VISVESWARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590018

TECHNICAL SEMINAR REPORT ON

"WEB ARCHIVING FIXITY VERIFICATION FRAMEWORK"

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING
Submitted By

Likith S
(1BY18CS081)

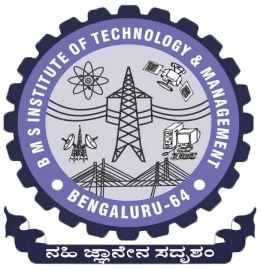
Under The Guidance Of
Mrs. Durga Bhavani A
Assistant Professor
Department of CSE

BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Avalahalli , Yelahanka , Bengaluru – 560064.

2021-2022

VISVESWARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590018

BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Avalahalli , Yelahanka , Bengaluru – 560064.



CERTIFICATE

This is to certify that the Seminar work entitled "Archive Assisted Archival Fixity Verification Framework" has been carried out by Mr/Ms. Likith S, 1BY18CS081, a bona fide student of BMS Institute of Technology and Management in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2021-2022. It is certified that all corrections/suggestions indicated for assessment have been incorporated in the report deposited in the department library. The Seminar report has been approved as it satisfies the academic requirements in respect of Seminar work prescribed for the said degree.

_____
Signature of the Guide
Mrs. Durga Bhavani A
Assistant Professor
Department of CSE
BMSIT & M

_____
Signature of the HOD
Dr. Bhuvaneshwari C M.
Professor & Head of department
Department of CSE
BMSIT & M

DECLARATION

I, Likith S**[USN: 1BY18CS081]**, student of VIII Semester BE, in Computer Science and Engineering, BMS Institute of Technology and Management hereby declare that the Seminar entitled "WEB ARCHIVING FIXITY VERIFICATION FRAMEWORK**"** has been carried out by me and submitted in partial fulfillment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during the academic year 2021-22.

Date :

Place :

Likith S

USN **:1BY18CS081**

ACKNOWLEDGEMENT

We are happy to present this Seminar report after completing it successfully. This seminar would not have been possible without the guidance, assistance, and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and everyone who has helped us make this a success.

We heartily thank our **Principal, Dr. MOHAN BABU G N, BMS Institute of Technology & Management,** for his constant encouragement and inspiration in taking up this project.

We heartily thank our **Head of the Department, Dr. Bhuvaneshwari C.M, Department of Computer Science and Engineering, BMS Institute of Technology &Management,** for her constant encouragement and inspiration in taking up this project.

We gratefully thank our Seminar Guide, **Mrs. Durga Bhavani A, Assistant Professor, Department of Computer Science and Engineering,** for his/her guidance, support, and advice.

Special thanks to all the staff members of the Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given to us in completing this precious work successfully.

Likith S
1BY18CS081

ABSTRACT

The number of public and private web archives has increased, and we implicitly trust the content delivered by these archives. Fixity is checked to ensure an archived resource has remained unaltered since the time it was captured. Some web archives do not allow users to access fixity information and, more importantly, even if fixity information is available, it is provided by the same archive from which the archived resources are requested.

In this research, two approaches are proposed namely Atomic and Block, to establish and check the fixity of archived resources. In the Atomic approach, the fixity information of each archived web page is stored in a JSON file (or a manifest), and published in a well-known web location (an Archival Fixity server) before it is disseminated to several on-demand web archives. In the Block approach, we first batch together fixity information of multiple archived pages in a single binary-searchable file (or a block) before it is published and disseminated to archives.

In both approaches, the fixity information is not obtained directly from archives. Instead, we compute the fixity information (e.g., hash values) based on the playback of archived resources. One advantage of the Atomic approach is the ability to verify the fixity of archived pages even with the absence of the Archival Fixity server. The Block approach requires pushing fewer resources into archives, and it performs fixity verification faster than the Atomic approach.

On average, it takes about 1.25X, 4X, and 36X longer to disseminate a manifest to perma.cc, archive.org, and webcitation.org, respectively, than archive.is, while it takes 3.5X longer to disseminate a block to archive.org than perma.cc. The Block approach performs 4.46X faster than the Atomic approach in verifying the fixity of archived pages.

CONTENTS

List Of Figures

Chapter 1
Introduction

Web archives, such as the Internet Archive (IA) and UK Web Archive, 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 such archives, but with the current trend of extended use of other public and private web archives[12, 15], we should consider the question of validity. For instance, if a web page is archived in 1999 and replayed in 2019, how do we know that it has not been tampered with during those 20 years? One potential solution is to generate a cryptographic hash value on the HTML content of an archived web page or memento. A memento is an archived version of an original web page. Figure 1.1 shows an example where the cURL command downloads the raw HTML code of the memento

https://web.archive.org/web/20181219102034/https://2019.jcdl.org/

and then the hashing function sha256sum generates an SHA-256 hash on this downloaded code. By running these commands at different times we should always expect to obtain the same hash.

```
$ curl -s https://web.archive.org/web/20181219102034id_/https://2019.jcdl.org/ | sha256sum
2769b5f71c64794ae76267b3a6385847e202bf0fd9fbcb8e0633427759fac128 -
```

Figure 1.1: Commands to generate a hash value of a memento.

1.1 Context

In the context of web archiving, fixity verifies that archived resources have remained unaltered since the time they were received[11]. The final report of the PREMIS Working Group[14] 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 a common Internet-related crime in which content is altered by malicious users and activities. Part of the problem is the lack of standard techniques that users can apply to verify the fixity of web content. Jinfang Niu[9] mentioned that none of the web archives declare the reliability

of the archived content in their servers, and some archives, such as the Internet Archive, WAX3, and Government of Canada Web Archive4, have a disclaimer stating that they are not responsible for the reliability of the archived content they provide.

1.2 Motivation

A motivating example, which shows the importance of verifying the fixity of mementos, is the story of Joy-Ann Reid, an American cable television host at MSNBC. In December 2017, she apologized for writing several “insensitive” LGBT blog posts nearly a decade ago when she was a morning radio talk show host in Florida[12]. In April 2018, Reid, supported by her lawyers, claimed that her blog and/or the archived versions of the blog in the Internet Archive had been compromised and the content was fabricated. Even though the Internet Archive denied that their archived pages had been hacked, a stronger case could be made if we had an independent service verifying that those archived blog posts had not changed since they were captured by the archive. Here, two approaches are introduced, Atomic and Block, to make archived web resources verifiable.

In the Atomic approach, the fixity information of each archived web page is stored in a single JSON file, or manifest, published on the web and disseminated to several on-demand web archives. In the Block approach, we batch together fixity information, or records, of multiple archived pages to a single binary-searchable file, or block. The block then is published at a well-known web location before disseminating to archives. While we make a chain of blocks, we are not attempting to create yet another Blockchain. Manifests’ chain of blocks is limited in scope as we do not need to worry about consensus, eventual consistency, or proof-of-work because these blocks are generated and published by a central authority (the Block approach is described in 3.3). In both approaches, the fixity information, such as hash values, is not directly provided by archives (server-side) even though some archives’ APIs (e.g., the Internet Archive CDX server) allows accessing such information. Alternatively, it was decided to calculate the fixity information based on the playback of archived resources (client-side) for two reasons. First, we are not expecting hashes generated and stored in WARC files by archives at crawl time to match those generated on the playback of mementos. Second, if an archive has been compromised then it is likely the corresponding

hashes have been also compromised, so we need to have the fixity information stored in independent archives.

Chapter 2
Literature Survey

In order to automatically collect portions of the web, web archives employ web crawling software, such as the Internet Archive's Heritrix[12]. 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, 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 or mementos. To provide access to their archived pages, many web archives use OpenWayback[16], the open-source implementation of IA's Wayback Machine, to 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. In addition to OpenWayback, PyWb[8] is another replaying tool, which is used by Perma and Webrecorder.

2.1 Memento

Memento is an HTTP protocol extension[7] 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.

To establish trust in repositories and web archives, different publications and standards have emphasized the importance of verifying the fixity of archived resources. The report Trusted Repositories Audit & Certification (TRAC) by the Task Force on Archiving of Digital Information introduces criteria for identifying trusted digital repositories. In addition to the ability to reliably provide access, preserve, and migrate digital resources, digital repositories which include web archives must create preservation metadata that can be used to verify that content is not tampered with or corrupted (fixity). The report recommends 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. Eltgrowth[3] 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 of 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.

2.2 Vulnerabilities

Different vulnerabilities were discovered in the Internet Archive's Wayback Machine by Lerner et al. and Berlin. They are Archive-Escapes, Same-Origin Escapes, Archive-Escapes + Same-Origin Escapes, and Anachronism-Injection[11]. 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. Cushman and Kreymer created a shared repository in May 2017 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[13]. The authors provide recommendations on how to avoid such threats. Rosenthal et al., on the other hand, described several threats against the content of digital preservation systems (e.g., web archives)[5]. The authors indicated that designers of archives must be aware of threats, such as media failure, hardware failure, software failure, communication errors, failure of network services, media hardware obsolescence, software obsolescence, operator error, natural disaster, external attack, internal attack, economic failure, and organizational failure.

Several tools have been developed to generate trusted timestamps. For example, OriginStamp allows users to generate a trusted timestamp using blockchain-based networks on any file, plain text, or hash value. The data is hashed in the user's browser and the resulting hash is sent to OriginStamp's server which then will be added to a list of all hashes submitted by other users. Once per day, OriginStamp generates a single aggregated hash of all received hashes. This aggregated hash is 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 (chainpoint.org) and OpenTimestamps (opentimestamps.org), are based on the same concept of using blockchain-based networks to timestamp digital documents. Even though users of these services can pass data by value, they are not allowed to submit data by reference (i.e., passing a URI of a web page). In other words, these tools are not directly timestamping web pages. The only exception is a service established by OriginStamp that accepts URIs from users, but the service is no longer available on the live web at

www.isg.uni-konstanz.de/web-time-stamps/

A number of problems with blockchain-based networks are described by Rosenthal[4]. He indicates that having a large number of independent nodes in the network is what makes it secure, but this is not the case with many blockchain-based services, such as Ethereum (www.ethereum.org).

There are issues related to how web archives preserve and provide access to mementos that make it difficult to generate repeatable fixity information. When serving mementos, web archives often 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., im- ages) 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, and dynamic/randomly-generated content, illustrate how difficult it is to generate repeatable fixity

information. Taking into account all of these archive-related issues, it becomes a challenging problem to distinguish between legitimate changes by archives and malicious changes.

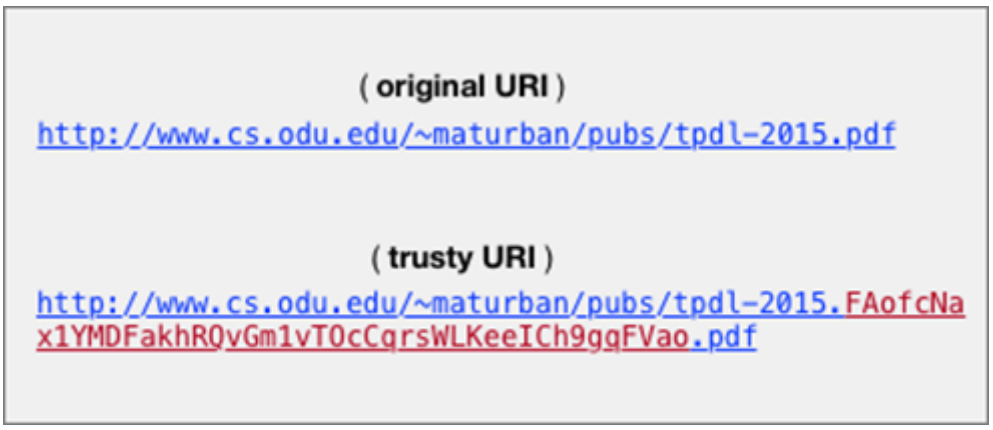


Figure 2.1: An example showing an original URI vs. a trusty URI.

2.3 Trusty URI

Kuhn et al.[6] define a trusty URI as a URI that contains a cryptographic hash value of the content it identifies as shown in Figure 2.1.

With the assumption that a trusty URI, once created, is linked to 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 (e.g., to prevent man-in-the-middle attacks). In their second paper, they introduce two different modules to allow creating trusty URIs on different kinds of content. In 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.

Chapter 3
Methodology

The process of fixity verification of mementos can broadly be described in three phases: 1) generating manifests for mementos, 2) disseminating those manifests into different web archives, and 3) at a later date, generating manifests of the current state and comparing them with their corresponding previously archived versions.

3.1 Manifest Generation

A manifest (identified by URI-Manif) consists of metadata summarizing the fixity information of a memento. A manifest can be generated at or after a memento's creation DateTime. The proposed structure of a manifest file is illustrated in Figure 3.1, and should have the following properties:

@context: It specifies the URI where names used in the manifest file are defined.

created: The creation DateTime of the manifest. It must be equal to or greater than the memento's creation DateTime.

URI-R, URI-M, and Memento-Datetime: It refers to the URI of an original resource, the URI of a memento, and the DateTime when a memento was created, respectively.

Original-Content refers to the raw memento accessing unaltered archived content because archives by default return the memento after transforming its content.


```
{   "@context": "http://manifest.ws-dl.cs.odu.edu/",
    "created": "Sun, 23 Dec 2018 11:43:55 GMT",
    "@id": "http://manifest.ws-dl.cs.odu.edu/manifest/20181223114355/c6ad485819abbe20e37c0632843081710c
            95f94829f59bbe3b6ad3251d93f/d2/https://web.archive.org/web/20181219102034/https://2019.jcdl
            .org/",
    "uri-r": "https://2019.jcdl.org/",
    "uri-m": "https://web.archive.org/web/20181219102034/https://2019.jcdl.org/",
    "memento-datetime": "Wed, 19 Dec 2018 10:20:34 GMT",
    "http-headers": {
        "Content-Type": "text/html; charset=UTF-8",
        "X-Archive-Orig-date": "Wed, 19 Dec 2018 10:20:36 GMT",
        "X-Archive-Orig-link": "<https://2019.jcdl.org/wp-json/>; rel=\"https://api.w.org/\"",
        "Preference-Applied": "original-links, original-content"
    },
    "hash-constructor": "(curl -s '$uri-m' && echo  -n '$Content-Type $X-Archive-Orig-date $X-Archive-O
                         rig-link') | tee >(sha256sum) >(md5sum) >/dev/null | cut -d ' ' -f 1 | paste
                         -d':' <(echo -e 'md5\nsha256') -  | paste -d' ' - -",
    "hash": "md5:969d7aba4c16444a6544bdc39eefe394 sha256:c68a215eb1c3edbf51f565b9a87f49646456369e5179
             1a8610ba666763073a6"       }
```


Figure 3.1: A manifest showing fixity information of the memento https://web.archive.org/web/20181219102034/https://2019.jcdl.org/

hash-constructor: The commands that calculate hashes. The variable $uri-m is replaced with the URI-m value and the selected headers (e.g., $Content-Type) are replaced with the corresponding values in the HTTP headers The hashes are generated on both the HTML of a memento and selected-response headers, and they are calculated using two different hashing algorithms, MD5 and SHA256, so even if the two functions are vulnerable to collision attacks, it becomes difficult for an attacker to make both functions collide at the same time.
hash: The hash values are calculated based on commands defined in the hash-constructor.

3.2 Atomic Dissemination

In the Atomic approach, each memento that we are interested in verifying should have at least one corresponding manifest file containing fixity information of the memento. Once generated, the manifest should be published on the web and disseminated to different web archives. The main concept of this approach is to store the fixity information of a memento in different archives in addition to the archive in which the memento is preserved. This practice is recommended by the TRAC report where content is maintained separately from its fixity information. Disseminating manifests can be archived through four steps:

(1) Push a web page into one or more archives. This will create one or more mementos, URI-M.
(2) Generate a manifest by computing the fixity information of the memento.
(3) Publish the manifest at a well-known location, URI-Manif.
(4) Disseminate the published manifest in multiple archives. This will generate archived manifests, URI-M-Manif.

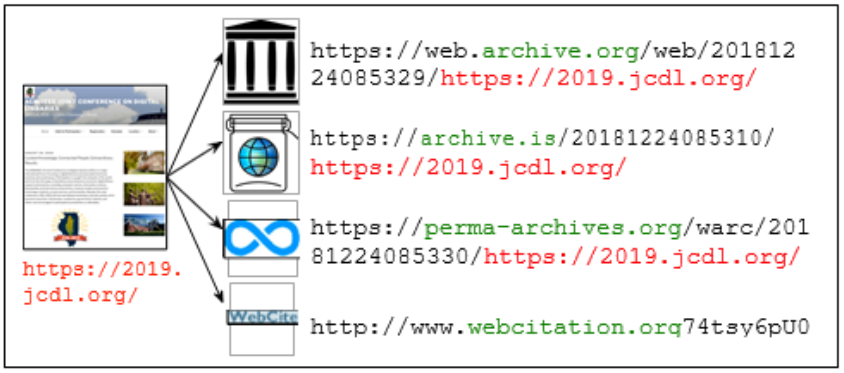


Figure 3.2: A web page is pushed into multiple archives:archive.org, archive.is, perma.cc, and webcitation.org.

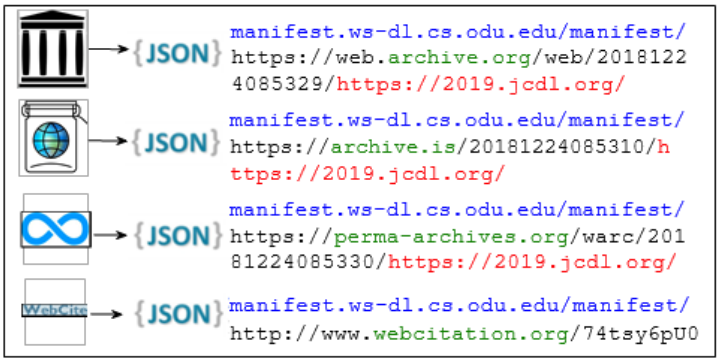


Figure 3.3: Compute fixity and publish it on the web

```
$ curl -sIL https://manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org/web/20181224085329/https
://2019.jcdl.org/ | egrep -i "(HTTP/|^location:)"
HTTP/2 302
location: https://manifest.ws-dl.cs.odu.edu/manifest/20181224093024/8c31ccfbb3a664c9160f98be466b7c9fb9a
1a805804b5052011174be59c6a73a/https://web.archive.org/web/20181224085329/https://2019.jcdl.org/
HTTP/2 200
```

Figure 3.4: The manifest identified with the generic URI redirects to the manifest with the trusty URI

Figure 3.5: Push the fixity information into multiple archives

3.3 Block Dissemination

As opposed to the Atomic approach, in the Block approach we batch multiple manifests together in a single binary-searchable file along with some additional metadata (using the UKVS file format) and add the reference of the previously published latest block. Then, we generate the content-addressable identity of the block, compress it, and archive it into multiple web archives by making it available at a well-known content-addressable URI (and allow people to keep local copies anywhere). While we make a chain of blocks, we are not

attempting to create yet another Blockchain. Manifests' chain of blocks is limited in scope as we do not need to worry about consensus, eventual consistency, or proof-of-work because these blocks are generated and published by a central authority. Linking blocks in a chain using their content-addressable hashes provides tamper-proofing and enables the discovery of previous blocks (starting from the latest or anywhere in the middle of the chain). Additionally, as long as we are depending on an archived page to be available in the archive, we can count on the archived metadata about the page to be available too. Creation and dissemination of manifest blocks are performed in the following steps:

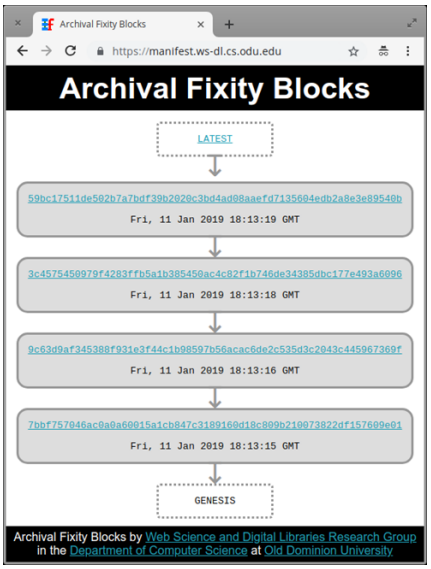


Figure 3.6: Block Dissemination.

3.4 Verifying Fixity of Mementos

Verifying the fixity of a memento in both the Atomic and Block approaches can be achieved through three common steps:

(a) For the given memento, discover one or more manifests URI-Manif. In the Atomic approach, this step requires also discovering archived copies URI-M-Manif of the manifest.
(b) Recompute the current fixity information of the memento.
(c) Compare current fixity information with discovered manifests.

Chapter 4
Evaluation

A study was conducted on 1,000 mementos from the Internet Archive which are a subset of a larger set of URI-Ms involved in a different research project. The size of mementos was not taken into consideration (i.e., the number of embedded resources, such as images and JavaScript/CSS files) because fixity is computed based on only the returned raw HTML content of the base file. The main reason for choosing a small set, only 1,000 URI- Ms, is because the study requires pushing at least 12 manifests for each memento in multiple archives. Sending too many archiving requests to archives might result in technical issues, such as blocking IP addresses. Part of the evaluation is measuring the time it takes to generate, disseminate, and verify manifests in the Atomic and Block approaches. In addition, we want to compare the size of files created in both approaches and whether all mementos are going to be verified successfully.

4.1 Atomic Python Scripts

Python scripts were written for performing different functions:

generate atomic(): Accepts a URI-M and returns the filename of a JSON file containing the fixity information of the memento. The resulting JSON file contains the fixity information including the hash calculated on the returned HTML of the memento.

```
$ python fixity.py generate_atomic https://web.archive.org/web/20051123211159/http://www.whitehouse.org
43423e8ad464461ec196c21033451c07b71b1ec4fbd3be013e3235093abac56b.json
```

Figure 4.1: An example of generating a manifest of a memento.

publish atomic(): Submits a given JSON to the Archival Fixity server at https://manifest.ws-dl.cs.odu.edu. The server will insert @id and created metadata before publishing the new manifest on the web. It returns the generic URI of the manifest URI-Manif and the trusty URI.

```
$ python fixity.py publish atomic 43423e8ad464461ec196c21033451c07b71b1ec4fbd3be013e3235093abac56b.json
http://manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org/web/20051123211159/http://www.whiteho
use.org
http://manifest.ws-dl.cs.odu.edu/manifest/20181212013507/43423e8ad464461ec196c21033451c07b71b1ec4fbd3be
013e3235093abac56b/https://web.archive.org/web/20051123211159/http://www.whitehouse.org
```

Figure 4.2: An example of publishing a manifest at the Archival Fixity server.

disseminate atomic(): Pushes a published manifest into different archives using ArchiveNow

```
$ python fixity.py disseminate atomic http://manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org
/web/20051123211159/http://www.whitehouse.org
http://archive.is/egyVY
https://perma.cc/VMQ3-E45U
http://www.webcitation.org/74bAo5hJ4
https://web.archive.org/web/20181212013856/http://manifest.ws-dl.cs.odu.edu/manifest/20181212013507/434
23e8ad46446lec196c21033451c07b71b1ec4fbd3be013e3235093abac56b/https://web.archive.org/web/2005112321115
9/http://www.whitehouse.org
```

Figure 4.3: An example of disseminating a manifest to four archives.

verify atomic(): Accepts a URI-M, It discovers a manifest closest to the memento's creation DateTime. In addition, the function discovers archived copies of the manifest in the four archives using TimeGates and TimeMaps. Then, it computes the current fixity information using generate atomic(). Finally, it compares current fixity information with the discovered manifests and their archived copies. As a result, for each URI-M, the function returns either "Verified" or "Failed" with other information, such as hash values, URI-Manifs, and URI-M-Manifs.

4.2 Block Python Scripts

generate block(): Accepts multiple JSON files. It generates one or more blocks depending on the selected block size.

disseminate block(): Pushes a block into two archives (archive. org and perma.cc).

verify block(): Accepts a URI-M, and discovers fixity information of the URI-M from the published blocks. Then, it computes current fixity information using generate atomic(). Finally, it compares current and discovered fixity. The function returns either "Verified" or "Failed" with other information, such as hash values.

The selected number of records per block affects the total size of all blocks and the time required to generate these blocks. Figure 4.4 illustrates that creating large blocks with a slowly growing chain is more efficient than a rapidly growing chain of small blocks.

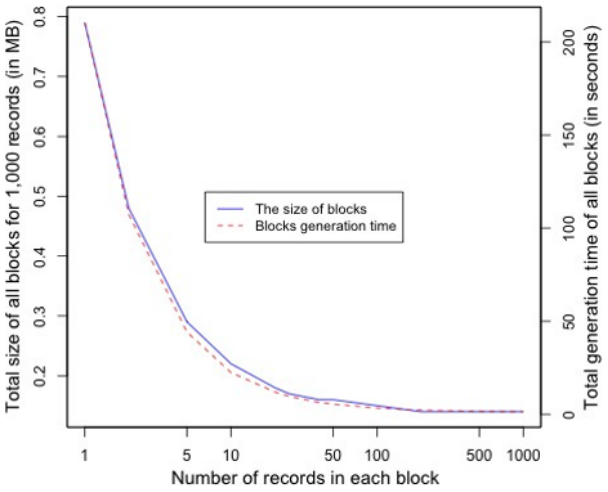


Figure 4.4: The effect of the selected number of records per block.

Chapter 5
Results

Figure 5.1 illustrates the distribution of the average time taken to generate manifests. Three manifests for each memento were generated and calculated the average time, so the total number of generated manifests is 3,000. The manifest generation time includes:

1) Downloading the raw HTML content using the Requests module in Python
2) Calculating the fixity information of the downloaded content
3) Storing the fixity information locally in JSON format. The average size of the generated manifest files is 1,157 bytes. This size represents 2.79% of the actual download HTML content, which is 41,392 bytes on average. The total size of all manifests is 1,156,657 bytes, while the total size of the blocks is 176,128 bytes. This indicates that the Block approach requires less storage space than the Atomic approach to store fixity information of the same number of mementos.

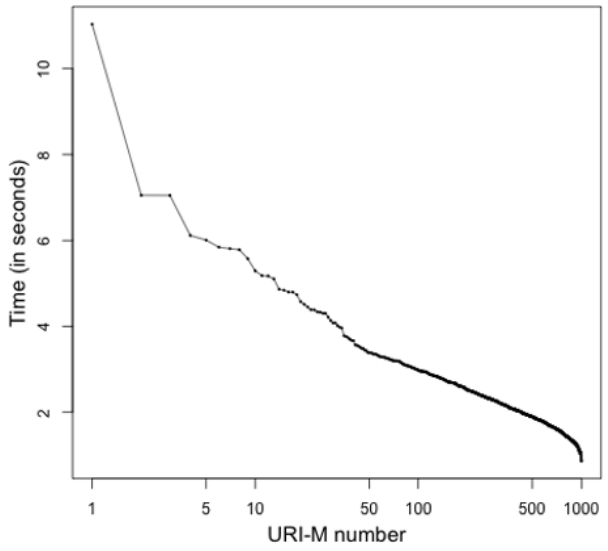


Figure 5.1: Generating manifests of mementos.

```
$ python fixity.py generate_atomic https://web.archive.org/web/20051123211159/http://www.whitehouse.org
0c6c4f7435a79e9a756fb892dc602f9cd1e71a7f74b0346d999b6c1834c703a4.json

$ python fixity.py verify_atomic 0c6c4f7435a79e9a756fb892dc602f9cd1e71a7f74b0346d999b6c1834c703a4.json
Current hash: md5:c4cf615c62a14df9ac2d610873794555 sha256:e5832410226e1e9637eb9fb7c97cf2065851fb106b925
1412e282196be02305c

5 matched manifests:
        http://archive.is/egyVY
        https://perma.cc/VMQ3-E45U
        http://www.webcitation.org/74bAo5hJ4
        http://manifest.ws-dl.cs.odu.edu/manifest/20181212013507/43423e8ad464461ec196c21033451c07b71b1e
        c4fbd3be013e3235093abac56b/https://web.archive.org/web/20051123211159/http://www.whitehouse.org
        https://web.archive.org/web/20181212013856/http://manifest.ws-dl.cs.odu.edu/manifest/2018121201
        3507/43423e8ad464461ec196c21033451c07b71b1ec4fbd3be013e3235093abac56b/https://web.archive.org/w
        eb/20051123211159/http://www.whitehouse.org

0 mismatched manifests:
```

Figure 5.2: An example of verifying the fixity of the memento https://web.archive.org/web/20051123211159/http://www. whitehouse.org. The current fixity information should be generated first. Then, the function verify atomic() finds a published manifest in the Archival Fixity server and its archived versions in web archives. Finally, the function compares current fixity information with the fixity information in the discovered manifest and its archived captures.

As expected, the time for disseminating manifests and blocks was the maximum time compared with other operations, such as generating and verifying manifests. Figure 5.3 shows that pushing manifests into webcitation.org (or WebCite) takes a much longer time than other archives. On average, we wait for 33.82 seconds before WebCite finishes processing an archival request of a manifest, while the manifest disseminating average time drops down dramatically in the other three archives as Table 5.1 indicates. We observed that archive.org and webcitation.org add a few seconds response delay after receiving the first tens of archiving requests. In sum, it takes about 1.25X, 4X, and 36X longer to disseminate a manifest to perma.cc, archive.org, and webcitation.org, respectively, than archive.is, while it takes 3.5X longer to disseminate a block to archive.org than perma.cc. The average dissemination time of blocks in archive.org and perma.cc is shown in Figure 5.4.

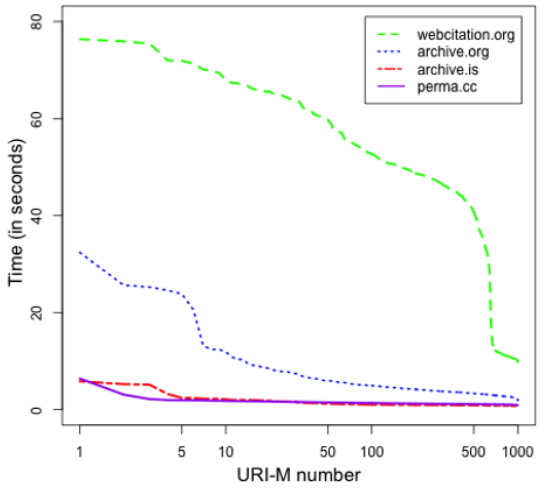


Figure 5.3: Disseminating manifests to four archives.

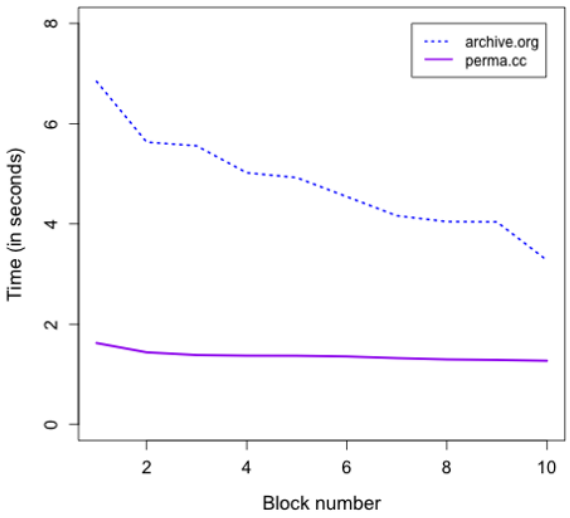


Figure 5.4: Disseminating blocks to two archives.

Operation	archive.is	perma.cc	IA	WebCite
Manifest dissemination	0.94	1.18	3.74	33.82
Block dissemination	-	1.37	4.80	-
Manifest download	0.47	0.60	1.42	4.55
Block download	-	0.30	7.19	-

Table 5.1: Average time (in seconds) for disseminating and downloading of manifests and blocks.

The verification time includes discovering manifests, computing current fixity information, downloading copies of manifests (in the Atomic approach), and comparing manifests. On

average, the verification time of a memento is 6.65 seconds by the Atomic approach and 1.49 seconds by the Block approach, so the Block approach per- forms 4.46X faster than the Atomic approach on verifying the fixity of the memento. Although we have predicted that some mementos might not be verified for reasons like an archive responding with “HTTP 500 Error”, we have not yet encountered any failed cases (i.e., all mementos are verified successfully).

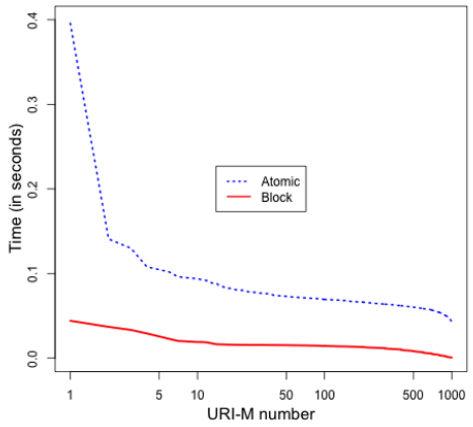


Figure 5.5: Discovering manifests by both approaches.

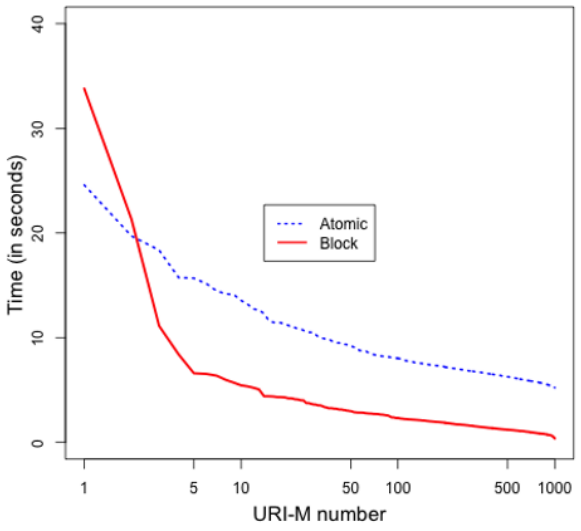


Figure 5.6: Verifying mementos by both approaches.

Figure 5.5 shows the time required to discover manifests of each memento from the Archival Fixity server. Figure 5.6 illustrates the total time for verifying the fixity of all mementos by both approaches.

Chapter 6
Conclusion

Most web archives do not allow users to access fixity information. Even if fixity information is accessible, it is provided by the same archive delivering content. In this proposal, we have described two approaches, Atomic and Block, for generating and verifying the fixity of archived web pages. The proposed work does not require any change in the infrastructure of web archives and is built based on well-known standards, such as the Memento protocol. While a central service is used to create manifests, this approach does not exclude additional, centralized manifest servers, possibly tailored to specific communities. The Block approach creates fewer resources in archives and reduces fixity verification time, while the Atomic approach has the ability to verify the fixity of archived pages even without involving the Archival Fixity server. On average, it takes about 1.25X, 4X, and 36X longer to disseminate a manifest to perma.cc, archive.org, and webcitation.org, re- respectively, than archive.is, while it takes 3.5X longer to disseminate a block to archive.org than perma.cc. The Block approach performs 4.46X faster than the Atomic approach in verifying the fixity of archived pages.

The Atomic and Block approaches can be adopted to verify the fixity of particular archived web pages with important content. Some future improvements can be applied to those approaches so they become scalable and can work with any number of mementos. Varying or increasing the block size in the Block approach might be one potential solution to improve its performance and reduce the number of resources created in archives. Caching archived manifests in the Archival Fixity server should also improve the performance of the two approaches, so instead of discovering those manifests from the archives, we may use cached copies in the Archival Fixity server.

References:

[1] 2017. WARC file format. https://www.iso.org/standard/68004.html. ISO 28500:2017

[2] Sawood Alam. 2013. HTTP Mailbox - Asynchronous Restful Communication. Mas- ter's thesis. Old Dominion University. https://doi.org/10.25777/wh13-fd86

[3] Sawood Alam. 2019. Unified Key Value Store. https://github.com/oduwsdl/ORS/blob/master/ukvs.md. (January 2019).

[4] Sawood Alam, Michele C. Weigle, and Michael L. Nelson. 2019. MementoMap Framework for Flexible and Adaptive Web Archive Profiling. In Proceedings of the 19th ACM/IEEE Joint Conference on Digital Libraries (JCDL).

[5] Shadi Aljawarneh, Christopher Laing, and Paul Vickers. 2008. Design and experimental evaluation of Web Content Verification and Recovery (WCVR) system: A survivable security system. In Proceedings of the 3rd Conference on Advances in Computer Security and Forensics (ACSF).

[6] Mohamed Aturban and Sawood Alam. 2019. Archival Fixity. https://github.com/oduwsdl/archival-fixity. (2019).

[7] Mohamed Aturban, Sawood Alam, Michael L. Nelson, and Michele C. Weigle. 2019. Web Archiving Fixity Verification Framework. In Proceedings of the 19th ACM/IEEE Joint Conference on Digital Libraries (JCDL).

[8] 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).

[9] Mohamed Aturban, Michael L. Nelson, and Michele C. Weigle. 2017. Difficulties of Timestamping Archived Web Pages. Technical Report arXiv:1712.03140.

[10] Mohamed Aturban, Michael L Nelson, and Michele C Weigle. 2018. It is Hard to Compute Fixity on Archived Web Pages. In Proceedings of the Workshop on Web Archiving and Digital Libraries (WADL) held in conjunction with the 18th ACM/IEEE Joint Conference on Digital Libraries (JCDL).

[11] Jefferson Bailey. 2014. Protect Your Data: File Fixity and Data Integrity

[12] 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. (Feb- ruary 2017).

[13] John A. Berlin. 2018. To Relive the Web: A Framework for the Transformation and Archival Replay of Web Pages. Master's thesis. Old Dominion University. https://doi.org/10.25777/n8mg-da06

[14] Chris Butler. 2018. Addressing Recent Claims of "Manipulated" Blog Posts in the Wayback Machine. http://blog.archive.org/2018/04/24/ addressing-recent-claims-of-manipulated-blog-posts-in-the-wayba ck-machine/. (2018).

[15] Miguel Costa, Daniel Gomes, and Ma´rio 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

[16] 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).

[17] 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).

[18] Caleb Ecarma. 2018. EXCLUSIVE: Joy Reid Claims Newly Discovered Homophobic Posts From Her Blog Were "Fabricated". https://www.mediaite.com/online/exclusive-joy-reid-claims-newly-discovered-homophobic-posts-from-her-blog-were-fabricated/. (2018).