Establishing and Verifying Fixity of Archived Web Pages

Mohamed Aturban
Old Dominion University
Department of Computer Science
Norfolk, Virginia 23529, USA
maturban@cs.odu.edu

ABSTRACT

The number of public and private web archives has increased, and we implicitly trust content delivered by these archives. Currently, users can access web archives without the ability to check fixity of content. Checking fixity is performed to ensure archived resources have remained unaltered since the time they were captured. We have noticed that most web archives do not allow users to access fixity information. More importantly, even if fixity information is available, it is provided by the same archive delivering the content. In this research, we propose a framework to establish and check fixity of archived resources. This framework does not require changes in the current web archiving infrastructure, and it will be built based on well-known web archiving standards, such as the Memento protocol. In addition, the proposed framework will result in two main benefits. First, any user can generate fixity information, not only the archive serving the content. Also, it allows anyone to check fixity of archived content regardless of which archive/user generated the fixity information. Second, the proposed framework defines a process to store fixity information independently from the archive of the associated content.

1 INTRODUCTION

Web archives, such as the Internet Archive[1] (IA) and UK Web Archive[2], have made great efforts to capture and archive the web to allow access to prior states of web resources. We implicitly trust the archived content delivered by archives, such as the Internet Archive, but with the current trend of extended use of other public and private web archives [5, 8], we should consider the question of validity of archived web pages by checking fixity to ensure archived resources have remained unaltered since the time they were received. For example, if a web page is archived in 1999 and replayed in 2017, how do we know that it has not been tampered with during those 18 years?

Generating and verifying fixity of archived content in web archives is a challenging problem because:

- Web archives continuously modify archived pages to be appropriately replayed. Web archives rewrite links, insert banners and HTML tags, create additional format like ZIP, WARC and JPEG/screenshots, delete malicious code, and others.
- With those changes, how to detect tampering of archived resource? In other words, how to differentiate between malicious and non-malicious changes?

1.1 Motivating Examples

When replaying the same archived web page in a web browser at different points in time, a user should be presented with the same content. Figure 3 shows an archived web page, or memento[3], captured by a private web archive, "Michael's Evil Wayback"[4], on July 17, 2017 at 18:51 GMT. This memento is a copy of the original web page `https://climate.nasa.gov/vital-signs/carbon-dioxide/`. Figures 3a and 3b demonstrate an unexpected result — when replaying the memento in August 2017, the level of CO_2 (or carbon dioxide in the Earth's atmosphere) was 406.31 *ppm*, but when revisiting the same archived page in October 2017, CO_2 became 270.31 *ppm*. So which one is the "real" archived page?

We can detect that the content of an archived web page has been altered by generating a cryptographic hash value on the returned HTML content. For example, the `cURL` command in Figure 2 downloads the HTML code of the main file of the memento and then the command `shasum` generates a SHA-256 hash on the downloaded content. We can detect that the content of an archived page has been altered by running these commands on the same archived web page at two different times and observing the two different hash values. Figure 4 illustrates how the simple approach of generating hashes using `cURL` and `shasum` commands can detect any tampering with content of archived pages. In this example, the "black hat" in the figure (i.e., Michael's Evil Wayback) has changed the CO_2 to a lower value, and the change can be detected by comparing the hashes. However, in our recent technical report [3], we showed that it is not easy as it might seem to generate repeatable hash values on an archived web page. We introduced some requirements to be fulfilled in order to generate repeatable hashes.

The previous example is about a malicious change. There are cases where changes are unexpected but non-malicious. One example is illustrated in Figure 1. When downloading the archived page `perma-archives.org/warc/20170101182813/http://umich.edu/` on November 16, 2017, the image `http://perma-archives.org/warc/20170101182814/http://umich.edu/includes/image/type/gallery/id/113/name/ResearchDIL-19Aug14_DM%28136%29.jpg/width/152/height/152/mode/minfit/` was embedded in the page. On December 25, 2017, downloading the same archived page, we noticed a simlilar looking image was included `http://perma-archives.org/warc/20170101182814/http://umich.edu/includes/image/type/gallery/id/113/name/ResearchDIL-19Aug14_DM%28136%29.jpg/width/152/height/152/mode/minfit/` but their hashes are different as Figure 1 depicts. We used Resemble.js [9] to compare the two images pixel by pixel. The mismatched pixels are shown in Figure 1c in pink.

[1] `http://archive.org`
[2] `http://www.webarchive.org.uk/ukwa/`
[3] A memento is an archived version of an original web page [31].
[4] We established this archive to demonstrate different scenarios in this paper.

(A) Downloaded on November 16, 2017. Its hash ends in "4465eb88c7".

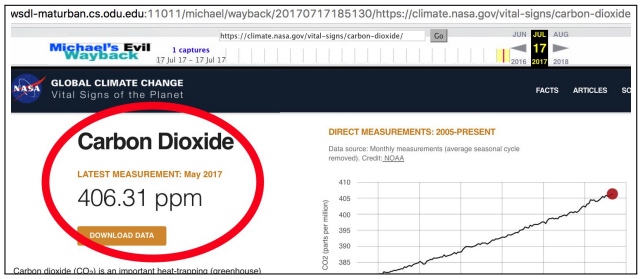
(B) Downloaded on December 25, 2017. Its hash ends in "021e7b224b".

(C) Comparing images (a) and (b) using [9] (mismatched pixels in pink).

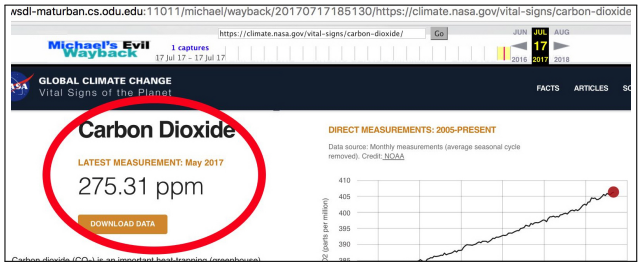
FIGURE 1: The same image from perma-archives.org downloaded at two different times, produced two different hashes.

```
% curl -s http://wsdl-maturban.cs.odu.edu:11011/michael/wayback/20170717185130/https://climate.nasa.gov/vital-signs/carbon-dioxide/ | shasum -a 256
e834c71aefda284fe03a...a8884aecec29bd2d66cbf521 -
```

FIGURE 2: Commands to generating a hash value of a memento.



(A) Accessing the archived page in August 2017 (CO_2 was 406.31 ppm)



(B) Accessing the same memento in October 2017 (CO_2 became 270.31 ppm)

FIGURE 3: A change is made in an archived page. Which one is the real archived page?

1.2 The importance of fixity checking

In the context of web archiving, fixity verifies that archived resources have remained unaltered since the time they were received [26]. The final report of the PREMIS Working Group [16] defines information used for fixity as "information used to verify whether an object has been altered in an undocumented or unauthorized way." Web content tampering is one of the common Internet-related crimes in which the content is altered by malicious users and activities [19]. A part of the problem is the lack of standared techniques that users can apply to verify the fixity of web content [1, 13, 27].

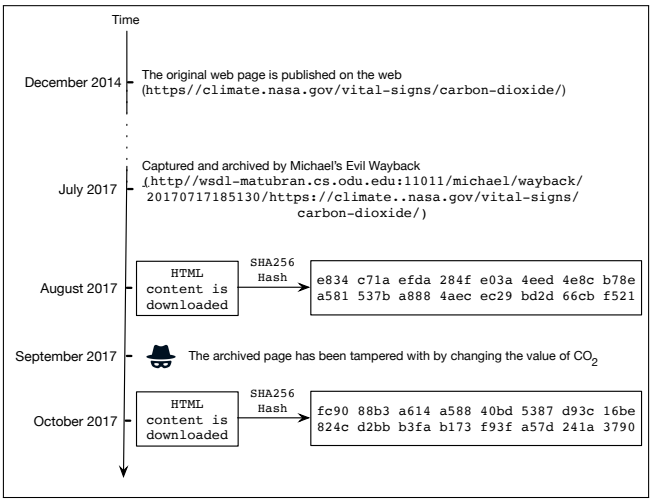


FIGURE 4: Generating a hash on the HTML content successfully detected that the archived web page has been tampered with.

Jinfang Niu mentioned that none of the web archives declare the reliability of the archived content in their servers, and some archives (e.g., The Internet Archive, WAX[5], and Government of Canada Web Archive[6]) have a disclaimer stating that they are not responsible for the reliability of the archived content they provide [25]. The concern now even if a web resource is found in public archives, there are still questions about how to verify that these archived resources have not been tampered with or corrupted. Therefore, we are trying to provide a mechanism to make archived web resources verifiable. We think that there are three main reasons why verifying fixity of archived content is important.

1.2.1 *Establishing trust in web archives.*

The report Trusted Repositories Audit & Certification (TRAC) by The Task Force on Archiving of Digital Information introduces criteria for identifying trusted digital repositories [11]. In addition to the ability to reliably provide access, preserve, and migrate digital resources, digital repositories which include web archives must

[5]http://wax.lib.harvard.edu/collections/home.do
[6]http://www.collectionscanada.gc.ca/webarchives/index-e.html

create preservation metadata that can be used to verify that content is not tampered with or corrupted (fixity) according to sections B2.9 and B4.4. It is recommended that preserved content is stored separately from fixity information, so it is less likely that someone is able to alter both the content and its associated fixity information. Thus, generating fixity information and using it to ensure that archived resources are valid will help to establish trust in web archives.

1.2.2 *Threats.*

Rosenthal et al. [28] described several threats against the content of digital preservation systems (e.g., web archives). The authors indicated that designers of archives must be aware of such threats.

- Media Failure
- Software Failure
- Failure of Network Services
- Software Obsolescence
- Natural Disaster
- Internal Attack
- Organizational Failure
- Hardware Failure
- Communication Errors
- Media & Hardware Obsolescence
- Operator Error
- External Attack
- Economic Failure

1.2.3 *Issues related to the playback of archived web pages.*

There are issues related to web archives that make it difficult to generate fixity information and checking fixity of archived resources. Before sending any requested archived web page to a user, web archives may apply some transformation to appropriately replay content in the user's browser. This includes (1) adding archive-specific code to the original content, (2) rewriting links to embedded resources (e.g., images) within an archived page so these resources are retrieved from the archive, not from the live web, and (3) serving content in different file formats like images (or screenshots), ZIP files, and WARC format. Furthermore, issues, such as reconstructing archived web pages, caching, dynamic/randomly-generated content, illustrate how difficult it is to generate repeatable fixity information. Taking into account all of these archive-related issues, the main challenging problem becomes how to distinguish between legitimate changes by archives and malicious changes.

2 RESEARCH GOALS

This proposal aims to introduce a framework to enable web archives and their users to generate fixity information of archived content and use this information to verify that received content has not been tampered with or corrupted. We set our research goals for this proposal as following:

(1) Define requirements for generating repeatable fixity information on the playback of archived web pages. The framework should also allow generating fixity information on packaging content (e.g., WARC or ZIP files).

(2) Identify and quantify changes in the playback of archived web pages.

(3) Web archives continuously apply changes to archived pages for better replay. With those changes, it is difficult to detect tampering of archived resources, so one of my research goals is to introduce approaches to differentiate between malicious and non-malicious changes.

(4) Describe the framework's structure and vocabulary. To establish trust in archived resources, the framework should allow generating fixity information at crawling time and also at any time when an archived web page is replayed. This part should have a clear answers to the questions: how fixity information is generated, how fixity information is discovered, where to store fixity information, who can generate fixity information, etc.

(5) For serialization, study different existing data models and standards like Open Annotation Data Model (OA), Open Annotation Protocol (OAP), Open Archives Initiative: Object Reuse and Exchange (OAI-ORE), Linked Data Platform (LDP), WAT/WET, and BagIt specifications.

(6) Evaluate the proposed framework. This can be achieved by building web services for publishing fixity information on the web, discover fixity information, preserve fixity information, and validate the fixity of archived resources.

3 BACKGROUND AND RELATED WORK

In order to automatically collect portions of the web, some web archives employ web crawling software, such as the Internet Archive's Heritrix [29]. Having a set of seed URIs placed in a queue, Heritrix will start by fetching web pages identified by those URIs, and each time a web page is downloaded, Heritrix writes the page to a WARC file [18], extracts any URIs from the page, places those discovered URIs in the queue, and repeats the process.

The crawling process will result in a set of archived pages. To provide access to their archived pages, many web archives that use OpenWayback [17], the open-source implementation of IA's Wayback Machine, allow users to query the archive by submitting a URI. OpenWayback will replay the content of any selected archived web page in the browser. One of the main tasks of OpenWayback is to ensure that when replaying a web page from an archive, all resources that are used to construct the page (e.g., images, style sheets, and JavaScript files) should be retrieved from the archive, not from the live web. Thus, at the time of replaying the page, OpenWayback will rewrite all links to those resources to point directly to the archive [30]. In addition to OpenWayback, PyWb [20] is another tool for replaying archived web pages, and it is used by Perma [34] and Webrecorder [21].

Memento [32] is an HTTP protocol extension that uses time as a dimension to access the web by relating current web resources to their prior states. The Memento protocol is supported by most public web archives including the Internet Archive. The protocol introduces two HTTP headers for content negotiation. First, Accept-Datetime is an HTTP Request header through which a client can request a prior state of a web resource by providing the preferred datetime (e.g., *Accept-Datetime: Mon, 09 Jan 2017 11:21:57 GMT*). Second, the Memento-Datetime HTTP Response header is sent by a server to indicate the datetime at which the resource was captured. The Memento protocol also defines the following terminology:

- URI-R - an original resource from the live Web
- URI-M - an archived version (memento) of the original resource at a particular point in time
- URI-T - a resource (TimeMap) that provides a list of mementos (URI-Ms) for a particular original resource

Different vulnerabilities were discovered in the Internet Archive's Wayback Machine by Lerner et al. [24]. They are Archive-Escapes, Same-Origin Escapes, Archive-Escapes + Same-Origin Escapes, and Anachronism-Injection. Attackers can leverage these vulnerabilities to modify a user's view at the time when a memento is rendered in a browser. The authors suggested some defenses that could be deployed by either web archives or web publishers to prevent abusing these vulnerabilities.

A shared repository was created in May 2017 by Cushman and Kreymer [10] to describe potential threats in web archives, such as controlling a user's account due to Cross-Site Request Forgery (CSRF) or Cross-Site Scripting (XSS), and archived web resources reaching out to the live web. The authors provide recommendations on how to avoid such threats.

Eltgrowth [12] outlined several judicial decisions that involve evidence (i.e., archived web pages) taken from the Internet Archive. The author mentions that there is an open question whether to consider an archived web page as a duplicate of the original web page at a particular time in the past. This concern might prevent considering archived web pages as evidence.

Several tools have been developed to generate trusted timestamps in blockchain-based networks. OriginStamp [15] allows users to generate a trusted timestamp on plain text, a hash value, or any file. The data is hashed by the user's browser and the resulting hash is sent to OriginStamp's server. Once delivered, it will be added to the list of all hashes submitted by other users. Once per day, OriginStamp generates a single aggregated hash of all received hashes. This hash is then converted to a Bitcoin address that will be a part of a new Bitcoin transaction. The timestamp associated with the transaction is considered a trusted timestamp. A user can verify a timestamp through OriginStamp's API or by visiting their website. Other services, such as Chainpoint[7] and OpenTimestamps[8], are based on the same concept of using blockchain-based networks to timestamp digital documents.

Even though users of the tools mentioned above can pass data by value, they are not allowed to submit data by reference (i.e., passing the URI of a web page). In other words, these services are not directly timestamping web pages. The only exception is a service [14] established by OriginStamp that accepts URIs from users, but the service is no longer available on the live web[9]. In this service the hash is only generated on the HTML content of the main file identified by the URI, ignoring all embedded resources like images, scripts, and style sheets [14]. As illustrated in our previous work [3], not including embedded resources in hash calculation may leave the page vulnerable to undetected changes.

Kuhn et al. [22] define a trusty URI as a URI that contains a cryptographic hash value of the content it identifies. With the assumption that a trusty URI, once created, is linked from other resources or stored by a third party, it becomes possible to detect if the content that the trusty URI identifies has been tampered with or manipulated on the way. In their second paper [23], they introduce two different modules to allow creating trusty URIs on different kind of content: in the module F, the hash is calculated on the byte-level file content while in the second module R, the hash is calculated on RDF graphs. Even though Trusty URIs detect altered documents, there are some limitations. First, a Trusty URI is created by an owner of a resource it identifies. Second, trusty URIs can be generated on only two types of content RDF graphs and byte-level content (i.e., no modules introduced for HTML documents). Third, one hash function SHA-256 is used to generate the hash value which might not be suitable for some use cases. Also, there should be alternatives if this hash function becomes vulnerable.

Multihash [7] is a protocol introduced by Juan Benet to mainly create self identifying hashes for IPFS content [6], but the protocol can be actually utilized to identify other content like regular web pages.

[7] chainpoint.org
[8] opentimestamps.org
[9] www.isg.uni-konstanz.de/web-time-stamps/

4 PRELIMINARY WORK

The preliminary work focuses on gaining an understanding the difficulties of generating repeatable fixity information on archived web pages, introducing some requirements for calculating repeatable fixity information, and exploring the types of changes in the playback of archived web pages.

4.1 ArchiveNow: Simplified, Extensible, Multi-Archive Preservation

Preserving a web page in only a single web archive is risky. Archives may be vulnerable to security threats, or may also become temporarily or permanently unreachable. Thus, preserving web pages in multiple web archives should decrease the danger of losing archived data. Hence we created `ArchiveNow` [2], a Python library for preserving web pages in on-demand web archives. This library allows a user to submit a URI of a web page for archiving at several configured web archives. Once the web page is captured, `ArchiveNow` notifies the user with links to the archived copies of the web page. Another important feature of this tool is the creation of WARC files by `wget` and `Squidwarc`, which helps users to create personal and private archives. `ArchiveNow` is initially configured to use four archives but is easily configurable to add or remove other archives

4.2 Requirements for generating repeatable hashes

One of the most common approaches for fixity is to generate a hash value that represents a state of a digital resource at a particular date. The hash value should fulfill Requirement 1 emphasizing reproducing the same hash of a particular memento at different points in time.

Requirement 1: Repeatable hash values
If we download a memento *URI-*M_x at time t_n (denoted as *URI-*M_x@t_n), download the same memento at time t_m (denoted as *URI-*M_x@t_m), and apply a hash function H on the content of *URI-*M_x@t_n and *URI-*M_x@t_m, then H(*URI-*M_x@t_n) = H(*URI-*M_x@t_m)

In our recent technical report [3], we showed that it is not easy as it might seem to generate repeatable hash values on an archived web page. We introduced some requirements to be fulfilled in order to generate repeatable hashes. The proposed requirements include:

(1) Generate a hash on a composite memento.
(2) Exclude archive-specific resources.

(3) Avoid resources from the live web.
(4) Avoid content served from the cache.
(5) Changes in TimeMaps might affect the computation of hashes.
(6) Avoid including dynamic or randomly-generated content.
(7) Include selected HTTP response headers in hash calculation.

The report describes each requirement in detail with examples. In general, getting repeatable fixity information is challenging because web archives continually apply changes and transformations to the original content as explained in Section 1.2.3.

4.3 Identifying and Quantifying Changes in the Playback of Archived Web Pages

When replaying the same archived web page in a web browser at different points in time, a user should be presented with the same content. We conducted a study to identify and quantify different types of changes in the playback of archived web pages leading to a better understanding of web archives' behavior on their archived resources. The study was performed on 18,472 archived web pages (mementos) from 17 different web archives. We downloaded these archived pages 10 times during 45 days between November 16, 2017 and December 31, 2017. Then, a hash value is generated for each downloaded memento considering the requiremnts mentioned in Section 4.2. The hash value summarizes the memento at a particular datetime. Finally, we identify the types of changes that cause the same memento to map to different hashes at different points in time. At first glance, a change in the generated hash value may indicate malicious modification, yet different hash values for the same memento may be caused by both expected playback-related changes or unexpected, but non-malicious, changes. Quantifying the types of changes present in today's archives will help us to differentiate between malicious and non-malicious changes in mementos in the future. Understanding these changes is important because conventional archival approaches regarding fixity are not applicable for web archives [4].

We compared each hash value with the remaining 9 hashes generated after downloading a memento 10 times. Each time Requirement 1, defined in Section 4.2, is not satisfied, we should identify the types of changes producing different hash values by comparing the content of downloaded mementos. The change could affect the main HTML file of the memento, embedded resources, or HTTP Response headers. We noticed the following types of changes:

HTTP status code: A web archive could respond with different HTTP status code when requesting the same URI-M. For example, the archive returns "404 Not Found" for previously "200 Ok" resource because it was deleted from the server.

HTTP Response Headers: For instance, the MIME type (i.e., Content-Type Response header) of a resource might be converted (e.g., from GIF to PNG).

TimeMaps: The same archived resource might redirect differently each time it is requested (i.e., a change in the "Location" HTTP header).

Transient Error: For example, a web server is unable a request or because of "server connection timeout error".

HTTP entity body. Changes in the HTTP entity occur because of dynamic content or random content generated by JavaScript.

Other. This would include any other type of change than those mentioned above. For example, similar to HTTP entity, URI-Ms of an embedded resource of a memento may have random values generated by JavaScript code, such as values associated with the current datetime, geolocation, weather, etc.

Table 1 shows that 19.48% of mementos (3, 599 out of 18, 472 URI-Ms) have at least two different final hashes when downloaded 10 times. Archive.is is a special case since it ships a requested memento in a single ZIP file. This technique results in only 6 mementos out of 1,600 having different hash values.

Table 1: Number of mementos with at least one change

Archive	URI-Ms	URI-Ms with changes (%)
web.archive.org	1,600	673 (42.06)
archive.is	1,600	6 (0.38)
archive.bibalex.org	1,600	300 (18.75)
webarchive.loc.gov	1,600	88 (0.55)
arquivo.pt	1,600	807 (50.44)
webcitation.org	1,600	365 (22.81)
wayback.vefsafn.is	1,600	378 (23.62)
wayback.archive-it.org	1,407	220 (15.64)
swap.stanford.edu	1,233	96 (7.79)
nationalarchives.gov.uk	1,011	37 (3.66)
europarchive.org	990	24 (2.42)
webharvest.gov	733	150 (20.46)
veebiarhiiv.digar.ee	518	16 (3.09)
webarchive.proni.gov.uk	477	16 (3.35)
webarchive.org.uk	362	256 (70.72)
collectionscanada.gc.ca	359	45 (12.53)
perma-archives.org	182	122 (67.03)
(total)	**18,472**	**3,599 (19.48)**

5 PROPOSED WORK

In previous research, we have introduced several requirements for generating repeatable fixity information. In addition, we identified and quantified changes in the playback of archived web pages. My future work will focus on establishing a framework for verifying fixity of archived resources. we will start by describing the overview of the framework. Then, we will explore existing data models to define the format of the serialization of fixity information and terminology. Finally, we will be evaluating the proposed framework. Table 2 shows the plan towards the completion of my thesis.

5.1 Establishing Fixity of Archived Resources

Generating and storing fixity information of archived web pages can be performed in four steps.

(1) Push a web page into multiple archives
(2) Compute fixity information of the archived pages
(3) Publish the fixity information at a well-known URI
(4) Push the published fixity information into multiple archives

We briefly describe each of the steps involved in generating fixity of archived web pages with examples. Figure 5 shows the web page `eaw.rhizome.org` pushed into multiple archives resulting in

four different archived pages of the original web page. The tool `ArchiveNow` can send parallel requests to multiple archives for preserving a web page.

FIGURE 5: Pushing a web page into multiple archives.

Next, as shown in Figure 6, fixity information is generated on the archived web pages and then published at a well-known location on the web, `manifest.org`. For example, the fixity information of the archived web page `archive.is/20180321/eaw.rhizome.org` is available at `manifest.org/20180322/archive.is/20180321/eaw.rhizome.org`. Archives may modify, transform, and rewrite archived content. Thus, when computing fixity, we should only consider content that should not change (e.g., JPEGs, original HTML code, and certain HTTP response headers). we are still investigating what content should be included in computing fixity.

FIGURE 6: Compute fixity and publish it on the web

The final step is to push the published fixity information into multiple archives. This means that the archived web page stored in one archive has its fixity information stored in several archives. In the example shown in Figure 7, the fixity information of the archived page from `archive.org` is captured by four different archives.

FIGURE 7: Push the fixity information into multiple archives

5.2 Verifying Fixity of Archived Resources

Verifying the fixity of an archived web page can be achieved by the following:

(a) Discover the published original fixity information of the archived web page
(b) Discover copies of the original fixity information
(c) Recompute current fixity information of the archived web page
(d) Compare current fixity information with the published original fixity information and its copies

Given an archived web page, we can discover the original fixity information through `manifest.org`'s API. Copies of the fixity information can also be discovered in different archives through the archives' API or Memento protocol. Currently, most web archives are Memento-compatible, which helps in discovering archived resources.

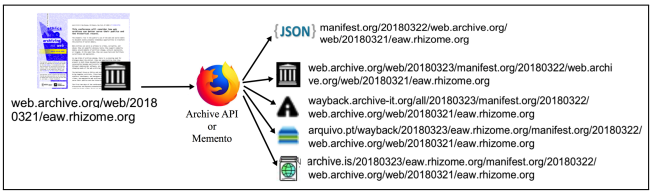


FIGURE 8: Discover fixity information in `manifest.org` and web archives

To verify fixity, we recompute the current fixity information of the given archived web page and compare it with the fixity information discovered in `manifest.org` and its copies in archives. By taking a majority vote of which copies agree or disagree with the current fixity information, we can identify if an archived web page has been changed or not. For example, Figure 8 shows that five different votes can be taken on the current fixity information.

5.3 Tamper-proof fixity information

Distributing fixity information in multiple archives has two main benefits. First, it is recommended that fixity information is stored separately from its associated archived content [11], so it is less likely that someone can modify both the content and its fixity information. The other reason is that it is safer to have more copies of fixity information distributed among different archives. If an archive becomes unreachable or compromised, we still have other copies. we will explore more places and approaches for storing fixity information, such as blockchain-based networks and InterPlanetary File System (IPFS). In addition, using Trusty-URI and Multihash can create more complex URIs that identify fixity information on the web but at the same time those URIs can be used to validate fixity information they identify.

5.4 When to generate fixity information?

We can only trust the content of archived web pages from the time when fixity information is computed and published. There are two different scenarios based on when fixity is generated. The first scenario is illustrated in Figure 9, where fixity information is generated at ingest by the archive. In other words, the archive crawls a web page which will result in creating an archived web page (denoted by `URI-M1`), then the archive computes the fixity information, or `manifest`, of `URI-M1` and publishes it by submitting this information to `manifest.org`. The published manifest is denoted by `Manifest-URI-1`. The archive pushes `Manifest-URI-1` into multiple archives. The archived manifest is denoted by `URI-1` in Figure 9. Thus, the archived resource creation date and the creation date of the fixity information should be (close to) the same.

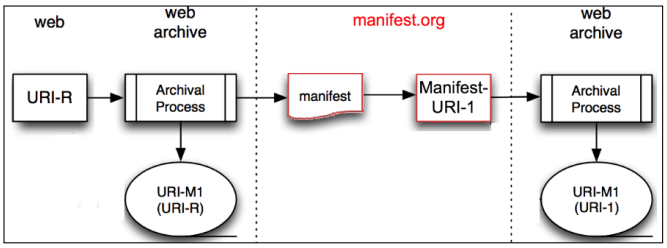


FIGURE 9: Generating fixity information at the time when the archived page is created. This is a modified version of an original diagram contributed by Herbert Van de Sompel (from LANL).

The other scenario is when fixity information computed at any point in time after an archived web page is created. This scenario is illustrated in Figure 4 where the original web page was captured by the archive in July 2017 and the fixity information was computed in August 2017.

5.5 Fixity of web packaging formats vs. on the playback of archived resources

We showed in the preliminary work (Section 4) that computing fixity on packaging format (e.g., ZIP files) usually produces the same repeatable fixity information, which is what we want, but most archives do not serve the archived content in packaging format, such as WARCs, ZIPs, or even screenshots in JPEG format. Instead, users access archives through tools like OpenWayback for replaying archived content. Generating fixity information based on the playback of archived web pages is important but complicated because each time an archived web page is played back, it may produce different content for different reasons. We identified and quantified changes in the playback of archived resources in the preliminary work, and we are planning to investigate how to generate fixity information on web packaging format, such as WARC files and "webpackage" [33] (under development).

5.6 Adopting a Data Model

We will study different existing data models to be used in our framework. We will define new terminology, but we will try to benefit from terminology introduced in other data models. We will explain advantages and disadvantages of each model resulting in selecting one data model for the framework. The suggested data models and standards are: Open Annotation Data Model (OA), Open Annotation Protocol (OAP), Open Archives Initiative Protocol - Object Exchange and Reuse (OAI-ORE), Linked Data Platform (LDP), Open Annotation Protocol (OAP), Web Archive Transformation (WAT), WARC Encapsulated Text (WET), and BagIt.

5.7 Evaluation

We will develop prove-of-concept implementation mainly for publishing fixity information on the web. The evaluation will also include developing services to generate fixity information, validate fixity information, and verify the fixity of archived resources. The evaluation will be performed on a set of archived web pages from different web archives including private archives. Fixity information will be generated and published on the web at different points in time. Then, the published fixity information will be used to verify fixity. The evaluation process will consider the following issues and scenarios:

- Fixity information at `manifest.org` has been tampered with
- `manifest.org` becomes unreachable
- Archived fixity information has been tampered with
- A web archive disappears
- When taking a majority vote, an archive is not independent from other archives (e.g., Archive-It and Internet Archive)
- When taking a majority vote, what if archived fixity information and the archived page are from the same archive
- When taking a majority vote, archived fixity information is a copy of another archived fixity information
- An archive performs non-malicious modification to archived pages
- An archived pages is a copy another archived page, not a copy of the original web page

6 CONCLUSIONS

We noticed that most of web archives do not allow users to access fixity information. Even if fixity information is available, it is provided by the same archive delivering archived content. In this proposal, we have described a framework for generating fixity information and checking fixity of archived web pages. This framework will establish trust in public and private web archives by allowing any user to generate fixity information and check fixity of archived resources. The proposed work does not require any change in the infrastructure of web archives and is build based on well-known standards and data models, such as Memento protocol.

7 ACKNOWLEDGEMENTS

Mohamed Aturban's advisor is Michele C. Weigle and Michael L. Nelson (co-advisor). Herbert Van de Sompel (LANL), and Martin Klein (LANL) provided helpful feedback and contributions. This work is supported in part by The Andrew W. Mellon Foundation (AMF) grant 11600663.

REFERENCES

[1] Shadi Aljawarneh, Christopher Laing, and Paul Vickers. 2008. Design and experimental evaluation of Web Content Verification and Recovery (WCVR) system: A survivable security system. (2008).
[2] Mohamed Aturban, Mat Kelly, Alam Sawood, John Berlin, Michael L Nelson, and Michele C Weigle. 2018. ArchiveNow: Simplified, Extensible, Multi-Archive Preservation. In *Proceedings of the 18th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*.
[3] Mohamed Aturban, Michael L. Nelson, and Michele C. Weigle. 2017. *Difficulties of Timestamping Archived Web Pages.* Technical Report arXiv:1712.03140.
[4] Jefferson Bailey. 2012. File Fixity and Digital Preservation Storage: More Results from the NDSA Storage Survey. `https://blogs.loc.gov/thesignal/2012/03/file-fixity-and-digital-preservation-storage-more-results-from-the-ndsa-storage-survey/`. (March 2012).
[5] Jefferson Bailey, Abigail Grotke, Edward McCain, Christie Moffatt, and Nicholas Taylor. 2017. Web Archiving in the United States: A 2016 Survey. `http://ndsa.org/documents/WebArchivingintheUnitedStates_A2016Survey.pdf`. (February 2017).
[6] Juan Benet. 2014. *IPFS-content addressed, versioned, P2P file system.* Technical Report arXiv:1407.3561.
[7] Juan Benet. 2014. Multihash - Self describing hashes for future proofing. `https://github.com/multiformats/multihash`. (May 2014).

Table 2: The schedule of the research plan.

Time Frame	Phase Description	Evaluation
present - 04/2018	Different approaches for storing fixity information.	Using Trust-URIs, Multihash, blockchain-based networks, and IPFS.
05/2018 - 06/2018	Identifying malicious changes on the playback of archived web page.	Conducting an extended study based on results from Section 4.3.
07/2018 - 09/2018	Candidacy proposal.	Finish writing my candidacy proposal and defend it.
10/2018 - 11/2018	Select a data model	Compare the data models OA,OAP, OAI-ORE, LDP, WAT, and WET
12/2018 - 02/2019	Define and develop several fixity information services: publishing fixity information on the web, generating fixity information, validating fixity information, verifying fixity of archived pages using discovered fixity information.	Selecting a set of archived pages for which fixity information is published. The published fixity information will be used for verifying fixity of archived pages.
03/2019 - 05/2019	Writing the thesis and the Defense.	Graduating by May 2019.

[8] Miguel Costa, Daniel Gomes, and Mário J. Silva. 2017. The evolution of web archiving. *International Journal on Digital Libraries* 18, 3 (2017), 191–205. https://doi.org/10.1007/s00799-016-0171-9

[9] James Cryer and Huddle. 2012. Resemble.js - Image analysis and comparison. https://github.com/Huddle/Resemble.js. (February 2012).

[10] Jack Cushman and Ilya Kreymer. 2017. Thinking like a hacker: Security Considerations for High-Fidelity Web Archives. http://labs.rhizome.org/presentations/security.html. (May 2017).

[11] Robin L Dale and Bruce Ambacher. 2007. Trustworthy repositories audit and certification: Criteria and checklist. Report of the RLG-NARA Task Force on Digital Repository Certification. (February 2007).

[12] Deborah R Eltgrowth. 2009. Best evidence and the Wayback Machine: toward a workable authentication standard for archived Internet evidence. *Fordham Law Review* 78 (2009), 181.

[13] Peng Gao, Hao Han, and Takehiro Tokuda. 2012. IAAS: An integrity assurance service for Web page via a fragile watermarking chain module. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*. ACM.

[14] B. Gipp, N. Meuschke, and C. Breitinger. 2016. Using the Blockchain of Cryptocurrencies for Timestamping Digital Cultural Heritage. In *Proceedings of the Workshop on Web Archiving and Digital Libraries (WADL) held in conjunction with the 16th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. https://www.gipp.com/wp-content/papercite-data/pdf/gipp2017a.pdf

[15] Bela Gipp, Norman Meuschke, and André Gernandt. 2015. *Decentralized trusted timestamping using the crypto currency Bitcoin*. Technical Report arXiv:1502.04015.

[16] PREMIS Working Group et al. 2005. Data dictionary for preservation metadata: final report of the PREMIS Working Group. *OCLC Online Computer Library Center & Research Libraries Group, Dublin, Ohio, USA, Final report* (2005).

[17] International Internet Preservation Consortium (IIPC). 2005. OpenWayback. https://github.com/iipc/openwayback/wiki. (October 2005).

[18] ISO 28500. 2009. WARC (Web ARChive) file format. http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml. (August 2009).

[19] Ramniwas Kachhawa, Nikhil Kumar Singh, and Deepak Singh Tomar. 2014. A Novel Approach to Detect Web Page Tampering. *IJCSIT) International Journal of Computer Science and Information Technologies* 5, 3 (2014), 4604–4607.

[20] Ilya Kreymer. 2013. PyWb - Web Archiving Tools for All. https://github.com/ikreymer/pywb. (December 2013).

[21] Ilya Kreymer. 2015. Webrecorder - a web archiving platform and service for all. (2015). https://webrecorder.io

[22] Tobias Kuhn and Michel Dumontier. 2014. Trusty URIs: Verifiable, immutable, and permanent digital artifacts for linked data. In *European Semantic Web Conference*. Springer, 395–410.

[23] Tobias Kuhn and Michel Dumontier. 2015. Making digital artifacts on the web verifiable and reliable. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2390–2400.

[24] Ada Lerner, Tadayoshi Kohno, and Franziska Roesner. 2017. Rewriting History: Changing the Archived Web from the Present. In *Proceedings of the 16th ACM conference on Computer and Communications Security (CCS)*. 1741–1755.

[25] Jinfang Niu. 2012. Functionalities of Web Archives. *D-Lib Magazine* 18, 3/4 (2012). http://www.dlib.org/dlib/march12/niu/03niu2.html

[26] Trevor Owens. 2014. Protect Your Data: File Fixity and Data Integrity. https://blogs.loc.gov/thesignal/2014/04/protect-your-data-file-fixity-and-data-integrity/. (April 2014).

[27] Robert L Probert, Bernard Stepien, and Pulei Xiong. 2005. Formal testing of web content using TTCN-3. In *TTCN-3 User Conference*, Vol. 2005.

[28] David Rosenthal, Thomas Robertson, Tom Lipkis, Vicky Reich, and Seth Morabito. 2005. Requirements for Digital Preservation Systems. *D-Lib Magazine* 11, 11 (2005). www.dlib.org/dlib/november05/rosenthal/11rosenthal.html

[29] Kristinn Sigurdsson. 2005. Incremental crawling with Heritrix. In *Proceedings of the 5th International Web Archiving Workshop (IWAW)*.

[30] Brad Tofel. 2007. Wayback for accessing web archives. In *Proceedings of the 7th International Web Archiving Workshop (IWAW)*. 27–37.

[31] Herbert Van de Sompel, Michael L. Nelson, and Robert Sanderson. 2013. HTTP framework for time-based access to resource states – Memento, Internet RFC 7089. http://tools.ietf.org/html/rfc7089. (2013).

[32] Herbert Van de Sompel, Michael L. Nelson, Robert Sanderson, Lyudmila L. Balakireva, Scott Ainsworth, and Harihar Shankar. 2009. *Memento: Time Travel for the Web*. Technical Report arXiv:0911.1112.

[33] Jeffrey Yasskin. 2017. Use Cases and Requirements for Web Packages. (2017). https://tools.ietf.org/id/draft-yasskin-webpackage-use-cases-00.html

[34] Jonathan Zittrain, Kendra Albert, and Lawrence Lessig. 2014. Perma: Scoping and addressing the problem of link and reference rot in legal citations. *Legal Information Management* 14, 2 (2014), 88–99.