Elias Rohrer, Steffen Heidel, Florian Tschorsch

Webchain: Verifiable Citations and References for the World Wide Web

Rohrer, Elias; Heidel, Steffen; Tschorsch, Florian (2018): Webchain: Verifiable Citations and References for the World Wide Web. BLOCKCHAIN '18: Proceedings of the IEEE International Conference on Blockchain. https://doi.org/10.1109/Cybermatics_2018.2018.00235

Webchain: Verifiable Citations and References for the World Wide Web

Elias Rohrer
Distributed Security Infrastructres
Technical University of Berlin

Steffen Heidel
Distributed Security Infrastructres
Technical University of Berlin

Florian Tschorsch
Distributed Security Infrastructres
Technical University of Berlin

Abstract—**Readers' capability to consider and assess sources is imperative. Digital preservation efforts, however, mostly neglected citation provenance, which is a necessity for transparent source verification. We therefore present Webchain, a new system enabling verifiable citations and references on the World Wide Web. Its architecture combines a distributed ledger with secure timestamping to ensure history of creation, ownership, and referential integrity of online resources. With Webchain, readers can independently detect content manipulation by verifying authenticity, integrity, and time consistency. At the same time, authors gain a proof of existence for referenced articles. Webchain extends a well-known distributed timestamping scheme to handle an open and dynamic network topology by providing a solution for membership management. We examine the security of our approach, particularly regarding forging attacks. Our results show that we are able to render such attacks infeasible, even in the face of a powerful attacker.**

I. Introduction

The verifiability of sources is, also beyond science and journalism, an essential method to substantiate arguments, give authors credit, and enable readers to reproduce conclusions. Traditionally, readers' ability to verify claims and citations depends on the respective author's careful handling of citations and references. Nevertheless, it also depends on the reader's capability to retrieve and check the referenced materials. Due to the fact that more and more information is published online, and given the highly volatile nature of digital information, it is currently still unclear how to deal with digital sources. It has been shown that loss of referential information is a real issue [6], [12], and recent events [11] indicate that even large publishers—which should provide a trust anchor—do not guarantee the availability of their online articles. While web archives such as the Wayback Machine [1] sustain availability of web pages, they do not address verifiability.

In this paper, we therefore propose *Webchain*, a verifiable citation system providing *citation provenance* for resources on the World Wide Web (WWW). Participating authors can use Webchain to cite other author's online articles and give readers the ability to independently verify the *authenticity*, *integrity*, and *time consistency* of resources and references. Hence, Webchain makes manipulation of cited content detectable and provides *non-repudiable* publication of articles. That is, Webchain does not only yield author attribution, it also provides authors referencing content the necessary data to prove that they referred to a certain "version" of an article.

In order to achieve these properties, Webchain's design is inspired by distributed timestamping [10] and blockchain-based systems like Bitcoin [15]. To this end, Webchain builds upon replicated data structures consisting of cryptographically interlinked blocks. Each block represents an article and is linked to all referenced articles, i. e., blocks. When new blocks are created, they are distributed in a network of infrastructure nodes that verify them and, if they are indeed valid, apply them to their local ledger. To enable a proof of existence of articles, i. e., a proof that articles existed at a certain point in time, the Webchain architecture incorporates a secure and distributed timestamping service. It ensures that blocks were first seen in a certain time interval, effectively inserting them into a total order over all blocks. As a consequence, re-dating of articles becomes infeasible.

Webchain employs an extended version of a well-known timestamping scheme [10]. It serves as an alternative to the currently prevalent proof-of-work-based schemes to agree on a network state based on a distributed trust model. Therefore, we extend the scheme for use in dynamic network environments. In order to still provide a verifiable validator election, the notion of epochs is introduced to the system. Moreover, its resilience and reliability is increased by validator redundancy and replication. We evaluate the security of the Webchain system, i. e., forging attacks on block timestamps. We show such attacks to be infeasible for a resonable epoch length and number of validators, even when assuming a powerful adversary.

The contributions of our paper are summarized as follows: We propose the Webchain system architecture enabling referential and citation provenance in a secure and verifiable way (Sec. II). We integrate distributed timestamping to provide a proof of existence (Sec. III). We evaluate Webchain's resilience in the face of a forging attack (Sec. IV).

II. System Overview

In the following, we give an overview of Webchain's core architecture, before discussing individual system components.

A. The Webchain Architecture

The backbone protocol of the Webchain architecture is inspired by blockchain-based systems, such as Bitcoin [15]. Like these systems, Webchain draws its security and accountability from replicated data structures consisting of cryptographically

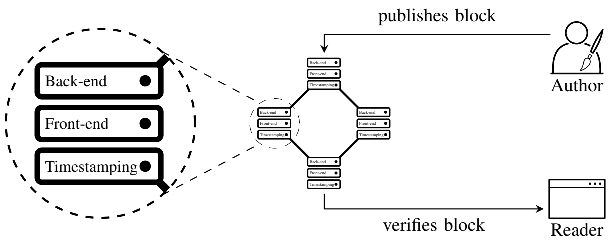


Fig. 1. Webchain architecture.

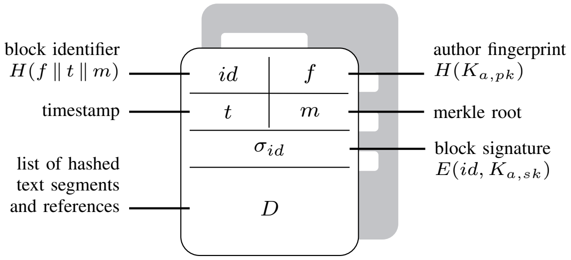


Fig. 2. Webchain block layout.

interlinked blocks, i. e., distributed ledgers. In Webchain however, every block represents an article published on the WWW. Each of these blocks is linked to every block corresponding to a cited article. Updates to a distributed ledger of blocks are broadcast in a network of *infrastructure nodes* that verify and apply them to their local ledger state. Infrastructure nodes consist of three components (cf. Figure 1): a back-end component that manages and provides access to the distributed ledger data, a front-end component that integrates an author-facing content management system, and a timestamping component that securely keeps track of block creation.

The Webchain system knows two types of users: *authors* and *readers*. Authors register at an infrastructure node to create and publish articles, which may cite other articles. The Webchain front-end provides an interface which authors use to add or edit articles. The front-end also takes care of all Webchain-related procedures. In particular, it verifies the validity of citation chains, creates a new block, and embeds the Webchain metadata into the article. In addition, articles are signed by the author. We envisage that readers use a lightweight *client* software, implemented as a browser plugin. The client scrapes visited websites for embedded Webchain metadata, queries Webchain infrastructure nodes for the block data, and verifies the citation chain. Therefore, the client does not need additional external knowledge to verify block integrity locally and in particular is not required to trust a specific infrastructure node to be trustworthy.

B. Block Layout

A Webchain block B is defined as the 6-tuple $B = (id, f, t, m, \sigma_{id}, D)$, which is also illustrated in Figure 2. A block's main purpose is to hold and secure a set of data entries D. In the Webchain system, we differentiate two types of data entries: (i) hash values of text segments, which are part of an article and (ii) references to other blocks' data entries. These entry types are stored as a list, discriminated by a preceding type header. In order to secure data entries, they are used to construct a Merkle tree [14]. As all values are incorporated in the root value m, the *Merkle root*, the value m secures the integrity of all block data. Furthermore, Webchain adds a POSIX timestamp t and the block author's fingerprint f, which is given by applying a cryptographically secure hash function H to the author's public key $K_{a,pk}$, i. e., $H(K_{a,pk})$. A block is uniquely identified by the block identifier $id = H(f \,\|\, t \,\|\, m)$, which is also recorded in a block. Finally, the author adds her digital signature $\sigma_{id} = sign(id, K_{a,sk})$, where $K_{a,sk}$ denotes her private key. This cryptographic block construction ensures that block identifiers depend on the block content as well as on the time it was created. Therefore, the construction allows to check block integrity by reproducing and verifying the block identifier based on the article data itself. Since a cryptographic signature of the identifier is provided by the author, a block's authenticity can also be validated. We assume that the respective key material is known by the validating party. This can be implemented by employing a variety of well-known identity management techniques.

C. Block Creation

From an author's perspective creating a new article feels very much the same as before: she signs in to her respective infrastructure node's CMS and uses the web interface to edit an article. After saving the article, however, the infrastructure node prepares the article by subdividing the text into text segments. The reason for this kind of segmentation is to provide other authors the ability to cite individual text segments rather then requiring to cite complete articles only. The segmentation can be achieved in different ways, e. g., by subdividing HTML elements. Alternatively, the text could be segmented based on sentence boundaries or semantically coherent text spans. Both are well-studied areas in the field of natural language processing [9], [16], [17]. An additional normalization step might be necessary to yield an deterministic segmentation.

After preprocessing, the infrastructure node applies the hash function H to each of the article's segments and appends the results to the data section D of the new block B. Given D, the node derives the Merkle root m, adds f and t, and calculates the block identifier id, which is signed by the author. The block identifier is then embedded in the page header as a corresponding HTML `meta` tag.

If an author cites an article which is part of the Webchain ecosystem, a respective reference has to be added to D. In this case, the client software automatically detects and includes Webchain metadata when copy-pasting from a web site that is part of the Webchain system. The reference, in form of a block identifier, is also included as an HTML attribute.

When citations are translated to block references, so-called *citation chains* emerge. Moreover, since various blocks can reference the same block, multiple citation chains can form a *citation graph* which is basically a directed acyclic graph (DAG). Please note that many independent citation graphs can coexist as blocks are not guaranteed to reference prior blocks. This is especially noteworthy, as it differentiates the Webchain design clearly from other blockchain-based approaches: the Webchain system does not host a single ledger of blocks, but many ledgers that only have to be stored by the involved parties, which follows the separation of concerns principle.

Blocks are announced in a peer-to-peer network of infrastructure nodes. Every node receiving a new block verifies its validity before forwarding the block to its neighbors. If one of the node's registered authors is associated with the block, i. e., a path between a block of the author and the new block exists, the block is replicated by the infrastructure node. Otherwise, the block is discarded after forwarding.

D. Block Verification

When infrastructure nodes receive a previously unknown block $B_0 = (id_0, f_0, t_0, m_0, \sigma_{id_0}, D_0)$, they perform a number of checks to verify its *validity*:

1) Construct a Merkle tree from D_0 with its root m'_0.
2) Check that the Merkle roots match: $m_0 = m'_0$.
3) Check the block signature: $id_0 = \textit{verify}(\sigma_{id_0}, K_{a_0,pk})$.

Additionally, when the nodes are involved with the block's citation graph, they inspect D_0 and recursively retrieve all referenced blocks B_i, $i \in \{1 \dots N\}$, i.e., they traverse the citation graph. This enables them to verify the *citation provenance*, by running additional checks:

1) Follow all references $d_j \in D_i$, $i \in \{0 \dots N\}$ and assert that d_j is indeed included in the referenced block.
2) Assert that all referenced blocks are *valid* by running the validity checks described before.

Therefore, a block is considered *valid*, when the integrity of the data is confirmed through use of hashing and the Merkle tree construction. Furthermore, *citation provenance* is ensured by following the cryptographically secure links between blocks and verifying the existence of data items in the cited sources.

The client software checks the accessed websites for Webchain metadata. When the client detects such metadata, it extracts the block identifier and retrieves the corresponding block B_0 and referenced blocks B_i from an infrastructure node in the background. Based on this data, the client constructs its own local data set D'_0. In order to verify the article's content, the client proceeds with the following steps:

1) Calculate Merkle root m'_0 from D'_0.
2) Assert that m'_0 and the retrieved block's m_0 match.
3) Assert the *validity* and *citation provenance* of the retrieved blocks B_i, as described before.

Given that blocks can only reference other blocks that already exist, it should be the case that timestamps in each individual citation chain follow a certain pattern. In particular, when following an individual link from a citing block B_i to a referenced block B_j, it should hold that $t_j < t_i$. However, this property can only be guaranteed to hold, if nodes have a meaningful way to verify timestamps for a specific block, which then is approved or discarded accordingly.

So far, we neglected to describe how infrastructure nodes are able to verify the timestamp information provided in the blocks. Webchain addresses this issue by implementing a distributed timestamping scheme, which is described in the following section.

III. Distributed Timestamping

Distributed timestamping [10] is based on the idea that a user should—rather than placing trust in a single centralized authority—distribute the trust to a whole network of timestamping authorities, which we call *timestamping validators*. By cryptographically signing hashed data together with a timestamp, these validators attest that they have seen the data at a specific point in time. Of course, it would immensely reduce the security properties of the timestamping protocol, if an attestation would no longer be required to come from a specific validator, but could stem from any of the validators. Therefore, Haber and Stornetta introduced the idea of deterministically electing k designated timestamping validators based on the content of the data itself, k being a global constant. This is done by using the hash y of the data (which is in our case $y = id$) as seed for a pseudorandom number generator (PRNG) function G, whose output sequence directly determines the k designated validator nodes as

$$V_y = \bigcup_{i=1\dots k} G(y, i)$$

The hash y is sent to each of the selected validator nodes V_y, which respond with a digital signature, effectively creating a timestamp.

Verification of these timestamps is done as follows: the verifying party hashes the data to gain y and calculates k executions of G to reconstruct the set of designated validators. The data is considered valid, if all of the determined validators have signed the input. Thereby, the security of this approach does not directly come from increasing the number of validators (this only distributes the trust), but from the deterministic validator election. For this, it is assumed that a cryptographically secure hash function is used (i. e., the probability of collisions can be neglected), and that it is hard to spawn validator nodes with arbitrary identifiers.

In order to adapt this scheme for the Webchain system, we assume that every infrastructure node also functions as a timestamping validator and holds a public-private key pair specifically for attestation purposes. Validators are addressed by its fingerprint $v_i = H(pk_i)$ with pk_i being their public key. When new blocks are distributed in the network, every node, including the validators, are able to calculate the set of designated validators V_{id} for a block identifier id. This of course also applies to the designated validators, which issue a corresponding attestation upon reception of a block they are responsible for. The validator's attestation is again broadcast in the network, and stored replicas of the blocks are updated by appending the additional signatures to the field σ_{id}. Blocks are considered *valid*, if they not only feature the author's signature, but also the k validator signatures. However, deploying the

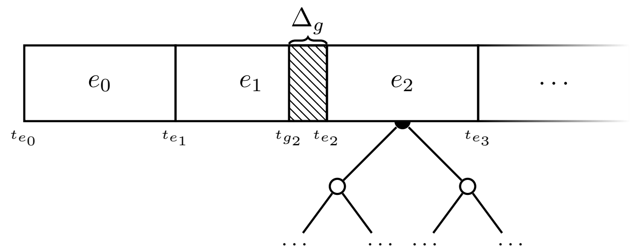


Fig. 3. Epochs define time intervals in which a specific validator set is considered active. The validator set is secured by a Merkle root. The grace period Δ_g leaves room for data propagation but also implies, for example, that valid announcements for e_2 need a timestamp dated before t_{g_2}.

scheme found in [10] is not entirely feasible for the Webchain system: based on the construction, it can be assumed that the authors required a network of static nodes, which however does not fit the dynamic design of Webchain's network. In Webchain, we therefore propose an extension that manages the network state in a dynamic overlay of validators.

A. Validator Membership Management

While we can assume that Webchain infrastructure nodes are relatively stable, the network still is of dynamic and open nature, which requires a flexible approach for addressing and determining the designated validators. We address this issue by—instead of directly using the output sequence of the PRNG—introducing an addressing scheme based on the XOR metric known from Kademlia [13]. This non-Euclidean metric assigns a distance to two binary values, by applying the XOR operation and interpreting the result as a integer number, i. e., $d(x, y) = (x \oplus y)_{10}$. By using the XOR metric, Webchain can assign a well-defined distance between each value from the sequence obtained by G and all existing validator nodes. This allows to determine the k designated validators, i. e., V_y, by selecting the k node identifiers *closest* to the values obtained from G. However, since the deterministic validator election scheme not only depends on the data input, but also on the validator set, a node verifying a block has to know the state of the validator network *at the time of the block's creation.*

In order to manage the network state, we divide time into *epochs*. In each particular epoch, the state of validators s builds the basis for the respective validator election. We denote such an epoch as $e_i = (i, s)$, including a continuous epoch number $i \in \mathbb{N}$ and a value representing the state s. When joining the network, every validator starts announcing her participation for a future epoch. Every infrastructure node keeps track of these *announcements* and thereby keeps a list of candidate validators active for a specific epoch i. All infrastructure nodes should reach consensus on which validators are considered active in a specific epoch. However, depending on a variety of factors (e. g., network performance), an announcement could reach some nodes in time to register for a specific epoch, while it may reach others too late. To mitigate this consistency problem, we use the timestamping solution itself: other validator nodes sign announcements and broadcast the attestations in the network. Similar to blocks, announcements are only considered valid if they are signed by k designated validators and received before the epoch starts. This is, announcing nodes are then deemed active in the respective epoch.[1]

In order to deal with network delays, infrastructure nodes require that announcements have a timestamp t_a that is dated at least some time Δ_g earlier than the start time t_{e_i} of the respective epoch e_i. Therefore, $\Delta_g = |t_{e_i} - t_{g_i}|$ defines a grace period, which is illustrated in Figure 3. The grace period is introduced to ensure that "fresh" announcements and the related attestations reach all nodes in the network, before the start of the new epoch. Therefore, we think t_{g_i} should be chosen so that Δ_g is at least double the maximum expected round-trip time (RTT) in the network, i. e., $\Delta_g > 2 \cdot RTT_{max}$. For practical considerations, a Δ_g of around 500 ms should suffice for most network topologies. From the list of active validators, every node builds a sorted Merkle tree, yielding a root value which succinctly and securely describes the membership state of this epoch, denoted as s. When new blocks are created, e_i is appended to the block and secured by the identifier hash value, now respectively denoted as $B = (id, f, t, m, \sigma_{id}, e_i, D)$ and $id = H(f \,\|\, t \,\|\, m \| \, e_i)$. The introduction of epochs allows to retroactively verify that a block was signed by the correct set of validators. As a consequence, joining nodes have to retrieve the historical epoch data needed for verifying a particular block. As the introduction of epochs limits the time a specific set of validators is designated for a given input, it also increases the system security (cf. Sec. IV).

B. Validator Redundancy

The described timestamping scheme allows to determine which validators are responsible for signing a specific data item in any given epoch. However, what happens if a designated validator becomes unavailable or maliciously refuses to issue attestations? As infrastructure nodes only accept new blocks and announcements when they are confirmed by k designated nodes, refusing to attest could be an easy way to run a Denial-of-Service (DoS) attack on the Webchain system. Fortunately, there is an effective way to mitigate such attacks in form of validator redundancy: infrastructure nodes do no longer check for k attestations of designated validators, but they check for a number $k' < k$. This solution was already proposed by Haber and Stornetta and in our case also helps to handle validator node failure: when a validator announces its participation in an epoch, but then fails to attest a block or announcement, the block is still valid, if at least k' validators are still online.

Leaving a certain degree of freedom and accepting a smaller number of attestations, though, inevitably reduces the security level. As we will see in our evaluation, it increases an attacker's chances to contribute the set of designated validators.

C. Validator Replication

As described before, infrastructure nodes replicate all citations graphs which are connected to any articles created by

[1] Note, that this introduces a bootstrapping problem: the announcements for the initial validator set cannot get attested by a previous set of validator nodes. Therefore, we assume a set of bootstrapping nodes for e_0 to be hard-coded in the Webchain software.

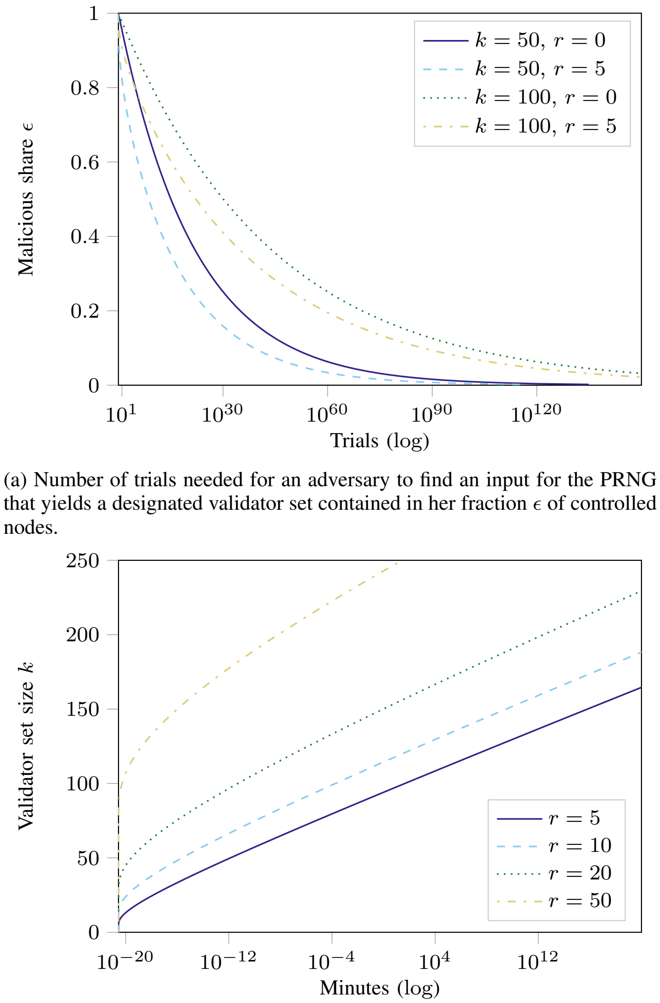
their authors. But, this also implies that only parties with some interest in the cited articles are storing the data. In particular, since they would be the only replicating nodes, this would enable malicious infrastructure nodes to remove old uncited articles from the network. To mitigate such an after-the-fact manipulation and to guarantee *non-repudiation* for articles in the Webchain system, we introduce validator replication. This is, in addition to the respective infrastructure nodes, the set of designated validators also replicate the attested blocks. Therefore, the block data is replicated k more times in the network, depending on its content. This ensures that an article that is once inserted into the Webchain system cannot be removed afterwards.

IV. Evaluation

In the following, we analyze the properties of the timestamping algorithm in case of an attack. We assume an adversary trying to construct a block with a forged timestamp. The adversary can spawn additional infrastructure nodes and hence control a fraction ϵ of all active validator nodes in a given epoch.[2] In order to succeed with forging a timestamp, the adversary can only try different combinations of input data, e. g., by adding or varying certain parts of the content, until the deterministic validator election yields k validators which are exclusively controlled by herself. As every run of the pseudorandom generator function is statistically independent, the probability of choosing k malicious validators is $p_k = \epsilon^k$. Assuming a uniform distribution of hash values, the number of trials before succeeding can be estimated as p_k^{-1}. However, the distribution changes when we introduce validator redundancy r: in case of allowing for a subset of at least $k' = k - r$ attestations, the probability for a successful attack can be calculated as

$$p_{k'} = \sum_{i=k'}^{k} P(X = i) = \sum_{i=k'}^{k} \binom{k}{i} \epsilon^i (1-\epsilon)^{k-i}.$$

Figure 4a shows the expected number of necessary trials for a varying fraction of malicious active validators ϵ and $k \in \{50, 100\}$ designated validators. For example, even when controlling a fraction $\epsilon = 0.5$ of all validators, an attack is only possible after approximately $1.13 \cdot 10^{15}$ and $1.26 \cdot 10^{30}$ trials for $k = 50$ and $k = 100$, respectively. Moreover, it is shown that redundancy significantly weakens security: when allowing $r = 5$ redundant validators (i. e., $k' = k - 5$), the number of trials is approximately $4.75 \cdot 10^8$ for $k = 50$ and $1.59 \cdot 10^{22}$ for $k = 100$. As each trial involves calculating a number of hashes, these numbers may be compared to a proof-of-work process, e. g., mining in Bitcoin [15]. As of March 2018, the biggest mining pool, BTC.com has an aggregated hash rate of around $6 \cdot 10^{18} H/s$ [5]. When we assume these efforts to be equivalent, this would mean that the biggest mining pool in Bitcoin could instantly run an attack on Webchain with $k = 50$, given that it controls more than 50% of infrastructure nodes. However, for the case of $k = 100$, it would take such an powerful adversary a much longer period of time, that is, approximately $2.10 \cdot 10^{11} s$ or around 6,660 years. Then again, when redundancy is introduced, this shrinks to around 45 minutes.



(a) Number of trials needed for an adversary to find an input for the PRNG that yields a designated validator set contained in her fraction ϵ of controlled nodes.

(b) Average time needed for a successful attack run by an adversary comparable to the largest Bitcoin pool, depending on replication parameter k and redundancy parameter r, assuming $\epsilon = 0.5$.

Fig. 4. Forging attack on block timestamps.

Clearly, there is a trade-off between the security and the reliability of the system. Therefore, we further investigate this trade-off to infer parameters rendering the forging attack infeasible, while still providing an adequate level of reliability. Since the attacker has to find an appropriate input value before the underlying state changes at the start of the next epoch, the time for running the attack is limited by the epoch length. Therefore, by choosing an epoch length that is sufficiently lower than the expected time needed for the attack, based on a given network size k and a desired redundancy factor r, this attack can be mitigated.

However, the shorter the epoch, the more epochs are created, hence the higher the overhead in terms of space and message complexity. Therefore, we propose to fixate the epoch duration

[2] Note that depending on the identity management system, rate limiting node registrations could mitigate spawning a large number of nodes.

at a reasonable default value, e. g., in the order of minutes to hours, and adjust r with respect to a given network size k. In the worst case, assume an adversary with comparable computational capabilities as BTC.com and in control of 50% of active validator nodes. Based on this adversary model, Figure 4b shows the average time needed for a successful attack, depending on the parameters k and r. Again, it can be seen that the redundancy factor has a big impact on how fast the attacker can conduct the attack. For example, if we assume a Webchain network of just $k = 200$ infrastructure nodes and a redundancy factor of $r = 20$, an attack would still take more than 4,685,000 years, well above any reasonable epoch value. If we assume a network of $k = 150$ nodes and $r = 10$, timestamps would be still secure, since an attack would take around 5,996 years. And, as mentioned before, for $k = 100$ and $r = 5$, the system would still hold for approximately 44 minutes, which may be considered insecure for an epoch length of 1 hour. Then again, given we are considering the worst-case, five redundant validators would probably still be fine for an epoch length of 10 minutes.

While the epoch length and r should be adapted based on the requirements experienced in practice, it is expected that—if gaining traction—the Webchain network would easily reach 100 nodes before drawing the interest of a powerful attacker. Therefore, we propose an epoch length of 10 minutes and $r = 5$ as default parameters for the Webchain system.

V. Related Work

The challenging goal of digital preservation is defined as the archival of authenticated content over time. To this end, web archives such as the Wayback Machine at the Internet Archive [1], Perma.cc [3], and WebCite [7] have evolved to ensure the preservation of digital content. While these approaches sustain availability of web pages, they do not protect from manipulation of the archive's contents [4].

To some extent, the OpenTimestamps project [2] tackles these weaknesses by using Bitcoin's blockchain as a secure time attestation service.

Moreover, another often neglected property is citation provenance (i. e., history of creation, ownership, and referential integrity), which is essential to ensure the reader's ability to verify claims and statements.

To the best of our knowledge, the Webchain system is the first approach that combines content authenticity and integrity with citation provenance and non-repudiation provided by secure timestamping. While approaches which require complete trust in a centralized authority exist [8], Webchain's architecture distributes the necessary trust. To this end, Webchain constructs a distributed ledger, which shows similarities to blockchain-based approaches such as Bitcoin [15] and directed acyclic graphs (DAGs) of blocks. Webchain, however, follows the separation of concerns principle and allows multiple independent ledgers.

A central component of Webchain's architecture is the timestamping service. In order to distribute the trust, we integrate Haber and Stornetta's distributed timestamping scheme [10]. While they assume a static network and do not specify node addressing, we fill the gap and develop a protocol for a dynamic overlay network, which includes maintaining network state in a distributed environment.

VI. Conclusion

With Webchain, we presented a novel approach to citation provenance for the World Wide Web. It delivers a transparent and secure way to verify citations and references. We achieve this goal by securing article and referential integrity with a distributed ledger maintained by infrastructure nodes. Furthermore, we integrated distributed timestamping and introduced a membership protocol, which makes use of epochs to manage and maintain the network state. As consequence, we gain time consistency and non-repudiation of articles and references. Moreover, we examined the system security in general and evaluated forging attacks in particular. The results provide insights on the parameter choice with respect to the security properties. We can conclude that the combination of distributed ledger technologies and distributed timestamping effectively renders forging attacks infeasible without requiring a central trust anchor.

Acknowledgment

The authors thank the German Federal Printing Office, in particular Alexander Mühle, for valuable discussions.

References

[1] "Internet archive: Wayback machine." [Online]. Available: https://archive.org/web/

[2] "Opentimestamps." [Online]. Available: https://opentimestamps.org

[3] "Perma.cc." [Online]. Available: https://perma.cc

[4] M. Aturban, M. L. Nelson, and M. C. Weigle, "Difficulties of timestamping archived web pages," *CoRR*, vol. abs/1712.03140, 2017.

[5] BTC.com, "Pool distribution," accessed on 12.3.2018. [Online]. Available: https://btc.com/stats/pool?pool_mode=week

[6] R. P. Dellavalle, E. J. Hester, L. F. Heilig, A. L. Drake, J. W. Kuntzman, M. Graber, and L. M. Schilling, "Going, going, gone: Lost internet references," *Science*, vol. 302, no. 5646, pp. 787–788, 2003.

[7] G. Eysenbach and M. Trudel, "Going, going, still there: Using the webcite service to permanently archive cited web pages," *JMIR*, vol. 7, no. 5, Dec 2005.

[8] M. Factor, E. Henis, D. Naor, S. Rabinovici-Cohen, P. Reshef, S. Ronen, G. Michetti, and M. Guercio, "Authenticity and provenance in long term digital preservation: Modeling and implementation in preservation aware storage," in *TaPP'09*.

[9] G. Grefenstette and P. Tapanainen, "What is a word, what is a sentence?: problems of tokenisation," 1994.

[10] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *CRYPTO '90*, 1991, pp. 437–455.

[11] J. C. Hernandez. (2017, Nov.) Leading western publisher bows to chinese censorship. [Online]. Available: https://nyti.ms/2z3K5aZ

[12] W. Koehler, "A longitudinal study of web pages continued: a consideration of document persistence," *Information Research*, vol. 9, no. 2, pp. 9–2, 2004.

[13] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *IPTPS '02*, pp. 53–65.

[14] R. C. Merkle, "A digital signature based on a conventional encryption function," in *CRYPTO '87*, pp. 369–378.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[16] D. D. Palmer and M. A. Hearst, "Adaptive multilingual sentence boundary disambiguation," *Computational Linguistics*, vol. 23, no. 2, pp. 241–267, 1997.

[17] J. C. Reynar and A. Ratnaparkhi, "A maximum entropy approach to identifying sentence boundaries," in *ANLP '97*, Mar. 1997, pp. 16–19.