access or completely inaccessible.

Web-archiving initiatives seek to prevent losses of online resources due to URL changes or deletions, thus playing an important role in the preservation of data and information for future researchers and the broader public. However, it is misguided to assume the completeness and perpetual existence of web archives, such as the Wayback Machine (Habibzadeh, 2013; Internet Archive, n.d.; Murphy, Hashim, & O'Connor, 2008). Challenges facing web archives include the apparent lack of a complete, unbiased web archive across the entire web, and the archives' obligations to remove copyright-infringing content.

The purpose of our study was to perform the initial development and testing of a decentralised application (DApp) that could be made publicly available as a non-repudiation tool for use by individuals and entities seeking to participate in web-archiving. The DApp that we began development of, which we have named Repudiation Chain, is designed to provide a means of storing pertinent metadata relating to any given URL, by using the Ethereum cryptocurrency's blockchain smart contract tool. Repudiation Chain seeks to provide non-repudiation properties, i.e., guarantees that, at a specific point in time, the integrity or contents of a file linked to a URL can be established as an immutable representation of that file's existence. Repudiation Chain seeks to provide proof of the origin and integrity of the data linked to a URL; to prevent participants from denying responsibility for actions they take, such as altering a file linked to a URL; and to preserve information on the entity requesting the immutable representation.

Repudiation Chain is designed in such a way that, upon request by an entity, such as an internet user, the tool is required to store, in a blockchain, a time-stamped, immutable, compact, and public representation of a file that has been given a URL. This stored representation of the file must be associated with the entity that originally made the request for storage, as well as the URL that points to the file location. Repudiation Chain must also be able to determine whether there is a difference between the file that has been stored in the blockchain and the file identified by the URL.

Repudiation Chain is thus designed to be a supplementary web-archiving tool that can be used in conjunction with a web-archive initiative. It is intended for use by those who need access to a free notary-like tool, and those who wish to track whether a specific file hosted online has changed over time. Repudiation Chain thus aims to supplement, and address some flaws inherent in, existing internet-archiving services, as described in the introduction. It also aims to help decentralise and democratise web-archiving, by allowing individual users and small entities to participate in, and increase the accuracy of, web-archiving.

2. Literature review

Web-archiving

Web-archiving can be divided into two categories:

- *micro-archiving*, performed on a small scale (including by individuals) for day-to-day purposes and aiming to preserve an object for further study; and
- *macro-archiving*, performed on a large scale by organisations with specialised technical expertise in the field for preservation purposes (e.g., of societal and cultural heritage) and to serve as primary sources of historic data and information.

Web-archiving services are provided by trusted third parties, and the services' basic operations are (ITU, 2000) storing of documents, and issuing of signed copies of stored documents, when requested by an authorised entity, with inclusion of the documents' dates of registration.

Distributed ledger systems and blockchains

A distributed ledger system is a database distributed across multiple hosts on a network. It is typically public and requires some consensus among contributors operating on the system so that it can synchronise, share, and replicate itself. One of the more popular types of distributed ledger system is a blockchain (UK Government Chief Scientific Adviser, 2016). A blockchain platform is an open, decentralised ledger that can record transactions between parties across a peer-to-peer network, without the need for a central certifying authority. Blockchain applications are increasingly being developed for services such as fund transfers, smart contracts, e-voting, and efficient supply chain management. Blockchains can now be found deployed in numerous sectors, including finance, healthcare, law, trade, and real estate.

A blockchain is a data structure consisting of a list of data records, called blocks, which are cryptographically and sequentially linked. A block contains and stores the following data (Conte de Leon, Stalick, Jillepalli, Haney, & Sheldon, 2017):

- *Block data:* the set of messages or transactions, analogous to a collection of records;

- *Chaining hash values:* the hash values of the previous block in the blockchain; and
- *Block hash*: a representation of a given block's content as well as its chaining hash.

The existence and authenticity of a block are reliant on the block's cryptographic signature. This signature is composed of a timestamp, the hash value of the given block, and the hash value of the preceding block—unless the block is the genesis block (the first block in a blockchain).

Blockchains provide certain strengths and advantages over other systems, including the following:

- they do not require a central authority and thus they move trust requirements to the underlying technology (Conte de Leon et al., 2017; Walch, 2017);
- in a publicly available distributed ledger, all participants can make transactions, thus providing transparency (UK Government Chief Scientific Adviser, 2016); and
- all blocks added are validated and distributed across the network to form an emergent blockchain (and not all participants will have the same blockchain, thus making the emergent blockchain the one that the majority of the network participants agree upon at any given point in time) (Conte de Leon et al., 2017).

Two notable weaknesses of blockchains are:

- the security and immutability of a blockchain rely on the strength of the cryptographic hash function used (Conte de Leon et al., 2017); and
- they currently lack sufficiently widespread adoption.

Trusted timestamping and non-repudiation

A digital trusted timestamp, analogous to a traditional physical timestamp, serves as proof that the timestamped data or information existed at the time indicated by the stamp. A trusted timestamp is one that cannot be tampered with and, accordingly, is one provided by a service provider whose validity is trusted by all involved parties (Truu, 2010; Adams, Pinkas, Cain, & Zuccherato, 2001). In trusted timestamping, the creation and modification times are stored. This requires the use of publicly available, trusted, timestamp management infrastructure in the form of a timestamping authority. Trusted timestamps are particularly important for cryptocurrency ledgers such as Bitcoin, as well as for anything where reliable proof is needed of data or information's existence in a particular form at a particular time (Nakamoto, 2008).

A topic closely related to trusted timestamping is non-repudiation. A general definition of a system that provides non-repudiation is a system where a user cannot deny actions or refuse accountability (Negus, 2020). In the context of a cryptographic

service, non-repudiation is a form of authentication that provides proof of the integrity and origin of a given file, with proof being unforgeable and verifiable by any third party at any time (McCullagh & Caelli, 2000).

In Repudiation Chain, the smart contract used can be extended to provide a non-repudiation service, but the primary property it seeks to provide, in the tool's current stage of development, is immutable proof that a given URL-linked file existed at a specific point in time in a specific form.

Cryptographic hash functions and Merkle trees

A hash function maps an input of arbitrary length to an output of fixed length (Goodrich & Tamassia, 2014). Hash functions have many functional properties, which have led to their widespread use as one of the underlying means of security for computing devices, online communication, and the internet (Merkle, 1989; Preneel, 2011). Many different hash functions have emerged, including the family of secure hash algorithms (SHAs), which are extensively used in blockchain implementations and other cryptographically sensitive applications. The contents of a typical blockchain can be altered only if enough computational power has reached consensus on the alteration, and the computational power required is heavily dependent on the strength of the hash function (the hash function's resilience to different types of attacks) used by the blockchain.

A Merkle tree is a data structure that is dependent on hash functions (Berman, Karpinski, & Nekrich, 2007). Merkle trees can be used to provide a representation of data, and to provide a secure means of verifying the contents of a data file (Nakamoto, 2008; Singhal, Dhameja, & Panda, 2018). Every leaf value in a Merkle tree can be identified with respect to a publicly known root and the authentication path of that leaf. A Merkle tree is built from its leaves to its root so that the root can be used as a representation of the entire data file. Repudiation Chain uses a Merkle tree to represent data files of varying length in a compact manner.

Ethereum DApps and smart contracts

Ethereum, which supports the Ether cryptocurrency, uses a blockchain, and it provides a platform for DApps and smart contracts (Buterin, 2014). DApps provide an interface between smart contracts and the Ethereum Virtual Machine (EVM) where smart contracts are executed. A smart contract is a self-executing, distributed code. Executing a smart contract accrues a "gas" cost, which represents the amount of computational work required. The price of gas is proportional to the priority that the EVM gives to executing a smart contract, based on block "miners" prioritising contracts that are seen as profitable. A miner is a person using computing technologies, such as the Antminer S9 mining hardware, to process blocks of transactions, such as

Bitcoin transactions, and add the blocks to the blockchain (Hertig, 2017). Table 1 provides approximate gas costs, in 2020, for some operations (Wood, 2020).

Table 1: Prices of some operations in an Ethereum smart contract

Operation	Gas
adding two numbers	3 gas
multiplying two numbers	5 gas
secure hash algorithm 3 (SHA3) operation	30 gas
cost per word of input for SHA3 operation	6 gas
storing one byte of non-zero transaction data	68 gas

3. Initial development and testing of Repudiation Chain

To conduct the initial development work on Repudiation Chain, we worked with the aforementioned EVM smart contract tool provided by Ethereum. The metadata components that needed to be stored were:

- an immutable representation of the URL-linked file;
- a timestamp that is separate from the one provided by Ethereum and is expensive to alter, thus providing additional protection against alteration;
- the URL in question; and
- an association to the entity making use of the service.

More trustworthy timestamping

Several different kinds of attacks exist for Ethereum smart contracts. One is based on an unmined block's timestamp. If a smart contract uses a block's timestamp, it is possible for a miner who holds some incentive to manipulate the given contract's functionality to tamper with the timestamp for a block that the miner mines. This vulnerability has been reduced to a few seconds, but it still presents a potential weakness (Atzei, Bartoletti, & Cimoli, 2017).

Accordingly, since Repudiation Chain relies on a trusted timestamp, it was determined that the native timestamp provided for each block by Ethereum was inadequate. An alternative means of providing a trusted timestamp, or of providing a fail-safe method to detect possible tampering, needed to be identified. To this end, we designed Algorithm 1. Algorithm 1 provides a simple means of identifying whether a list of timestamps deviates by more than some predetermined length of time in a list of timestamps. The acceptable time deviation can be altered to conform to the specifics of the required scenario. This determination of an acceptable time deviation would need to be based on a consideration of the amount of time it takes to get a timestamp, the size of the list of timestamps, whether getting a timestamp can be parallelised (i.e., via multiple timeservers operating in a near-simultaneous fashion), the amount of time it takes for a contract to be executed, the transaction fee, and the source of the timestamps.

Algorithm 1: Detect suspicious timestamp

```
function isSuspiciousTimestamp(ListofTimestamps, TimeDeviation) : returns Boolean
    Long averageTimestamp ← Mean(ListofTimestamps))
    for each timestamp in ListofTimestamps do
        if ((timestamp + TimeDeviation) > averageTimestamp) or
                ((timestamp − TimeDeviation) < averageTimestamp) then
                    return true
        end if
    end for
    return false
end function
```

If Algorithm 1 returns a value of false, then the average timestamp of the list of timestamps can be used in the smart contract. This provides an emergent, fail-safe timestamp. This fail-safe timestamp indicates that the participants external to the miner came to consensus regarding a time. Thus, this fail-safe can be used in conjunction with the block's timestamp, with any significant difference between the two indicating that either the miner or the provider of the list of timestamps tampered with the timestamps provided.

Algorithm 1 only provides a means of detecting whether a timestamp was tampered with. To further obscure the possibility or risk of tampering, the list of timestamps should be from multiple, geographically dispersed sources, i.e., to reduce the possibility of collusion between possible timestamp providers. The derived average timestamp should be presented to the smart contract only once it satisfies Algorithm 1. This average timestamp is presented to the smart contract only when consensus is reached on the client's side. This results in the miner having a diminished window in which to tamper, without being detected, with the block's timestamp or the underlying smart contract's functionality. This process leads to the improvement of a smart contract's usefulness as a means of providing trusted timestamping. Examples of possible sources for lists of timestamps include the NTP Pool Project and Google's Public NTP.

Ensuring secure hash algorithms and application

Repudiation Chain is designed to use hash functions to provided file identification and integrity—as the hash value of any file can be used to identify that file, and any changes made to that file will result in a change to the file's hash value (Mackall, 2006).

Smart contracts on Ethereum should not be used to find multiple hash values, as hashing is an expensive operation in Ethereum (Wood, 2014). An alternative to using a smart contract to find the hash value of a file is to find the hash value on the client's side (i.e., on the side of the entity using Repudiation Chain). But finding

the hash value of a large file can fail due to hardware limitations, as the entirety of a large file cannot "fit" in the memory of the machine when calculating the hash value. Algorithm 2 was designed to overcome this practical limitation by partitioning the given file into a list of m-partitions, iterating over the list, and calculating the hash value of the entire file in a linear manner. Algorithm 3 provides a generalisation of Algorithm 2 for application to a list.

Algorithm 2: Naïve file hash algorithm used by Repudiation Chain

```
function NaiveFileHash(File, HashFunction, BufferSize) : returns FileHash
    FileHash HashValue
    while (entire File not read) do
        curBuffer ← File.readNextBuffer(BufferSize)
        HashValue ← HashFunction(output + curBuffer)
    end while
    return HashValue
end function
```

Algorithm 3: Data structure devised from Algorithm 2 for direct comparison to Merkle trees

```
function HashLinkedChain(ListofItems) : returns HashValue
    HashValue current
    for each Items in ListofItems do
        current ← Hash(current + Items)
    end for
    return current
end function
```

We conducted an ANOVA statistical comparison between Algorithm 3 and Merkle trees (using a binary Merkle tree), whereby the null hypothesis was set as: *the mean performance of either algorithm is identical*. This comparison was based on the time required to generate a representation of files of varying lengths when using either algorithm. Algorithm 3 was found to perform significantly more quickly than the Merkle tree, at a p-value less than 0.00000001. (In lay terms, the p-value is the probability of obtaining the observed results when the associated null hypothesis is true, with accordingly smaller p-values being preferable (see Mendenhall & Sincich, 2012).) This difference in performance was likely due to Algorithm 3 calculating roughly half as many hash values as a Merkle tree for the same file. Algorithm 3 does not have the same authentication path properties as a Merkle tree, but due to Algorithm 3's superior performance for the desired use case, it is the preferred means of generating hash value representations of files. The sacrifice Algorithm 3 makes in order to achieve greater speed than a Merkle tree is a loss of functionality that was not needed by Repudiation Chain.

4. Comparison between Repudiation Chain and Bitcoin's Proof of Existence

To test Repudiation Chain, we compared its efficacy to that of Bitcoin's Proof of Existence non-repudiation service, which operates as a web application (Crosby, Nachiappan, Pattanayak, Verma, & Kalyanaraman, 2016; Proof of Existence, n.d.). Proof of Existence provides non-repudiation by storing the cryptographic hash of a file as part of a transaction on Bitcoin's blockchain, with each transaction having an associated timestamp (Crosby et al., 2016). Users of Proof of Existence upload the file they wish to non-repudiate and pay the associated fee of 0.00025 Bitcoins (BTC), which, at the time of the finalisation of this article in June 2020, had a value of USD2.34 (based on an exchange rate, calculated via CoinMarketCap, of BTC1=USD9,349.75 and ETC1=228.80 (CoinMarketCap, 2020)). Proof of Existence is a centralised web DApp, unlike Repudiation Chain, which is a smart contract that interacts with a client-side application. Unlike Proof of Existence, Repudiation Chain facilitates the transfer of trust to the client's hardware as opposed to a centralised third-party's hardware. The cost to use Proof of Existence is set at a flat flee, and this fee is larger than the cost to store the appropriate information, with this disparity presumably being necessary to pay for infrastructure costs and to maintain a profit. Repudiation Chain does not share the same cost burdens and possible profit motivations.

In order to compare this USD2.34 cost to use Proof of Existence with the cost to use Repudiation Chain, we derived a function that could be used to estimate the gas cost of Repudiation Chain use, as follows:

> For an adjusted R^2 value of 0.9997806 for a p-level less than 0.00001, the gas cost to use the RC is given by: $(695.81*x_1) + 210598.31$; where x_1 is the length of the URL. The lower bound of a URL's length is 13 characters, the mean is 35 characters and the upper bound is 425 (Ducut, Liu, & Fontelo, 2008).

Table 2 provides a summary of the expected costs associated with using Repudiation Chain.

Table 2: Repudiation Chain gas cost estimates

URL length	Gas cost	Gas price (Wei)	Transaction fee (Ether)	USD value (June 2020)
13	219643.84	4.92e+15	4.92e-03	USD1.13
35	234951.66	5.26e+15	5.26e-03	USD1.20
425	506317.56	1.13e+16	1.13e-02	USD2.59

Table 3 compares Repudiation Chain and Proof of Existence in terms of cost, how files are received, information that is stored, and anonymity.

Table 3: Comparison between Repudiation Chain and Proof of Existence

Factors	Repudiation Chain	Proof of Existence
Cost of service	Dependent on length of URL (see Table 2)	Flat fee of BTC0.00025
Approximate cost to add non-repudiated representation of a file	USD1.13 – USD2.59	USD1.97
How files are received	Download	Upload
Information stored	two distinct and independent timestamps hash of file hash of cargo address of the adder URL of file	timestamp hash of file
Anonymity	Derived from Ethereum's and device's anonymity	Derived from Bitcoin's, browser's and device's anonymity

As seen in Table 3, our proposed Repudiation Chain was found to provide more functionality and information storage than Proof of Existence, and to be cheaper to use for many URL lengths.

5. Directions for further testing and development of Repudiation Chain

The next key step will be extensive testing of the functionality and scalability of Repudiation Chain and its underlying system on the main Ethereum network. Additionally, there are several possible additional elements of development. Repudiation Chain could utilise available processing cores more effectively through the parallelisation of Algorithms 1 and 2 for timestamping and hash representation, potentially resulting in shorter allowed time deviations. Repudiation Chain could also be redesigned to incorporate a "token" or alt-currency system, which would allow for donation of funds (through a publicly accessible wallet) in support of Repudiation Chain's functioning, thus extending its democratisation. Such functionality would,

however, likely require investigation of cost-effective spam-filtering.

The functionality of Repudiation Chain could also be extended to provide additional front-end options for users, in the form of a web application or an application for smartphone devices (e.g., DApp, which would be very similar to a standard mobile application, but with some important computations and storage functionality handled by the Ethereum blockchain). Moreover, additional layers of anonymity could also be explored, in order to further protect the interests and privacy of users. This might include devising a means of "laundering" transactions between Ethereum smart contracts and Repudiation Chain, i.e., by obfuscating the origins of a given transaction or interaction. Consideration could also be given to the addition of a means to manipulate the speed at which information is stored by Repudiation Chain—e.g., by allowing users to define the gas price relative to the current market, thus improving the user experience.

References

Adams, C., Pinkas, D., Cain, P., & Zuccherato, R. J. (2001). *Internet X.509 public key infrastructure time-stamp protocol (TSP).* The Internet Society. https://doi.org/10.17487/rfc3161

Ainsworth, S. G., AlSum, A., SalahEldeen, H., Weigle, M. C., & Nelson, M. L. (2011). How much of the web is archived? In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries* (pp. 133–136). Association for Computing Machinery (ACM). https://doi.org/10.1145/1998076.1998100

AlNoamany, Y., AlSum, A., Weigle, M. C., & Nelson, M. L. (2014). Who and what links to the internet archive. *International Journal on Digital Libraries, 14*, 101–115. https://doi.org/10.1007/s00799-014-0111-5

Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts. In M. Maffei & M. Ryan (Eds.), *Principles of Security and Trust: 6th International Conference (POST 2017)* (pp. 164–186). https://doi.org/10.1007/978-3-662-54455-6_8

Bayardo, R. J., & Sorensen, J. (2005). Merkle tree authentication of HTTP responses. In *WWW '05: Special interest tracks and posters of the 14th International Conference on World Wide Web* (pp. 1182–1183). Association for Computing Machinery (ACM). https://doi.org/10.1145/1062745.1062929

Becker, G. (2008). *Merkle signature schemes, Merkle trees and their cryp*tanalysis. Ruhr-Universität Bochum, Germany. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.392.7879&rep=rep1&type=pdf

Berman, P., Karpinski, M., & Nekrich, Y. (2007). Optimal trade-off for Merkle tree traversal. *Theoretical Computer Science, 372*, 26–36. https://doi.org/10.1016/j.tcs.2006.11.029

Berners-Lee, T., & Fischetti, M. (2001). *Weaving the Web: The original design and ultimate destiny of the World Wide Web.* New York: HarperCollins. https://doi.org/10.5860/choice.37-3934

Buterin, V. (2014). Ethereum white paper: A next generation smart contract & decentralized application platform. Retrieved from https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf

Chohan, U. W. (2017). Cryptocurrencies: A brief thematic review. SSRN. https://dx.doi.org/10.2139/ssrn.3024330

CoinMarketCap. (2020). All cryptocurrencies. Retrieved 25 June 2020 from https://coinmarketcap.com/all/views/all/

Conte de Leon, D., Stalick, A. Q., Jillepalli, A. A., Haney, M. A., & Sheldon, F. T. (2017). Blockchain: Properties and misconceptions. *Asia Pacific Journal of Innovation and Entrepreneurship*, *11*(3) 286–300. https://doi.org/10.1108/apjie-12-2017-034

Crosby, M., Nachiappan, Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond Bitcoin. *Applied Innovation Review*, *2*, 6–19.

Ducut, E., Liu, F., & Fontelo, P. (2008). An update on uniform resource locator (URL) decay in MEDLINE abstracts and measures for its mitigation. *BMC Medical Informatics and Decision Making*, *8*, 23. https://doi.org/10.1186/1472-6947-8-23

Duranti, L. (1992). Origin and development of the concept of archival description. *Archivaria 35*, 47–54.

Dworkin, M. J. (2015). *SHA-3 standard: Permutation-based hash and extendable output functions.* Federal Information Process Standards Publication 202, NIST. https://doi.org/10.6028/nist.fips.202

Etherscan. (2020). Ethereum average gas price chart. Retrieved 25 June 2020 from https://etherscan.io/chart/gasprice

Everett, S., Calitz, A. P., & Greyling, J. (2017). The case for a 'sovereign' distributed securities depository for securities settlement. *Journal of Securities Operations & Custody*, *9*(3), 269–292.

Goodrich, M. T., & Tamassia, R. (2014). *Algorithm design and applications.* Hoboken, NJ: John Wiley & Sons.

Haber, S., & Stornetta, W. S. (1991). How to time-stamp a digital document. *Journal of Cryptology*, *3*, 99–111. https://doi.org/10.1007/bf00196791

Habibzadeh, P. (2013). Decay of references to web sites in articles published in general medical journals: Mainstream vs small journals. *Applied Clinical Informatics*, *4*(4), 455–464. https://doi.org/10.4338/aci-2013-07-ra-0055

Hertig, A. (2017). Ethereum 101: Chapter 5: How do Ethereum smart contracts work? CoinDesk. Retrieved from https://www.coindesk.com/learn/ethereum-101/ethereum-smart-contracts-work

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems, 19*(2), 4.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *Management Information Systems Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625

Internet Archive. (n.d.). [Website]. Retrieved from https://archive.org/

International Telecommunication Union (ITU). (2000). X. 842: Information technology: Security techniques: Guidelines for the use and management of trusted third party services. ITU-T Study Group 7, Telecommunication Standardisation Sector. https://doi.org/10.3403/02617626u

Masanés, J. (2006). Web archiving: Issues and methods. In *Web archiving* (pp. 1–53). Heidelberg: Springer. https://doi.org/10.1007/978-3-540-46332-0_1

Mackall, M. (2006). Towards a better SCM: Revlog and mercurial. In *Ottawa Linux Symposium 2* (pp. 83–90).

Manuel, S., & Peyrin, T. (2008). Collisions on SHA-0 in one hour. In Nyberg K. (Ed.), *Fast software encryption: FSE 2008* (pp 16-35). Lecture Notes in Computer Science, Vol. 5086. Berlin: Springer. https://doi.org/10.1007/978-3-540-71039-4_2

McCullagh, A., & Caelli, W. (2000). Non-repudiation in the digital environment. *First Monday, 5*(8). https://doi.org/10.5210/fm.v5i8.778

Mendenhall, W., & Sincich, T. (2012). *A second course in statistics: Regression analysis* (7th ed.). Saddle River, NJ: Prentice Hall.

Merkle, R. C. (1989). A certified digital signature. In G. Brassard (Ed.), *Advances in cryptology — CRYPTO' 89 proceedings* (pp. 218–238). Lecture Notes in Computer Science, Vol. 435. New York: Springer.

Murphy, J., Hashim, N. H., & O'Connor, P. (2008). Take me back: Validating the Wayback Machine. *Journal of Computer-Mediated Communication*, *13*(1), 60–75. https://doi.org/10.1111/j.1083-6101.2007.00386.x

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved from https://bitcoin.org/bitcoin.pdf

Naor, M., & Yung, M. (1989). Universal one-way hash functions and their cryptographic applications. In *STOC '89: Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing* (pp. 33–43). Association for Computing Machinery (ACM). https://doi.org/10.1145/73007.73011

Negus, C. (2020). *Linux bible* (10th ed.). Indianapolis: John Wiley & Sons.

Niels, B. (2011). Web archiving – Between past, present, and future. In C. Ess, & M. Consalvo (Eds.), *The handbook of internet studies* (pp. 24–42). Chichester, UK: John Wiley & Sons. https://doi.org/10.1002/9781444314861.ch2

Offermann, P., Levina, O., Schönherr, M., & Bub, U. (2009). Outline of a design science research process. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology.* Association for Computing Machinery (ACM). https://doi.org/10.1145/1555619.1555629

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/mis0742-1222240302

Preneel, B. (2011). Hash functions. In H. C. van Tilborg, & S. Jajodia (Eds.), *Encyclopedia of cryptography and security* (pp. 543–545). Berlin: Springer Science & Business Media.

Proof of Existence. (n.d.). [Website]. Retrieved from https://proofofexistence.com

Seo, J.-W., Kim, D.-K., Kim, H.-C., & Chung, J.-W. (2007). The algorithm of sharing incomplete data in decentralized P2P. *International Journal of Computer Science and Network Security*, *7*(8), 149–153.

Singhal, B., Dhameja, G., & Panda, P. S. (2018). *Beginning blockchain: A beginner's guide to building blockchain solutions.* Berlin: Apress.

Suzuki, K., Tonien, D., Kurosawa, K., & Toyota, K. (2006). Birthday paradox for multi-collisions. In M.S. Rhee, & B. Lee (Eds.), *Information security and cryptology – ICISC 2006* (pp. 29–40). Lecture Notes in Computer Science, Vol. 4296. Berlin: Springer. https://doi.org/10.1007/11927587_5

Thomsen, S. S., & Knudsen, L. R. (2009). *Cryptographic hash functions.* Technical University of Denmark, Kongens Lyngby.

Truu, A. (2010). *Standards for hash-linking based time-stamping schemes.* University of Tartu, Estonia.

Tsudik, G. (1992). Message authentication with one-way hash functions. In *IEEE INFOCOM '92: The Conference on Computer Communications* (pp. 2055–2059). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/infcom.1992.263477

UK Government Chief Scientific Adviser. (2016). *Distributed ledger technology: Beyond block chain.* London: Government Office for Science.

Vega-Redondo, F. (2003). *Economics and the theory of games.* Cambridge, UK: Cambridge University Press.

Walch, A. (2017). Open-source operational risk: Should public blockchains serve as financial market infrastructures? In D. L. Chuen, & R. H. Deng, (Eds.), *Handbook of blockchain, digital finance, and inclusion, volume 2* (pp. 243–269). Cambridge, MA: Academic Press. https://doi.org/10.1016/b978-0-12-812282-2.00011-5

Waldrop, M. M. (2016). The chips are down for Moore's law. *Nature*, *530*(7589), 144-147.

Wall, L. D. (2018). *Some blockchain challenges.* Atlanta: Federal Reserve Bank of Atlanta.

Wood, G. (2014). *Ethereum: A secure decentralised generalised transaction ledger.* Retrieved from https://gavwood.com/paper.pdf

Wood, G. (2020). *Ethereum: A secure decentralised generalised transaction ledger: Petersburg version.* Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf