Summer 8-2020

A Framework for Verifying the Fixity of Archived Web Resources

Mohamed Aturban
Old Dominion University, maturban@yahoo.com

A FRAMEWORK FOR VERIFYING THE FIXITY OF ARCHIVED WEB RESOURCES

by

Mohamed Aturban
B.S. June 2002, University of Tripoli, Libya
M.S. May 2011, New Mexico State University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2020

Approved by:

Michele C. Weigle (Director)

Michael L. Nelson (Member)

Sampath Jayarathna (Member)

Jian Wu (Member)

M'Hammed Abdous (Member)

ABSTRACT

A FRAMEWORK FOR VERIFYING THE FIXITY OF ARCHIVED WEB RESOURCES

Mohamed Aturban
Old Dominion University, 2020
Director: Dr. Michele C. Weigle

The number of public and private web archives has increased, and we implicitly trust content delivered by these archives. Fixity is checked to ensure that an archived resource has remained unaltered (i.e., fixed) since the time it was captured. Currently, end users do not have the ability to easily verify the fixity of content preserved in web archives. 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? In order for the users of web archives to verify that archived web resources have not been altered, they should have access to fixity information associated with these resources. However, most web archives do not allow accessing fixity information and, more importantly, even if fixity information is available, it is provided by the same archive delivering the resource, not by an independent archive or service.

In this research, we present a framework for establishing and checking the fixity on the playback of archived resources, or mementos. The framework defines an archive-aware hashing function that consists of several guidelines for generating repeatable fixity information on the playback of mementos. These guidelines are results of our 14-month study for identifying and quantifying changes in replayed mementos over time that affect generating repeatable fixity information. Changes on the playback of mementos may be caused by JavaScript, transient errors, inconsistency in the availability of mementos over time, and archive-specific resources. Changes are also caused by transformations in the content of archived resources applied by web archives to appropriately replay these resources in a user's browser. The study also shows that only 11.55% of mementos always produce the same fixity information after each replay, while about 16.06% of mementos always produce different fixity information after each replay. The remaining 72.39% of mementos produce multiple unique fixity information. We also find that mementos may disappear when web archives move to different domains or archives.

In addition to defining multiple guidelines for generating fixity information, the framework introduces two approaches, *Atomic* and *Block*, that can be used to disseminate fixity

information to web archives. The main difference between the two approaches is that, in the *Atomic* approach, the fixity information of each archived web page is stored in a separate file before being disseminated to several on-demand web archives, while in the *Block* approach, we batch together fixity information of multiple archived pages to a single binary-searchable file before being disseminated to archives. The framework defines the structure of URLs used to publish fixity information on the web and retrieve archived fixity information from web archives. Our framework does not require changes in the current web archiving infrastructure, and it is built based on well-known web archiving standards, such as the Memento protocol. The proposed framework will allow users to generate fixity information on any archived page at any time, preserve the fixity information independently from the archive delivering the archived page, and verify the fixity of the archived page at any time in the future.

I dedicate this dissertation to the soul of my father who passed away on September 04, 2017 for his endless support till his last moment, and to my mother for her continuous love and support. It is also dedicated to my beloved wife for her unwavering love and support, and to my wonderful son and my two beautiful daughters.

ACKNOWLEDGEMENTS

All praise to God (Alhamdulelah), the most Gracious and most Merciful, by whose grace and blessing I have been able to take and complete this academic journey. I thank all people who had supported me during my study.

Firstly, I would like to express my sincere gratitude to my advisor Dr. Michele Weigle and co-adviser Dr. Michael Nelson for the continuous support of my research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this dissertation.

I want to express my deep appreciation for the members of the committee Dr. Jian Wu, Dr. Sampath Jayarathna, and Dr. M'Hammed Abdous for their support and feedback to improve the dissertation. I am thankful for Dr. Hussein Abdel-Wahab, who passed away in 2016. He was always welcoming me to his office, listening and providing valuable advice and guidance. I would like also to thank Dr. Mohammad Zubair who helped me to join the Department of Computer Science in 2012, Dr. Desh Ranjan and Dr. Balsa Terzic who helped me to start my research.

I am grateful to the former and current members of the Web Science and Digital Library group for their support and perceptive discussions: Ahmed, Yasmin, Justin, Hany, Chuck, Mat, Lulwah, Alex, Shawn, Scott, Sawood, Nauman, Kritika, Hussam, and the rest of the WS-DL group. I also would like to thank Dr. Herbert Van de Sompel from Data Archiving and Networked Services (DANS) in The Netherlands and Dr. Martin Klein from Los Alamos National Lab for providing great feedback, help, and advice on my research and studies. I thank Ben Steinberg from the Perma.cc web archive for the generous support that allowed me to conduct several studies using their services. I would like also to acknowledge and thank my colleagues at University of Tripoli for their help, support, and great discussions: Mustafa Elfituri, Ahed Elmsallati, Hisham Benotman, Ibrahim Ben Mustafa, Abdussalam Alawini, Akram Milad, Adel Ali, and Ali Fanan.

I would like to thank the Department of Computer Science at Old Dominion University for providing me with opportunities and support to conduct my research. I am also thankful to University of Tripoli and Ministry of Higher Education and Scientific Research (MOHESR) in Libya for their support from August 2012 to November 2016. I would like also to thank the funding agencies that supported my research including the Andrew W. Mellon Foundation (AMF) grant 11600663, the National Science Foundation (NSF) grant III 1526700, and the National Endowment for the Humanities (NEH) grant HK-50181-14.

I would like also to express my deepest appreciation to all my family and friends for their continuing support and encouragement.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Web pages often change or even disappear over time due to the ephemeral nature of the web. These changes in existing web pages have had major implications on the integrity of information published online, such as scholarly documents, news articles, and blogs. For example, Klein et al. [9] conducted a study on over one million references from scientific articles and found that one in five articles suffers from Reference Rot, referring to links to web resources that no longer exist (link rot) or that have significantly modified content (content drift). To mitigate the impact of changing web pages, web archives, such as the Internet Archive (IA) [10], UK Web Archive [11], and `perma.cc` [12], have been established with the goal to preserve the web to allow access to prior states of web resources. Unlike live web pages, which are expected to change, archived web pages, or mementos (archived versions of an original web page [13]), should remain unchanged from the time of their capture. In general, 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 [14, 15], we should consider the question of validity by checking the fixity of archived web pages to ensure that those resources have remained unaltered since the time they were captured.

Figure 1 shows a motivating example that clarifies the problem we are trying to address and highlights the importance of verifying the fixity of mementos. A memento is captured by a private web archive, *Michael's Evil Wayback*[1], on July 17, 2017 at 18:51 GMT. This memento is a copy of the original web page:

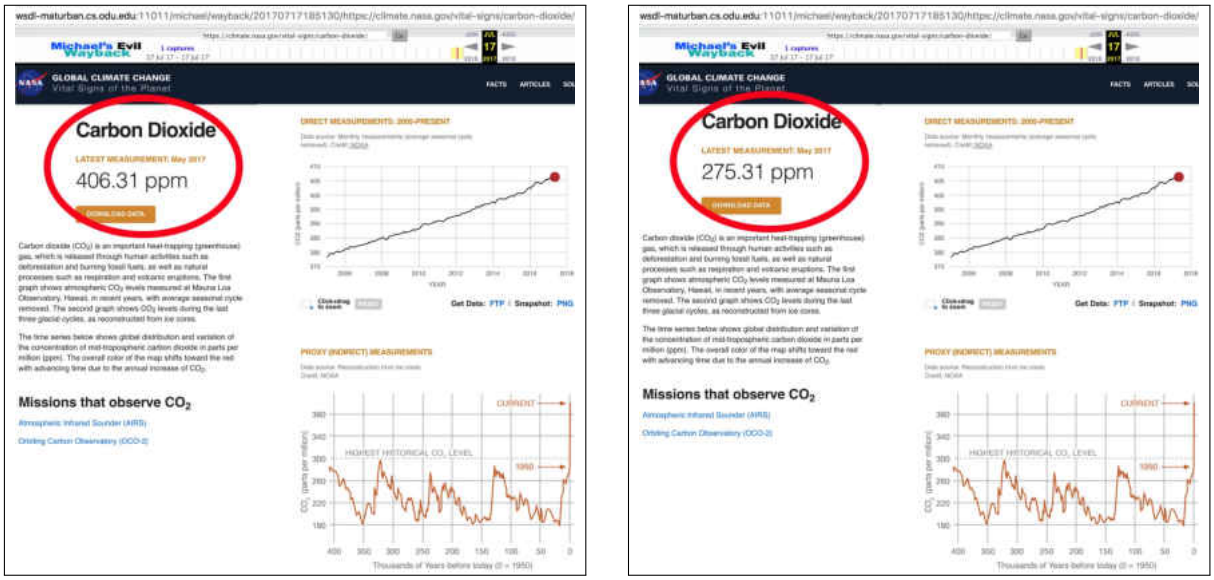
`https://climate.nasa.gov/vital-signs/carbon-dioxide/`

Figure 1 demonstrates an unexpected result. When replaying the memento in August 2017, the CO_2, the carbon dioxide level in the Earth's atmosphere, was 406.31 *ppm*, but when replaying the same memento in October 2017, CO_2 became 270.31 *ppm*. So, what was the value on the original page in July 2017? Which memento represents the"real" archived

[1]We established this archive to demonstrate different scenarios

page? Without a mechanism to verify the fixity of this memento, we will not be able to detect that the content of the memento has been altered.

Another 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 [16, 17]. 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 [18]. Even though the Internet Archive denied that their archived pages had been hacked [19], a stronger case could be made if we had an independent service to verify that those archived blog posts had not changed since they were captured by the archive.



(a) Replaying the archived page in August 2017 (CO_2 was 406.31 *ppm*).

(b) Replaying the same memento in October 2017 (CO_2 became 270.31 *ppm*.

Fig. 1: Replaying a memento at two different times. The content of the memento was altered, and without a way to verify the fixity of this memento the user can not determine which one is the real memento.

1.2 THE IMPORTANCE OF VERIFYING THE FIXITY OF MEMENTOS

In the context of web archiving, fixity ensures that mementos have remained unaltered since the time they were captured [20]. The final report of the PREMIS Working Group [21] defines information used for fixity as "information used to verify whether an object has

been altered in an undocumented or unauthorized way." A part of the problem is the lack of standared techniques that users can apply to verify the fixity of web content in general [22–24]. Jinfang Niu [25] 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, WAX[2], and Government of Canada Web Archive[3], have a disclaimer [26] stating that they are not responsible for the reliability of the archived content they provide, as shown in Figure 2. It is important to verify the fixity of mementos for the following three reasons:

1. The number of public and private web archives is increasing [14, 15], and we may not have the same level of trust in all of these archives.

2. There is a current trend of using web archives for evidentiary purposes in court cases or to generally prove the existence of a web resource at a particular time in the past [16, 27–35] (refer to Chapter 1.3.1).

3. There are different security threats against web archives [3, 36–43] that not only affect accessibility to archived collections but also would change the representation of replayed archived pages over time (refer to Chapter 3.1).

1.3 THE IMPORTANCE OF WEB ARCHIVES

Because web archives preserve important web resources, we need to verify the fixity of these resources. In this section, we show multiple examples that emphasize the importance of web archives.

1.3.1 EVIDENTIARY PURPOSES IN COURT CASES

There is a current trend of using web archives for evidentiary purposes in court cases. For example, the Internet Archive was used as evidence in the case of *Marten Transport v. PlatForm Advertising* [30] where Marten Transport claimed that PlatForm Advertising used their name without authorization in advertising truck driver jobs. The judge used archived pages of PlatForm Advertising's websites from The Internet Archive's Wayback Machine [44] as a reliable source of evidence [28,29] after hearing testimony from an employee

[2]The archive `http://wax.lib.harvard.edu/collections/home.do` which now has moved to `https://archive-it.org/organizations/935`

[3]`http://www.collectionscanada.gc.ca/webarchives/index-e.html`

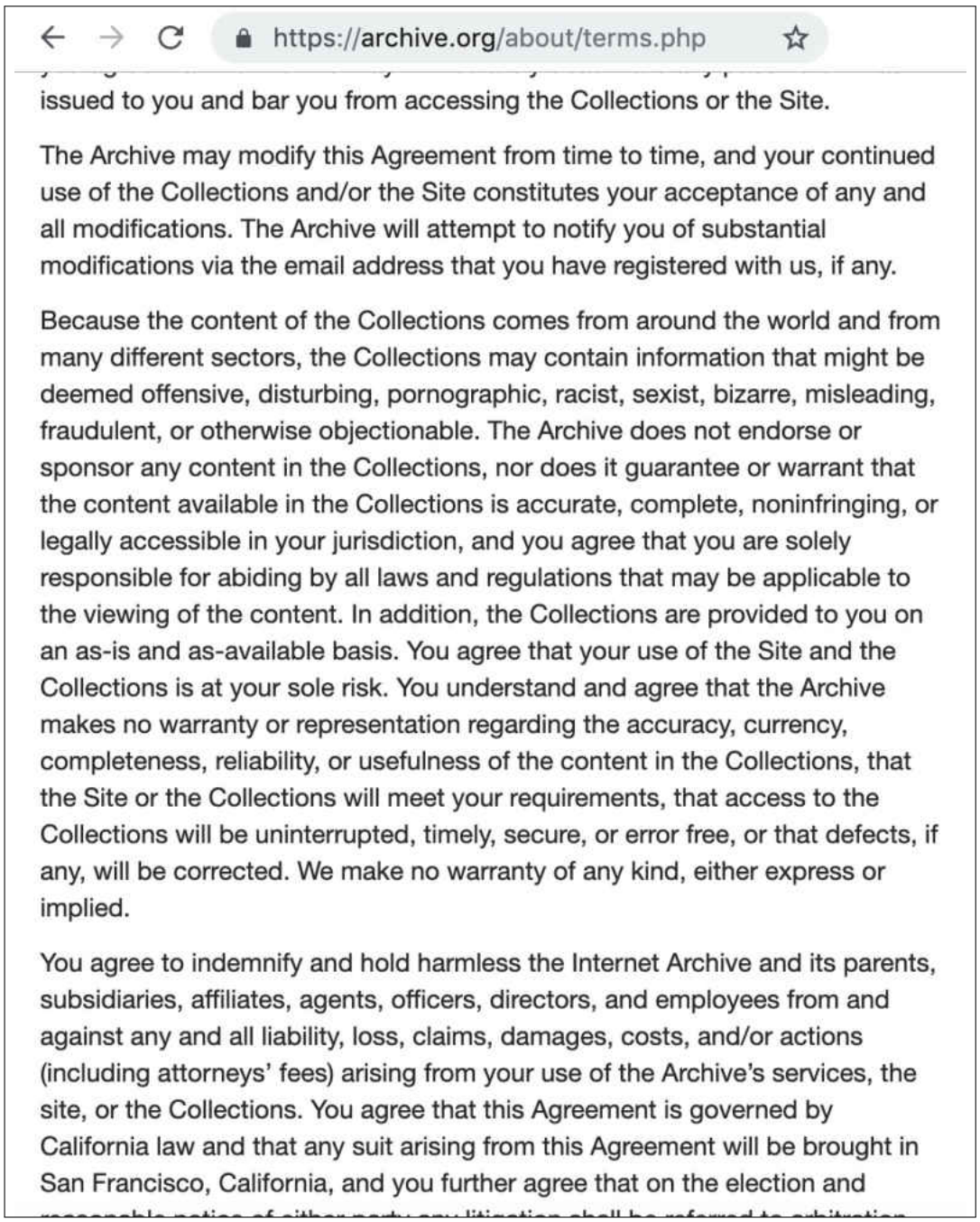


Fig. 2: A disclaimer from the Internet Archive stating that the archive is not responsible for the reliability of the archive resources.

at the Internet Archive. One of the points PlatForm Advertising argued about was that the testimony was not sufficient for multiple of reasons inclusing the quality of the archived content as embedded images did not appear in some archived pages. The court rejected this

argument noting that Marten's material was on PlatForm Advertising's website on certain dates, so even if Internet Archive was not able to capture everything within a web page, it does not indicate that The Internet Archive adds material that was not on the PlatForm Advertising's website other than the Wayback Machine's banner and some code that is necessary for the tool to work. Eltgrowth [27] outlines several judicial decisions that involve evidence (i.e., archived web pages) taken from the Internet Archive (e.g., court cases like *Telewizja Polska USA, Inc. v. Echostar Satellite Corp* [45], and *St. Luke's Cataract & Laser Institute v. James C. Sanderson* [46]).

1.3.2 PRESERVING FAKE NEWS AND IMPORTANT NEWS ARTICLES

The problem of "*fake news*" [47] posted on Twitter[4], Facebook[5], and other social media sites has been receiving increased attention recently. The online misinformation might have impacted crucial events, such as the 2016 US Presidential election. For example, according to *The New York Times* [48], Gen. Michael Flynn, who was selected by President Donald Trump to be his national security adviser, has used social media to promote a number of conspiracy theories about Hillary Clinton. Figure 3(b) shows one example of Flynn's unproven stores on Twitter—he wrote "U decide - NYPD Blows Whistle on New Hillary Emails: Money Laundering, Sex Crimes w Children, etc...MUST READ!". This tweet is no longer found in the live web. If we open the tweet

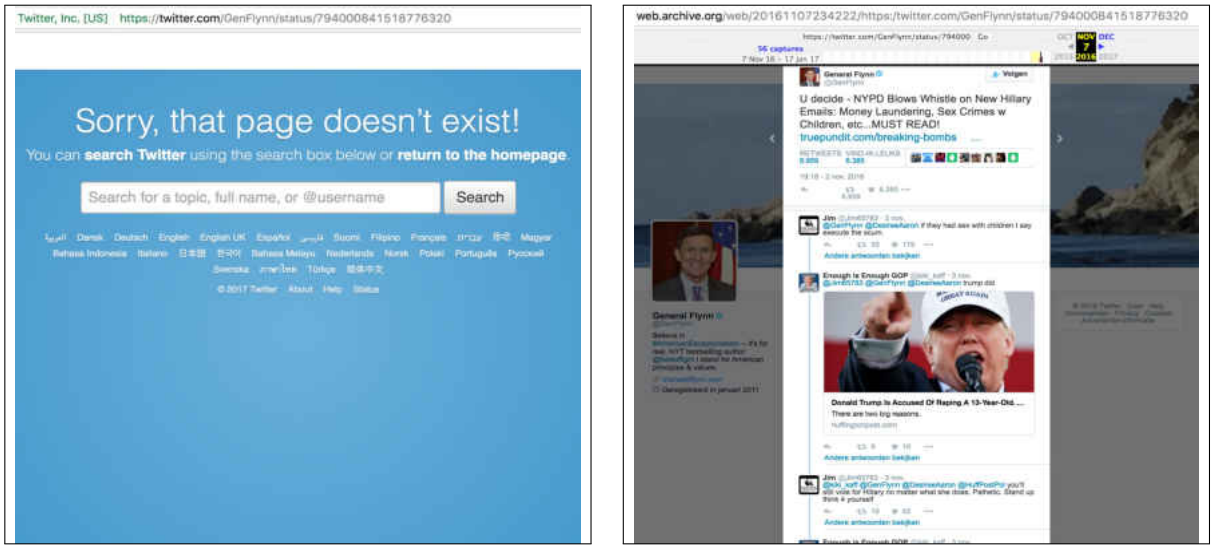
`https://twitter.com/GenFlynn/status/794000841518776320`.

in a web browser, we will get "Sorry, that page doesn't exist!" in return, as shown in Figure 3(a). An article from CNN [34] mentioned that Flynn has deleted this tweet from his own Twitter account, but it can still be viewed in the Internet Archive (Figure 3(b)).

1.3.3 PROVIDING INFORMATION ABOUT CERTAIN INCIDENTS OR CRIMES

A Boeing 777 belonging to Malaysia Airlines took off from Amsterdam, Netherlands on Thursday, July 17, 2014, for a 12-hour flight (MH-17) to Kuala Lumpur, Malaysia. The plane crashed in the Donetsk area of Ukraine after flying for about three hours. All 283 passengers were killed. The Internet Archive has been frequently visiting and capturing one of the main Russian social media websites VKontakte[6]. In these archived collections,

[4] `https://twitter.com`
[5] `https://www.facebook.com`
[6] `https://vk.com/`



(a) The tweet `https://twitter.com/GenFlynn/status/794000841518776320` is disappeared from the live web .

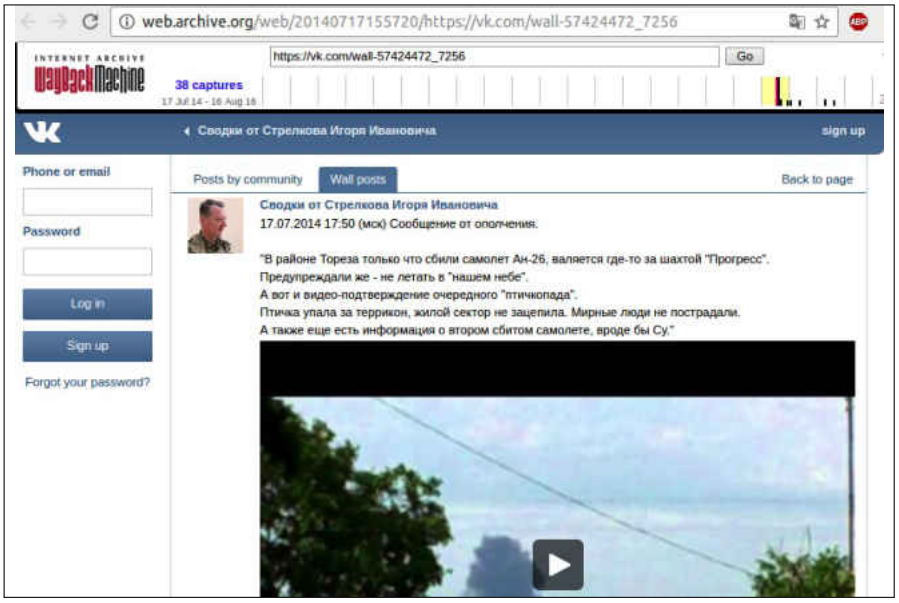
(b) The tweet is found in the Internet Archive at `https://web.archive.org/web/20161107234222/https://twitter.com/GenFlynn/status/794000841518776320`.

Fig. 3: Gen. Flynn's tweet is deleted but found in the Internet Archive.

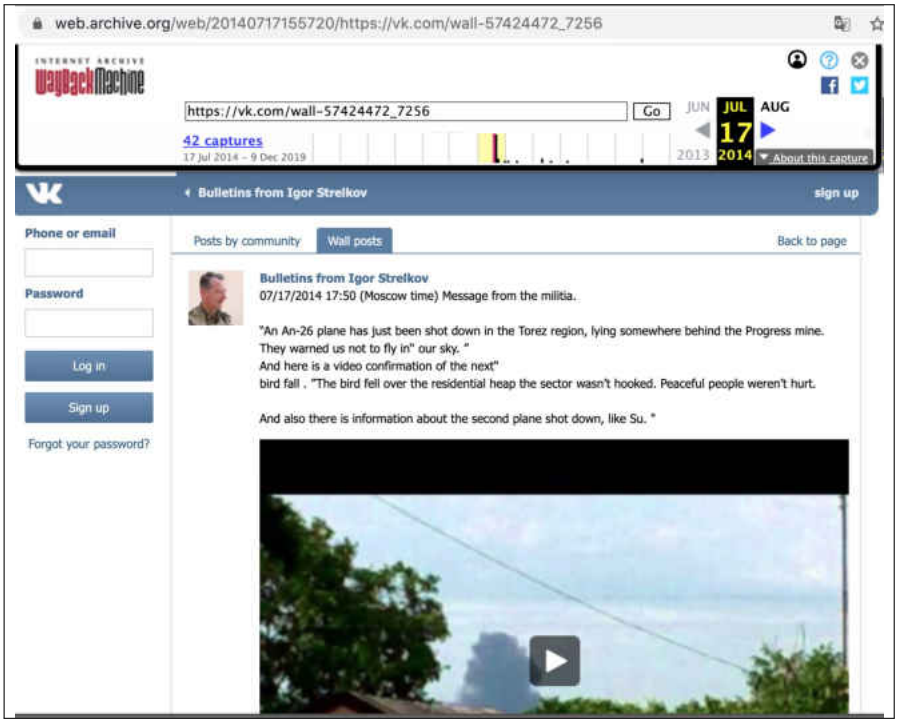
evidence (Figure 4(a)) was found which may identify who was responsible for the downing. One of the posts contains information translated as "An AN-26 plane has been shot down ..." (Figure 4(b)) with a video that was posted by Igor Girkin, a Russian army artillery veteran, at the time when the crash was reported by the Malaysia Airlines [32]. This post was deleted two hours after its creation, and it is no longer available on live web. Arthur Bright [31] used an archived version of the post from the Internet Archive's Wayback Machine [44] (Figure 4) to present a case that Igor Girkin might be responsible for the crash of the Malaysian plane.

1.4 GENERATING FIXITY INFORMATION ON ONLY THE BASE HTML PAGE

One conventional technique for verifying the fixity of archived resources is to generate a unique string, or hash value, that represents the content of an archived resource at a particular time using cryptographic hash algorithms, such as `MD5` or `SHA-256`. The resulting hash values cannot be converted back to the original content, and their output string has a fixed size. Other simple fixity-based approaches includes checking the file size, or the file count. Figure 5 shows an example where the `cURL` command downloads the HTML of the



(a) The archived page with the original text.



(b) The archived page translated by Google's Chrome browser.

Fig. 4: The archived page captured by the Internet Archive might lead to who was responsible for the Malaysian plane crash in Ukraine.

memento

`http://wsdl-maturban.cs.odu.edu:11011/michael/wayback/20170717185130/https://climate.nasa.gov/vital-signs/carbon-dioxide/`

a memento of `climate.nasa.gov/vital-signs/carbon-dioxide/` on 2017-07-17 at 18:51:30 GMT. Then the hashing function `sha256sum` generates a `SHA-256` hash on the resulting HTML.

```
$ curl -s http://wsdl-maturban.cs.odu.edu:11011/michael/wayback/2017071
  7185130/https://climate.nasa.gov/vital-signs/carbon-dioxide/ | shasum
  -a 256

e834c71aefda284fe03a4eed4e8cb78ea581537ba8884aecec29bd2d66cbf521 -
```

Fig. 5: cURL command to generate a SHA-256 hash value on the HTML content only.

By periodically generating and storing hash values separately from the original content, we can compare these hashes to identify if the resource has been changed from a prior state. For example, consider a scenario illustrated in Figure 6 where, in August 2017, a user generates a `SHA-256` hash on the base HTML file of the memento (from *Michael's Evil Wayback*):

`http://wsdl-maturban.cs.odu.edu:11011/michael/wayback/20170717185130/https://climate.nasa.gov/vital-signs/carbon-dioxide/`

resulting in a hash value that ends with `f521`. Two months later (i.e., October 2017), the user recalculates the hash (i.e., following the same commands shown in Figure 5) on the same memento. This results in a different hash value that ends with `3790`.

One possible cause of getting different hash values is demonstrated with the "black hat" in Figure 6: Michael's Evil Wayback has tampered with the memento by changing the value of CO_2 from 406.31 *ppm* (Figure 45(a)) to 270.31 *ppm* (Figure 45(b)). By applying the simple approach of computing hashes on the HTML content, the user becomes aware that the retrieved content in October 2017 cannot be identical to the content retrieved a couple of months earlier.

There is a major issue with considering only the HTML content in computing the fixity information. Ainsworth et al. [49] define a composite memento as the set of all resources comprising an archived page including the base HTML file, images, style sheets, JavaScript files, iframes, and others. If a hash is not calculated on a composite memento, any change in one or more embedded resources (e.g., the image illustrated in Figure 7), will not be

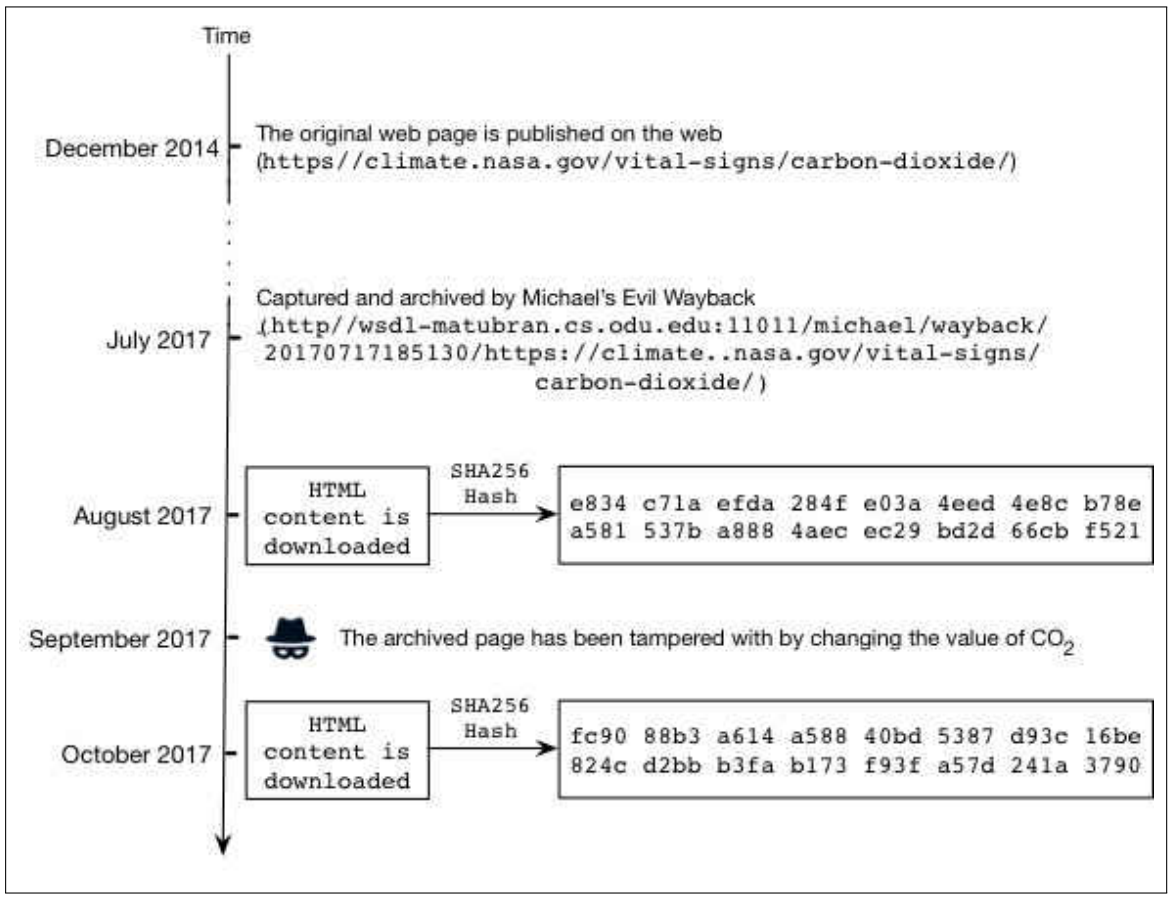
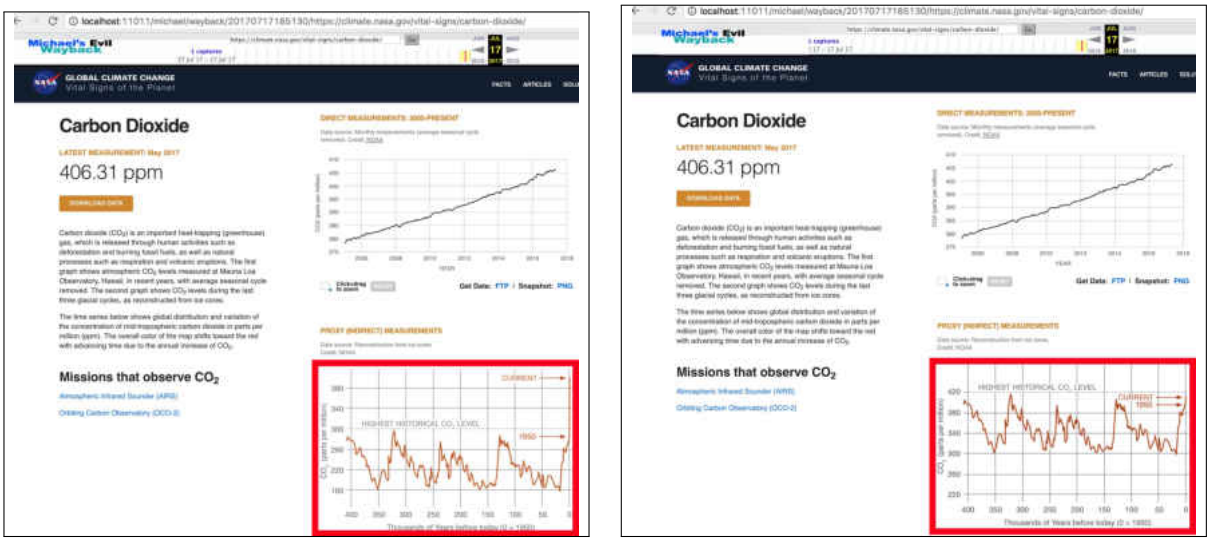


Fig. 6: The content of memento has been tampered with, and the simple approach of generating hashes based on the HTML content successfully detected that the page has been altered.

detected. In other words, this approach will produce false negative results indicating that the memento has not been altered. The approach works properly on HTML content only or on mementos that consists of only one resource, such as PDFs or ZIP files.

Even when only the HTML content is considered in computing the fixity, there are scenarios where getting repeatable fixity information is complicated for different reasons related to the playback of mementos (as explained in Chapter 6), such as content modified by a third-party service before reaching to the client, unavailability of mementos (i.e., indexes) sometimes, and transient errors. Getting repeatable fixity information will be even more complicated if all resources comprising a composite memento are included in computing the fixity (as explained in Chapter 1.5 and Chapter 6).

There are two ways for us to obtain fixity information of mementos. First, we can



(a) Replaying the memento before the image has been tampered with.

(b) Replaying the memento after the image has been tampered with.

Fig. 7: The simple approach of generating hashes on only the HTML content will not detect changes that affect embedded resources, such as the image of the historical CO_2 level (marked in red).

calculate fixity information on the playback of a memento (client-side) as shown in Figure 5. The second way is to get the information from the archive, but not many archives provide access to the fixity information. For example, the Internet Archive CDX server [50] allows accessing fixity information extracted from the WARC (Web ARChive) files (WARC format is described in Chapter 2.2.2). Ideally, fixity information should also be stored in independent archives, but this is not yet available for general web pages.

1.5 DIFFICULTIES OF GENERATING REPEATABLE FIXITY INFORMATION

We often get repeatable hash values computed on mementos that consists of one resource, such as a PDF or a ZIP, because the main characteristic of such files is that they can be downloaded with one HTTP request or in self-contained files, but generating repeatable hashes on composite mementos is more complicated. Figure 8 shows a shell script that generates one aggregated hash for a composite memento through the following steps:

1. Download a composite memento (the base HTML file and all embedded resources) using Wget [51] (line 1 in Figure 8). The output files will be stored in a directory

hierarchy related to the paths of the files in the server. For example, the file *index.html* identified by the URI `https://www.example.com/path.to.file/index.html` will be stored in the sub-directory *path.to.file* which is located under the directory named *www.example.com*.

2. Generate a SHA-256 hash on the entity body of each resource. The set of resulting hashes will be aggregated and stored in the file *allhashes.txt* (lines 5-10).

3. Read the hash values from *allhashes.txt*, use them as input to the hashing command `shasum -a 256` which will generate one aggregated SHA-256 hash value that represent the content of the composite memento (line 12).

```
1 FILE=$(wget --continue --unlink --page-requisites --timestamping -e
    robots=off -k --user-agent="Mozilla/5.0 (Macintosh; Intel Mac OS X
    10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.90
    Safari/537.36"  $1 2>&1 | egrep 'Saving to: .*' | sed -n 's/.*âĂŸ
    \([^ ]*\)âĂŹ/\1/p' | tr -d '')
2 echo ""
3 echo "  SHA256 hash                                  Resource"
4 echo "-----------------------------------------------------------------"
5 for TARGET in $FILE; do
6   CONT=$(cat $TARGET)
7   HASH=$(echo "$CONT" | shasum -a 256 | cut -d' ' -f1)
8   echo "$HASH $TARGET"
9   echo "$HASH" >> "allhashes.txt"
10 done
11 echo "-----------------------------------------------------------------"
12 FINAL_HASH=$(cat "allhashes.txt" | shasum -a 256 | cut -d' ' -f1)
13 echo "Aggregated hash: $FINAL_HASH"
14 echo ""
```

Fig. 8: The shell script *aggregated_hash.sh* for generating a single hash on the content of a composite memento by aggregating all hash values of the embedded resources in a single temporary file and hashing the file. This shell script is a modified version of the original script written by Branwen [1].

Figure 9 shows an example of generating one aggregated hash value on a composite memento using the shell script *aggregated_hash.sh*. The root URI-M[7] of the composite memento is:

`https://web.archive.org/web/20180104003111/https://climate.nasa.gov/vital-signs/carbon-dioxide/`

[7]URI-M identifies an archived version (memento) of an original resource (as described in Chapter 2.2.3)

The final aggregated hash of the composite memento ends with `83a1984`. We expect to obtain the same aggregated hash each time we run this script (Figure 9) on the same memento. However, after running this script on the same memento after a few minutes, we get a different aggregated hash value, ending with `a9974a3`. We obtain different hash values on composite mementos for different reasons related to how the playback of mementos works (as described in Chapter 2.2.5 and Chapter 6). In Sections 1.5.1, 1.5.2, and 1.5.3, we briefly describe three of those playback-related issues causing different fixity information.

1.5.1 ARCHIVES TRANSFORM THE CONTENT OF CAPTURED WEB PAGES

Archives transform the content of the captured web pages to appropriately replay them in a user's browser. For example, archives add their own banners to provide metadata about both the memento being viewed and the original page. Archives also rewrite links of embedded resources (e.g., images in a page) so that these resources are retrieved from the archive, not from the original server. Figure 10 shows an example of the live web page:

`http://money.cnn.com/2018/01/27/technology/future/spacex-falcon-heavy-everything-you-need-to-know/index.html`

from `cnn.com` and the replay of three mementos of the web page from different archives. Each archive adds its banner to the original content. Including archive-specific content (e.g., banners which contain dynamic information reflecting the current state of the archive) in the hash calculation will prevent generating repeatable hashes. For example, the information in the banner in Figure 11 that the archive `webarchive.proni.gov.uk` conveyed in 2016 for the memento:

`http://webarchive.proni.gov.uk/20150826163149/http://www.ulster.ac.uk`

is different from the information conveyed in 2017 for the same memento. In general, we should exclude any archive-specific content in hash calculation.

1.5.2 ISSUES IN RECONSTRUCTING COMPOSITE MEMENTOS

The HTTP Archive [52] reports that the median number of requests that comprises a web page is 75. Because we are interested in computing fixity information on composite mementos (by including all embedded resources), any small change in the replay process affecting either the base HTML file or an embedded resource will result in different fixity information. For example, we replayed the memento

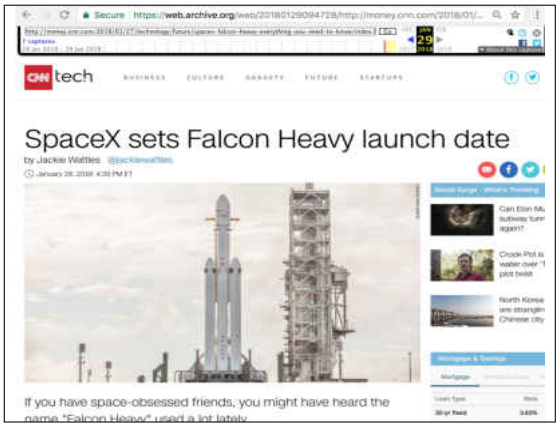
```
$ ./aggregated_hash.sh https://web.archive.org/web/20180104003111/
    https://climate.nasa.gov/vital-signs/carbon-dioxide/

 SHA256 hash                                                    Resource
-------------------------------------------------------------------------
9cb44bb4cf52252aee9602e4036e63aa968bd8e777c5a99fb905c246b58282e4 web.
    archive.org/web/20180104003111/https:/climate.nasa.gov/vital-signs/
    carbon-dioxide/index.html
526abb641edc0696331c1948c8be4394fc8570663488da99ad6493567c7eae7e web.
    archive.org/_static/js/wbhack.js
85283789b3433b7e9ccc48a181320121db1ac6e914d5ada6c45d4b872f8b9e6f web.
    archive.org/_static/css/banner-styles.css
a7070efc4a17d82f068df64fb5c2de15e1e061a46eba64251d243d69081153b2 web.
    archive.org/web/20180104003111im_/https:/climate.nasa.gov/favicon.
    ico
ffaaf8fc3ebcdde5890547115d3c07365a266dad416f2a30af9b5469c424a469 web.
    archive.org/web/20180104003111im_/https:/climate.nasa.gov/system/
    internal_resources/details/original/417_1263_banner-science-1600x500
    .jpg
4297294d0d438ad0cda59975edf024585f06e3fe1df4d05ac122448f22bc4f90 web.
    archive.org/web/20180104003111im_/https:/climate.nasa.gov/system/
    time_series_images/944_co2_2002_9_0000_720x360.jpg
ee054aa33fa2b886a531b90362445787ee0ede6c0469d25a65608c1248b2e15d web.
    archive.org/_static/css/record.css
ba67f3fd5d5ead8288150257279315eb1fafc3518f2f6656521c773ce236c68f web.
    archive.org/web/20180104011219im_/https:/cdnjs.cloudflare.com/ajax/
    libs/slick-carousel/1.6.0/ajax-loader.gif
4f92252c8e879afb62d40d121af584f446b064a5e93f8dfb2c753396798376cc web.
    archive.org/web/20180104011219im_/https:/cdnjs.cloudflare.com/ajax/
    libs/slick-carousel/1.6.0/fonts/slick.eot
ca940d012efb06aa73d863579cb1815957b6f9ad4f350739aaddb8cd5164bb24 web.
    archive.org/web/20180104011219im_/https:/climate.nasa.gov/assets/
    magnifying_glass_black.png
<...235 more resources...>
-------------------------------------------------------------------------
Aggregated hash: 3673d8d59096b07ccc0093d1b5258fec5fe30142c6438f92eb7a2
    be2d83a1984
```

Fig. 9: An example of generating an aggregated hash on a composite memento using the shell script *aggregated_hash.sh* (Figure 8). The hash of each resource is marked in blue. File names are marked in green, and the aggregated hash value is marked red. The composite memento consists of 245 resources (only 10 are shown).

`http://perma-archives.org/warc/20170115024943/https://www.justice.gov/`

15 times in different days (the memento currently is no longer available in the archive, but a similar archived page is available at `https://perma.cc/4HWW-WXXU`). The visual representation of each replay is shown in Figure 12. We do not get the same representation each

(a) From the live web (The screen shot is captured on January 28, 2018 at 9:28 PM GMT).



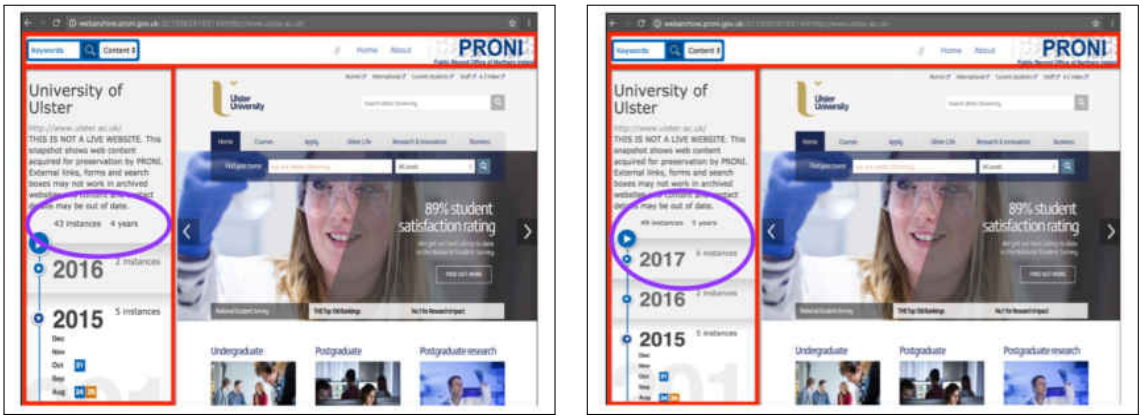
(b) In `archive.org`.

(c) In `perma.cc`.

(d) In `archive.is`.

Fig. 10: The live web page and its archived versions (three mementos). Each archive has applied a unique transformation to the original page.

time. We assume that the representation that does not have missing resources is considered the baseline (i.e., representation 1, 2, 3, 5, 7, 8, and 11). The other representations (marked in red) are incomplete for different reasons, such as having missing resources because of transient errors or being offline for server upgrades. For instance, representations 4, 6, and 14 do not include any image. Representation 15 has three missing images. Because of missing a CSS file that defines the design of the page (e.g., colors and layouts), representations 12 and 13 look totally different from the baseline. As explained in Chapter 6, there are other reasons causing incomplete representations of mementos.



(a) Replayed in 2016. (b) Replayed in 2017.

Fig. 11: The information in the archive's banner of the memento in 2016 (43 mementos were available in the archive for the original page `http://www.ulster.ac.uk`) is different from the information displayed in 2017 (49 mementos available and the year 2017 appears in the banner).

1.5.3 ARCHIVES OFTEN DO NOT SERVE A COMPOSITE MEMENTO PACKAGED IN A SINGLE FILE

Web archives usually do not provide a way for downloading a composite memento in a single file (i.e., bulk download). Instead, a composite memento is downloaded through multiple HTTP requests/responses. The more HTTP requests are needed, the longer it will take for the composite memento to download, and if the available computing resources in a server are not sufficient to handle the incoming request load, failed requests would occur (e.g., `timeout errors` or `5xx server errors`). This will affect generating repeatable fixity information. A few archives actually serve a composite memento packaged in a single file, such as the ZIP format by `archive.is` or the WARC format by `perma.cc`, but in both cases the files are generated dynamically upon request which makes the content of the files vary from time to time (Sections 2.2.6 and Chapter 6). For example, the ZIP file of the memento `http://archive.is/GrnkI` is available at `http://archive.is/download/GrnkI.zip`, and the WARC file for the memento `https://perma.cc/JX4V-WYAV` is available at `https://perma.cc/JX4V-WYAV?type=warc_download`. The WARC-related metadata `WARC-Date` in all WARC records always has the value of the current datetime.

At replay time, archives typically do not serve the content of a composite memento from a single WARC file. Multiple WARC files may be required to serve one composite memento.

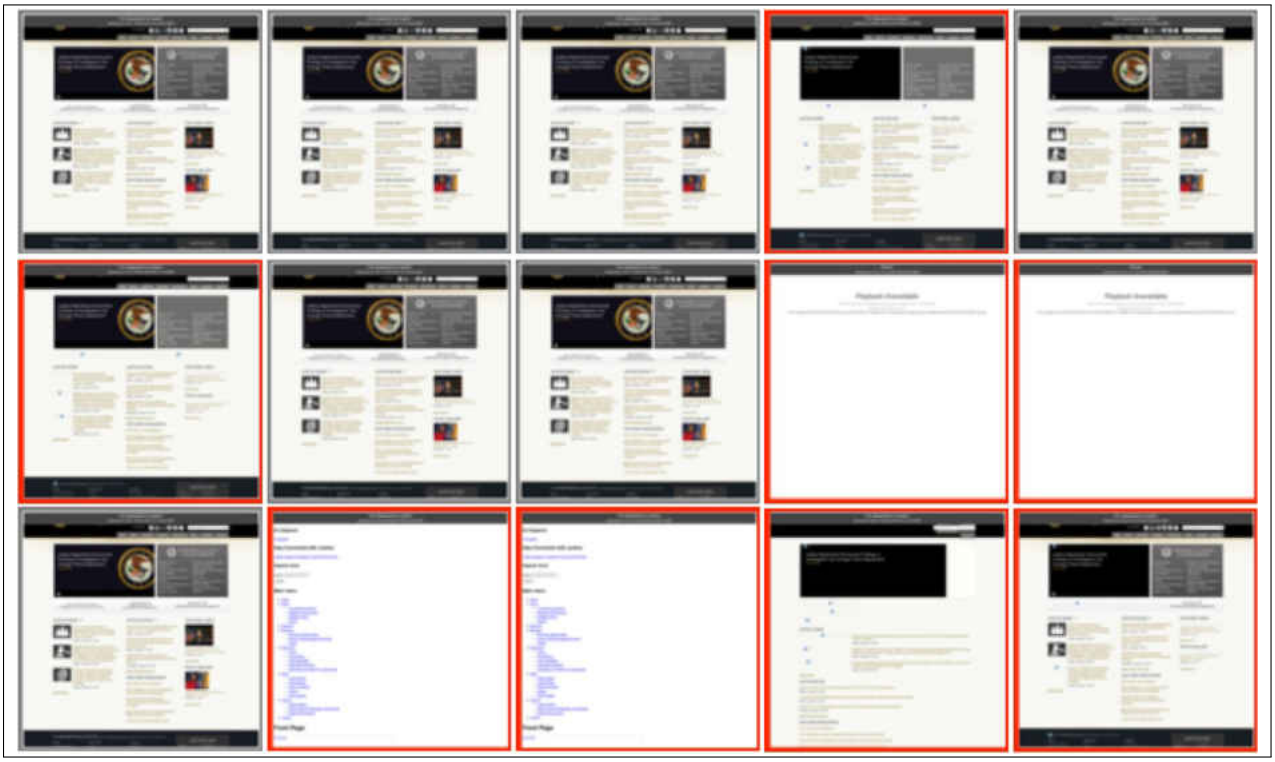


Fig. 12: A memento is replayed 15 times on different days. We do not get the same representation each time the memento is replayed. Representations 4, 6, 12-15 have missing resources. On 9 and 10, the archive was not functioning due to server upgrades.

For example, the composite memento:

`https://web.archive.org/web/20130530221910/http://www.cnn.com/`

consists of 128 resources including the base HTML file (there are 24 embedded resources added by the archive, which are not part of the original page). We download the memento using *Squidwarc* [53,54], which uses Headless Chrome [55], and store the resulting HTTP requests and responses in a WARC file. We name the resulting WARC file *www.cnn.com.warc*. We check the names of WARC files (via the HTTP response header `x-archive-src`) used by the archive to serve this composite memento. As Figure 13 shows, the archive uses 20 different WARC files to successfully deliver one composite memento.

The composite memento is aggregated from multiple WARC files at replay time. As described in Chapter 2.3.3, because of the techniques archives use for replaying mementos (i.e., indexes of the WARC files), a memento may become temporarily unavailable, the archive will locate the closest memento in time, which is likely to be in a different WARC file. This affects generating repeatable fixity information.

```
$ cat www.cnn.com.warc | grep "x-archive-src" | sort | uniq -c | sort -
    nr
  66 x-archive-src: ARCHIVEIT-823-DAILY-6320-00000/ARCHIVEIT-823-DAILY
     -6320-20130530143046793-00000-wbgrp-crawl055.us.archive.org-6443.
     warc.gz
  10 x-archive-src: edition.cnn.com-20130529-201414/IA-FOC-edition.cnn.
     com-20130531002415-00000.warc.gz
   9 x-archive-src: cnn.com-20130518-085929/IA-FOC-cnn.com
      -20130530185822-00000.warc.gz
   2 x-archive-src: live-20130622100220158-06734-20130622153024108/live
      -20130622153024108-06751.arc.gz
   2 x-archive-src: ARCHIVEIT-3611-NONE-16245-00000/ARCHIVEIT-3611-NONE
      -16245-20130405205913995-00000-wbgrp-crawl102.us.archive.org
      -6444.warc.gz
   1 x-archive-src: us.cnn.com-20130328-001336/IA-FOC-us.cnn.com
      -20130616192445-00000.warc.gz
   1 x-archive-src: liveweb-20170911161011/live-20170911160457-wwwb-
      app7.us.archive.org.warc.gz
   1 x-archive-src: live-20130709033519271-00942-20130709091502245/live
      -20130709091502245-00969.arc.gz
   1 x-archive-src: live-20130627111329865-07186-20130627183648217/live
      -20130627121528270-07192.arc.gz
   1 x-archive-src: live-20130619102716920-06458-20130619141235018/live
      -20130619130218274-06472.arc.gz
   1 x-archive-src: live-20130616155345785-06212-20130616181918863/live
      -20130616160613414-06214.arc.gz
   1 x-archive-src: live-20130612053900778-05874-20130612111534372/live
      -20130612094027220-05887.arc.gz
   1 x-archive-src: live-20130530192611915-04875-20130530221918640/live
      -20130530214811798-04889.arc.gz
   1 x-archive-src: live-20130530192611915-04875-20130530221918640/live
      -20130530193426196-04879.arc.gz
   1 x-archive-src: live-20130530185909274-04872-20130530225416173/live
      -20130530212951537-04888.arc.gz
   1 x-archive-src: live-20130530185909274-04872-20130530225416173/live
      -20130530210132581-04886.arc.gz
   1 x-archive-src: live-20130530185909274-04872-20130530225416173/live
      -20130530185909274-04872.arc.gz
   1 x-archive-src: edition.cnn.com-20130529-201414/IA-FOC-edition.cnn.
      com-20130626194633-00000.warc.gz
   1 x-archive-src: WIDE-20130528090214-crawl415/WIDE
      -20130528090711-03612.warc.gzG
   1 x-archive-src: ARCHIVEIT-1193-WEEKLY-13292-00000/ARCHIVEIT-1193-
      WEEKLY-13292-20130530202212164-00002-wbgrp-crawl060.us.archive.
      org-6441.warc.gz
```

Fig. 13: The archive serves one composite memento from 20 WARC files. The number of embedded resources served from each WARC file is marked in red.

1.6 RESEARCH QUESTIONS

After describing some difficulties of generating repeatable fixity information, we list our research questions.

RQ1: Can we identify and quantify the types of changes on the playback of mementos that prevent generating repeatable fixity information?

RQ2: Given the types of changes identified in the playback of mementos, what steps/guidelines should we consider in order to generate repeatable fixity information (defining an archive-aware fixity-based approach)?

RQ3: How can we store and retrieve fixity information independently from the web archives from which the associated mementos are preserved?

1.7 CHAPTER ORGANIZATION

The organization for the remainder of the proposal is described below.

Chapter 2: Background

We describe the web architecture including content negotiation and how live web pages are rendered. We give a brief introduction to web archiving, and how archives crawl the web and replay mementos. We briefly explain the WARC format, the common URI-M structures, Memento framework definitions, and raw mementos. Then, we give examples to show the effect of JavaScript of replayed mementos. We describe the Memento aggregators with examples of TimeGates and TimeMaps. We present a scenario where archives not able to deliver the right memento. We explain why verifying fixity can help to establish trusted web archives and describe some conventional approaches used to verify the fixity.

Chapter 3: Related Work

We describe some threats and attacks against web archives that can affect the representation of replayed mementos. We point to some tools for trusted timestamping using blockchain-based network. We explain trusty URIs and Multihash to show how URIs can contain hashes and be used to verify the fixity of web resources. We briefly explain how LOCKSS (Lots of Copies Keep Stuff Safe) works which shows the importance of replication.

Chapter 4: Defining an archive-aware hashing function

This chapter describes our archive-aware hashing function for generating fixity information on the playback of mementos. We start by introducing several initial guidelines that our archive-aware hashing function should follow for generating fixity information. Then, we introduce additional guidelines based on results from our 14-month study on 16,627 composite

mementos. This work addresses RQ1 and RQ2.

Chapter 5: Collecting a large dataset of mementos

This chapter describes four methods used to create a dataset of 16,627 URI-Ms from 17 public web archives. Even though we discover many mementos, we explain why we applied some filters and set some conditions to reduce the number of selected mementos. This dataset is used in our study (Chapter 6).

Chapter 6: Changes in the playback of mementos

This chapter explains our study toward identifying and quantifying the types of changes that cause different fixity information on composite mementos. The chapter describes in details changes caused by JavaScript, TimeMap inconsistency, transient errors, and others. The chapter also explains several scenarios of mementos moved from their original archives to other archives. This work addresses RQ1.

Chapter 7: Archive Assisted Archival Fixity Verification Framework

In this chapter, we introduce two approaches, *Atomic* and *Block*, that can be used to disseminate fixity information to independent web archives and verify the fixity of mementos. The chapter describes the evaluation of the two approaches using 16,627 mementos and four different public web archives. This work addresses RQ3.

Chapter 8: Contributions, Conclusions, and Future Work

This chapter summarizes how we have addressed the research questions. It also specifies the contributions, the future work, and the conclusions of our work.

CHAPTER 2

BACKGROUND

In this chapter, we explain some concepts related to the web architecture and web archiving. We describe how web pages are rendered and how archived web pages are replayed. We introduce some issues related to the replay of mementos including the effect of JavaScript on replayed mementos and why archives do not always give the requested memento. Finally, we explain how conventional cryptographic hashing algorithms can be used to verify the fixity of web resources.

2.1 WORLD WIDE WEB

The World Wide Web (WWW) [56], also known as the Web, was designed by Tim Berners-Lee in 1989. The web is a communication model where a client communicates with a server to request a resource. Each resource, which is the item of interest on the Web, is identified by a global identifier called a Uniform Resource Identifier (URI) [57]. The communication between clients and servers to establish connections and exchange messages is done through the Hypertext Transfer Protocol (HTTP) [58–64]. When you dereference a URI via HTTP (e.g., "click a link"), the server returns a representation of the resource. A single resource can have multiple representations. Figure 14 shows the relation between a resource, identifier, and representation on the Web. The URI identifies a resource without being associated with a particular representation of the resource.

In general, in response to an HTTP request, a server returns an HTTP response that should consist of the following:

1. HTTP status code: It indicates whether the HTTP request has been successfully handled by the server. For example, the status code `404 Not Found` indicates that the requested resource could not be found on the server while `200 OK` indicates that the server, successfully completed the request.

2. HTTP response entity headers: They contain information about the payload (e.g., `Content-Length` and `Content-Type`), the client/server connection (e.g., `Keep-Alive` and `Connection`), or the server (e.g., `Server`).

3. The HTTP response entity body: It is the response payload (i.e., the content of the requested resource). An HTTP response may not contain an entity body, for instance for responses with the HTTP status code `304`.

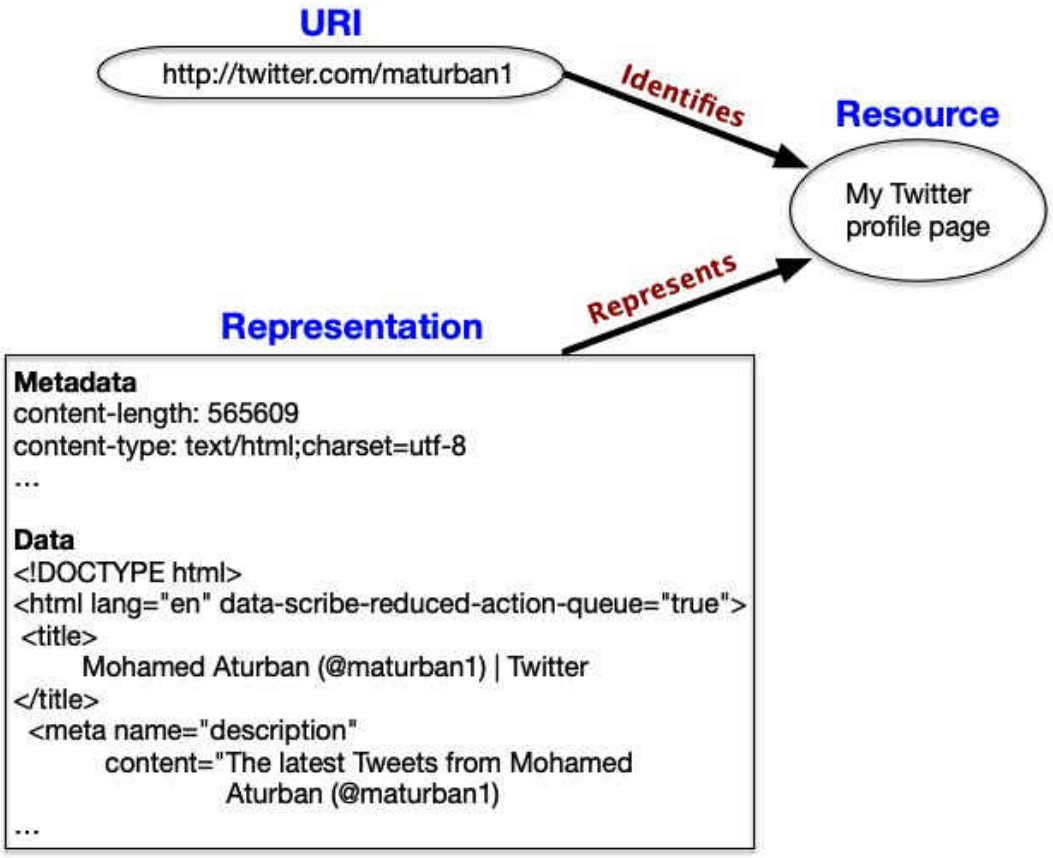


Fig. 14: The relation between a resource, URI, and representation on the Web.

A client sends requests to a server through a user-agent, as with the GUI web browser Google Chrome[1] or Mozilla Firefox[2]. A user-agent may also be a command-line based tool, such as `cURL` [65] or Wget [51].

2.1.1 URI REDIRECTION

When a server responds with the HTTP status code `30x` to an HTTP request of a resource, it indicates that the resource requested has been permanently moved (i.e., returning the `HTTP 301` status code) or temporarily moved (i.e., returning the `HTTP 302` status code) to the URI given by the HTTP response header `Location` [60]. Figure 15 shows an example

[1]https://www.google.com/chrome/
[2]https://www.mozilla.org/en-US/firefox/

where the client via the user-agent `cURL` sends HTTP request to `https://www.fb.com/`. The server responds with the HTTP status code `301` to `https://www.facebook.com/` as the header `Location` indicates. `cURL`'s options `--head` (or `-I`) is used to inform the server to return the HTTP response headers only without the HTTP entity body. The option `--location` (or `-L`) makes `cURL` follow redirects by recursively sending HTTP requests to the URI included in the HTTP response header `Location` if the returned HTTP status code is `30x`. As shown in Figure 15, the server returns `200 OK` as a response to the HTTP request to `https://www.facebook.com/`.

```
1  $ curl --head --Location https://www.fb.com/
2  HTTP/2 301
3  strict-transport-security: max-age=15552000
4  location: https://www.facebook.com/
5  access-control-expose-headers: X-FB-Debug, X-Loader-Length
6  access-control-allow-credentials: true
7  vary: Origin
8  access-control-allow-origin: https://www.fb.com
9  access-control-allow-methods: OPTIONS
10 content-type: text/html; charset="utf-8"
11 x-fb-debug: vnmL3ePSNLurmiRyp2znOHL2ttfuV0rovnuAwAANFTZDfU15kHGaT8tba+
       WKrYGTpJ9yBEXnHsZtdO89EyDT0Q==
12 content-length: 0
13 date: Mon, 29 Jul 2019 04:32:50 GMT
14
15 HTTP/2 200
16 set-cookie: fr=11x1LkeiVSeP9h1me..BdPnby.Sq.AAA.0.0.BdPnby.AWXvem4J;
       expires=Tue, 28-Jul-2020 04:32:49 GMT; Max-Age=31535999; path=/;
       domain=.facebook.com; secure; httponly
17 set-cookie: sb=8nY-XW3qKTcZl63wc0hJthL0; expires=Wed, 28-Jul-2021
       04:32:50 GMT; Max-Age=63072000; path=/; domain=.facebook.com; secure
       ; httponly
18 cache-control: private, no-cache, no-store, must-revalidate
19 pragma: no-cache
20 strict-transport-security: max-age=15552000; preload
21 vary: Accept-Encoding
22 x-content-type-options: nosniff
23 x-frame-options: DENY
24 x-xss-protection: 0
25 expires: Sat, 01 Jan 2000 00:00:00 GMT
26 content-type: text/html; charset="utf-8"
27 x-fb-debug: MSHG4LF4wQxdTmfqNAtF9xX054c6aFXodSskOAQXb20bDCkAZiRl25wn+
       SJDyLgN58LSatIIbffHwk0GUyviCw==
28 date: Mon, 29 Jul 2019 04:32:50 GMT
```

Fig. 15: The URI `https://www.fb.com/` redirects (i.e., via `301 Moved Permanently`) to `https://www.facebook.com/`.

2.1.2 CONTENT NEGOTIATION

Content negotiation [66] is a mechanism to serve multiple representations of a web resource. Content negotiation is used by the client (i.e., user-agent) to indicate which representation of a resource is preferred, such as content in a specific language or format. The server, on the other hand, uses content negotiation to convey some information, such as the format of the delivered content and its size in bytes. Information used for content negotiation is exchanged between the client and server through the HTTP request headers and HTTP response headers.

If the client does not provide any HTTP content negotiation preferences when requesting a resource, the server will respond with the default representation of the resource (e.g., representation in English). Figure 16 shows an example where `cURL` is used as a user-agent to request from the server `twitter.com` the representation of the resource `http://twitter.com/maturban1`. The server returns the default representation of the resource, and it is in English as line 12 indicates.

```
1  curl --include https://twitter.com/maturban1
2  HTTP/2 200
3  content-length: 564023
4  content-type: text/html;charset=utf-8
5  date: Wed, 24 Jul 2019 05:47:15 GMT
6  expires: Tue, 31 Mar 1981 05:00:00 GMT
7  last-modified: Wed, 24 Jul 2019 05:47:15 GMT
8  pragma: no-cache
9  server: tsa_b
10
11 <!DOCTYPE html>
12 <html lang="en" data-scribe-reduced-action-queue="true">
13   <head>
14       <title>Mohamed Aturban (@maturban1) | Twitter </title>
15       <meta name="robots" content="NOODP">
16   <meta name="description" content="The latest Tweets from Mohamed
       Aturban (@maturban1). Ph.D student in Computer Science (Old
       Dominion University). Member @WebSciDL. Norfolk, Virginia, USA">
17 ...
```

Fig. 16: The HTTP status code (marked in orange), and the HTTP response entity body (marked in blue) and headers (marked green) returned from the server in response to the request to `https://twitter.com/maturban1`. The representation (or in particular the entity body) is in English. The option `--include` in `cURL` is to include the HTTP response headers in the output.

Figure 17 shows how HTTP content negotiation can be used to request the same resource `http://twitter.com/maturban1` but with a different representation (e.g., a representation in Arabic language). The client sends to the server the HTTP request header (`Accept-Language: ar`) to specify the preferable language. The option `--header` (or `-H`) in cURL is used to include extra header in the HTTP request. The server returns the representation in Arabic as line 12 indicates (e.g., `lang="ar"`).

```
1 curl --include --header "Accept-Language: ar"  https://twitter.com/
    maturban1
2 HTTP/2 200
3 content-length: 575782
4 content-type: text/html;charset=utf-8
5 date: Wed, 24 Jul 2019 06:46:06 GMT
6 expires: Tue, 31 Mar 1981 05:00:00 GMT
7 last-modified: Wed, 24 Jul 2019 06:46:06 GMT
8 pragma: no-cache
9 server: tsa_b
10
11 <!DOCTYPE html>
12 <html lang="ar" data-scribe-reduced-action-queue="true">
13   <head>
14       <title>Mohamed Aturban (@maturban1) | تويتر </title>
15       <meta name="robots" content="NOODP">
16   <meta name="description" content=" أحدث التغريدات من Mohamed Aturban (
       @maturban1). Ph.D student in Computer Science (Old Dominion
       University). Member @WebSciDL. Norfolk, Virginia, USA">
17 ...
```

Fig. 17: Through content negotiation (e.g., `Accept-Language: ar`), the client requests the representation of the resource in Arabic.

2.1.3 RENDERING LIVE WEB PAGES

A web page typically consists of a base text file, written in HyperText Markup Language (HTML) [67], and other embedded files including Cascading StyleSheets (CSS) and JavaScript (JS) files, and images. HTML is the standard markup language that specifies how web pages are structured using a set of defined tags and tag attributes. Figure 18 shows the HTML (marked in blue) of a web page. Figure 19 illustrates the representation of the web page (i.e., the HTML) rendered in a web browser.

For a web page including its embedded resources to be fully rendered in a web browser, it goes through the following (simplified) steps:

```
curl -v https://maturban.github.io/playground/index.html
*   Trying 185.199.110.153...
> GET /playground/index.html HTTP/2
> Host: maturban.github.io
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/2 200
< server: GitHub.com
< content-type: text/html; charset=utf-8
< last-modified: Wed, 24 Jul 2019 22:28:50 GMT
< etag: "5d38dba2-163"
< access-control-allow-origin: *
< expires: Sun, 11 Aug 2019 03:07:28 GMT
< cache-control: max-age=600
< x-proxy-cache: MISS
< accept-ranges: bytes
< date: Sun, 11 Aug 2019 02:57:35 GMT
< via: 1.1 varnish
< age: 7
< x-served-by: cache-dca17765-DCA
< x-cache: HIT
< x-cache-hits: 1
< x-timer: S1565492255.111345,VS0,VE0
< vary: Accept-Encoding
< content-length: 355
<
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="styles.css">
    <title>Example Web Page</title>
</head>
<body>
    <p> An example web page with a link to <a href="https://ws-dl.cs.
       odu.edu/"> Old Dominion University</a></p>
    <p> The university logo is <img src="https://www.odu.edu/images/
       logo-university.png" >
</body>
```

Fig. 18: The HTML of the web page `https://maturban.github.io/playground/index.html`.

1. The user types the web page's URI into the address bar of the browser:

   `https://maturban.github.io/playground/index.html`

2. The browser sends an HTTP request to the server `maturban.github.io` requesting

Fig. 19: Rendering the HTML from Figure 18

the web page `/playground/index.html`.

3. The server handles the HTTP request and sends back an HTTP response to the browser. The returned HTTP entity body is the base HTML file `index.html` (Figure 18).

4. The browser parses the HTML and create a tree-based structure, called the Document Object Model (DOM) tree [68].

5. Upon creating the DOM, the browser usually issues additional HTTP requests to obtain the representations of resources embedded on the HTML page. The embedded resources are usually specified in HTML tags like `<img>` (for images), `<link>` (for CSS files), and `<script>` (for JavaScript files).

The number of resources embedded in a web page varies from zero to even hundreds. For example, the current `www.cnn.com` contains over 100 embedded resources. Web resources comprising a web page might be served from the same server hosting the base HTML file or any other server. The HTML page in Figure 18 contains only one image:

`https://www.odu.edu/images/logo-university.png`

and one style sheet, or CSS file,

`https://maturban.github.io/playground/styles.css`

which has a set of style rules to be applied to the HTML tags (to change the background color to light blue and the color of the text in paragraphs to blue). In sum, the browser requests the representations of three different resources (i.e., the base HTML file, CSS file,

and image) for the web page `https://maturban.github.io/playground/index.html` to be fully displayed.

2.2 WEB ARCHIVING

Web archiving is the process of preserving portions of the current web for future generations. Web archiving is not only concerned about collecting and preserving web pages but also about how to provide access to those archived resources. The Internet Archive (IA) is the world's largest public web archive. It holds hundreds of billions of archived web pages [69], and it tries to capture the entire web by employing large-scale web crawlers. The IA is not the only public archive on the web. Other archives were established with different objectives (e.g., focus on preserving special collections). For instance, the UK Web Archive was established with the objective of archiving only UK websites (e.g., `www.parliament.uk`) [70]. Other web archives, such as `perma.cc` and `archive.is`, capture web pages on demand, so they only preserve pages submitted by users, not through crawling the web. Other archives, such as `https://archive-it.org`, are subscription-based services where a subscriber can make a separate collection of URIs (i.e., seed URIs), and then the archive crawls these URIs based on options set by the subscriber.

2.2.1 WEB CRAWLERS

A web crawler is an automated program that is used by web search engines (e.g., Google and Bing) and web archives to systematically collect and discover web pages. The main purpose of crawling the web (e.g., via Google Googlebot [71]) by search engines is to index web pages to be able to respond to users with the most relevant web pages to their queries. Web archives use web crawlers, such as the Internet Archive's Heritrix [72], to collect and preserve original web pages, and allow access to those mementos. In general, an archive's web crawler follows the following (simplified) steps to crawl live web pages, or URI-Rs (a URI-R identifies an original resource from the live Web as defined in Chapter 2.2.3):

1. Insert a given set of URI-Rs (i.e., seed URI-Rs) in a queue.
2. Select (or dequeue) one URI-R from the queue.
3. Dereference the URI-R
4. Write the content of the dereferenced URI-R to a file. The common file format used by web archives is Web ARChive (WARC) [73] (as described in Chapter 2.2.2).

5. Extract any new URI-Rs that have not yet been crawled from the content.

6. Insert the newly discovered URI-Rs in the queue.

7. If the queue is not empty, go to step 2.

2.2.2 THE WEB ARCHIVE (WARC) ARCHIVE FORMAT

The Web ARChive (WARC) format [73] specifies a set of rules for aggregating multiple web resources (e.g., HTML files, images, and stylesheets) with the HTTP request/response entity and headers of each resource in addition to WARC-related metadata into a single file. A WARC file is made up of multiple records. Each record should have the metadata `WARC-Type` that indicates the type of the record. We describe four types of WARC records:

- `warcinfo` record: There is typically one `warcinfo` record in a WARC file, and this record appears at the beginning of the file. It contains metadata that describes the file, such as the WARC file's generator and datetime.

- `metadata` record: It describes content in other records. The metadata included in this record is not covered by other records. For example, the `metadata` record may include links found in HTML (i.e., outlinks).

- `request` record: It contains the full HTTP request sent to the server which should include the HTTP request headers (e.g., `Accept-Language`) and the request payload (if any).

- `response` record: It contains the full HTTP response received from the server. The record should contain the HTTP status code of the response, the HTTP response headers, and the HTTP entity body (if any). The entity body can be a text-based file (e.g., HTML or CSS file), or binary file (e.g., images).

Figures 20 and 21 show examples of four records. These records are part of a WARC file created by the browser extension WARCreate [74] for the web page `https://maturban.github.io/playground/index.html`.

2.2.3 MEMENTO FRAMEWORK DEFINITIONS

```
WARC/1.0
WARC-Type: warcinfo
WARC-Date: 2019-07-25T06:43:46Z
WARC-Filename: 20190725064346105.warc
WARC-Record-ID: <urn:uuid:57e1f2f2-62f8-df6f-ba86-2a722a87064e>
Content-Type: application/warc-fields
Content-Length: 464

software: WARCreate/0.2019.1.19 http://warcreate.com
format: WARC File Format 1.0
conformsTo: http://bibnum.bnf.fr/WARC/
    WARC_ISO_28500_version1_latestdraft.pdf
description:  Crawl initiated from the WARCreate Google Chrome
    extension
http-header-from: warcreate@matkelly.com

WARC/1.0
WARC-Type: metadata
WARC-Target-URI: https://maturban.github.io/playground/index.html
WARC-Date: 2019-07-25T06:43:46Z
WARC-Record-ID: <urn:uuid:6fef2a49-a9ba-4b40-9f4a-5ca5db1fd5c6>
Content-Type: application/warc-fields
Content-Length: 189

outlink: https://www.odu.edu/images/logo-university.png E =EMBED_MISC
outlink: https://maturban.github.io/playground/styles.css E link/@href
outlink: https://ws-dl.cs.odu.edu/ L a/@href
```

Fig. 20: A part of a WARC file shows two records (`warcinfo` and `metadata`). These records contain WARC-related metadata. The WARC file created by `WarcCreate` as a resulting of requesting `https://maturban.github.io/playground/index.html`.

Memento is an HTTP protocol extension that uses time as a dimension to access the web by relating current web resources to their prior states [13,75]. The Memento protocol is supported by many 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. Figure 22 shows an example of an HTTP request with the header `Accept-Datetime` and the subsequent response with the `Memento-Datetime` in the response headers. The Memento protocol also defines the following terminology:

- URI-R - an original resource from the live Web

```
1  WARC/1.0
2  WARC-Type: request
3  WARC-Target-URI: https://maturban.github.io/playground/index.html
4  WARC-Date: 2019-07-25T06:43:46Z
5  WARC-Record-ID: <urn:uuid:edb961e3-e05f-4e86-21eb-ff35a9acb1d8>
6  Content-Type: application/http; msgtype=request
7  Content-Length: 332
8
9  GET /playground/index.html HTTP/1.1
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit
       /537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
       webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
13
14 WARC/1.0
15 WARC-Type: response
16 WARC-Target-URI: https://maturban.github.io/playground/index.html
17 WARC-Date: 2019-07-25T06:43:46Z
18 WARC-Record-ID: <urn:uuid:8de6e979-f774-ee37-de60-7ecbbfde911c>
19 Content-Type: application/http; msgtype=response
20 Content-Length: 985
21
22 HTTP/1.1 200
23 content-type: text/html; charset=utf-8
24 date: Thu, 25 Jul 2019 06:43:40 GMT
25 ...
26 Content-Length: 356
27
28 <!DOCTYPE html><html lang="en"><head>
29     <link rel="stylesheet" href="styles.css">
30     <title>Example Web Page</title>
31 </head>
32 <body>
33     <p> An example web page with a link to <a href="https://ws-dl.cs.
           odu.edu/"> Old Dominion University</a></p>
34     <p> The university logo is <img src="https://www.odu.edu/images/
           logo-university.png">
35
36 </p></body></html>
```

Fig. 21: A part of a WARC file shows the two records (`request` and `response`). It contains the HTTP headers and HTTP entity bodies resulting from requesting `https://maturban.github.io/playground/index.html`.

- 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

- URI-G - a resource (TimeGate) that supports content negotiation based on datetime to access prior versions of an original resource

```
curl -vL -H "Accept-Datetime: Thu, 25 Jul 2019 21:29:38 GMT" http://web
    .archive.org/web/https://maturban.github.io/playground/index.html

> GET /web/https://maturban.github.io/playground/index.html HTTP/1.1
> Host: web.archive.org
> User-Agent: curl/7.54.0
> Accept: */*
> Accept-Datetime: Thu, 25 Jul 2019 21:29:38 GMT
>
< HTTP/1.1 302 FOUND
...
< Location: http://web.archive.org/web/20190725212938/https://maturban.
    github.io/playground/index.html
<
> GET /web/20190725212938/https://maturban.github.io/playground/index.
    html HTTP/1.1
> Host: web.archive.org
> User-Agent: curl/7.54.0
> Accept: */*
> Accept-Datetime: Thu, 25 Jul 2019 21:29:38 GMT
>
< HTTP/1.1 200 OK
< Memento-Datetime: Thu, 25 Jul 2019 21:29:38 GMT
...
<
<!DOCTYPE html>
<html lang="en">
...
```

Fig. 22: Time-based content negotiation using the HTTP request header `Accept-Datetime`.

2.2.4 TWO COMMON URI-M STRUCTURES

The common URI-M structure used by web archives to identify mementos is illustrated in Figure 23, especially for web archives [76, 77] that use one of the Wayback Machine's implementations, such as OpenWayback [78] or PyWb [7]. The examples below are URI-Ms from different web archives:

- `https://swap.stanford.edu/20120105233133/http://www.epa.gov/`

- https://web.archive.org/web/20161201155342/https://www.ap.org/en-us/
- http://wayback.archive-it.org/all/19961231235847/http://www.nasa.gov/
- http://wayback.vefsafn.is/wayback/20090422094000/http://www.columbia.edu/
- https://www.webharvest.gov/congress111th/20101202093134/http://www.ustr.gov/

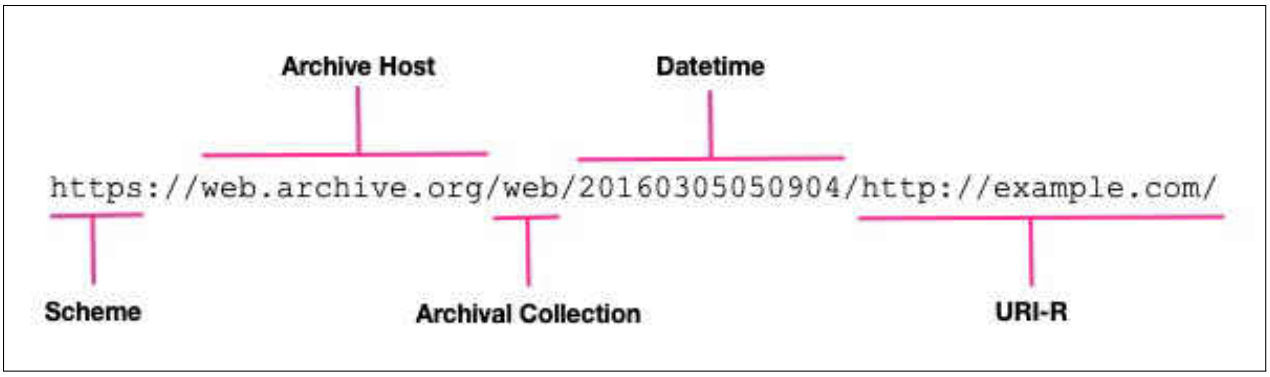


Fig. 23: The common URI-M structure used by Wayback Machine web archives.

The other URI-M structure is based on including in the URI-M a string of characters (e.g., a hash value) that is derived from the content. Archives such as webcitation.org, archive.is, and perma.cc use URI-Ms that include hash values as shown in the examples below:

- https://archive.is/Ceuov
- https://perma.cc/JX4V-WYAV
- https://www.webcitation.org/5BmjfFFB1

Several web archives support both URI-M structures, such as perma.cc[3] and archive.is. For example, the URI-M

https://archive.is/Ceuov

and the URI-M

https://archive.is/20190609163259/http://example.com/

[3]Based on changes made on February 03, 2020 (github.com/harvard-lil/perma/pull/2699), perma.cc no longer supports the long form URI-Ms.

are URI aliases for the same memento. The URI-M with hashes are short and can support deduplication, and because these URI-Ms are derived from the content (e.g., using cryptographic hash functions), they can be used to verify the fixity of the content.

The other URI-M structure has some advantages and features that are not available with the URI-Ms with hashes. We can extract some information from only the URI-M, such as the `Memento-Datetime` of a memento and its original resource (URI-R). In general, we consider the datetime string (e.g., `20190609163259` in the URI-M above) as the value of the `Memento-Datetime` header because archives usually use this datetime string to refer to the creation datetime of a memento. However, we may need to dereference a URI-M to get the actual `Memento-Datetime` because the URI-M may redirect to a different URI-M, which contains a different datetime string. The archives can use the information extracted from a URI-M to return alternative or closest mementos (e.g., using a Memento aggregator) if the requested memento is not available.

2.2.5 REPLAYING MEMENTOS

The crawling process will result in a set of archived pages stored in WARC files or in other formats. These files are indexed, which will create other files, such as CDX [79] and CDXJ [80, 81] to map a particular URI-R to a resource in a WARC file. Currently, the WARC format is not directly supported by web browsers. Thus, in order to replay the content of WARC files, many web archives use OpenWayback [78] or PyWb [7]. The replay process includes indexing WARC files, which helps in looking up archived pages by their URI-Rs. On the replay of an archived page, one of the main tasks of OpenWayback is to ensure that all resources comprising the page (e.g., images, style sheets, and JavaScript files) are retrieved from the archive, not from the live web. Thus, at the time of replaying the page, OpenWayback rewrites all links to those embedded resources to point to the archive [44].

To illustrate how web archives transform content of original web pages to appropriately replay them in a user's browser, we submitted the web page's URI-R

`https://maturban.github.io/playground/index.html`

to the Internet Archive using the "Save Page Now" feature [82]. The archive usually captures the root HTML file and all embedded resources included in the page. Table 1 shows the URI-Rs of the original resources comprising the web page and the corresponding URI-Ms to the mementos created by the Internet Archive (the `Memento-Datetime` of each memento is marked in red).

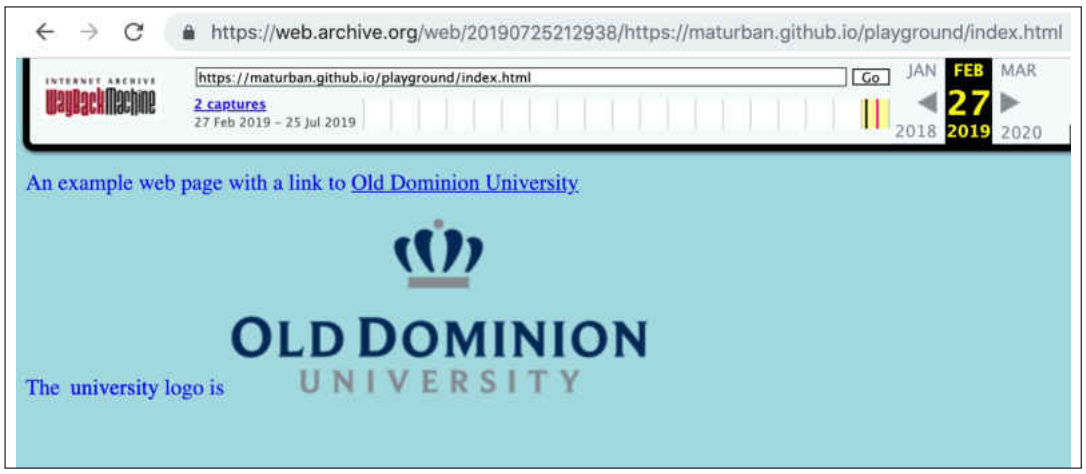


Fig. 24: The representation of the memento `web.archive.org/web/20190725212938/https://maturban.github.io/playground/index.html` (cf. the live web version in Figure 19).

TABLE 1: The mementos of the original resources comprising the web page in Figure 18. The original resource (URI-R) is shown in blue text and the Memento-Datetime is shown in red text.

Original resource (URI-R)	Memento (URI-M)
https://maturban.github.io/playground/index.html	https://web.archive.org/web/20190725212938/https://maturban.github.io/playground/index.html
https://maturban.github.io/playground/styles.css	https://web.archive.org/web/20190725212938/https://maturban.github.io/playground/styles.css
https://www.odu.edu/images/logo-university.png	https://web.archive.org/web/20190725212938/https://www.odu.edu/images/logo-university.png

Figure 24 illustrates the representation of the memento replayed in the browser. The archive transformation process of the original page may include adding multiple HTML tags and the archive's banner. As illustrated in Figure 25, the archive-specific code is marked in red and we used `cURL` to request the memento. The archive's banner provides information about both the memento being viewed and the original page (e.g., the top portion shown in Figure 24).

Archives can inform clients about resources that are not mementos (e.g., resources added by the archive) by returning the value:

```
<http://mementoweb.org/terms/donotnegotiate>; rel="type"
```

in the `Link` HTTP response header. Thus, the clients are aware that time-based content

negotiation cannot be attempted on these resources [13, 83].

```
curl https://web.archive.org/web/20190725212938/https://maturban.github
    .io/playground/index.html
<!DOCTYPE html>
<html lang="en">
<script src="//archive.org/includes/analytics.js?v=cf34f82" type="text/
    javascript"></script>
...
<link rel="stylesheet" type="text/css" href="/_static/css/banner-styles
    .css" />
<link rel="stylesheet" type="text/css" href="/_static/css/iconochive.
    css" />
<!-- End Wayback Rewrite JS Include -->

    <link rel="stylesheet" href="/web/20190725212938cs_/https://
        maturban.github.io/playground/styles.css">
    <title>Example Web Page</title>
</head>
<body>
    <p> An example web page with a link to <a href="https://web.archive
        .org/web/20190725212938/https://ws-dl.cs.odu.edu/"> Old Dominion
         University</a></p>
    <p> The University logo is <img src="https://web.archive.org/web
        /20190725212938im_/https://www.odu.edu/images/logo-university.
        png">
</body>
</html>

<!--
     FILE ARCHIVED ON 21:29:38 Jul 25, 2019 AND RETRIEVED FROM THE
     INTERNET ARCHIVE ON 21:51:15 Jul 25, 2019.
     JAVASCRIPT APPENDED BY WAYBACK MACHINE, COPYRIGHT INTERNET ARCHIVE
        .
     ...
-->
<!--
playback timings (ms):
  LoadShardBlock: 75.831 (3)
  esindex: 0.006
  captures_list: 97.72
  ...
-->
```

Fig. 25: The rewritten HTML of the memento `web.archive.org/web/20190725212938/https://maturban.github.io/playground/index.html`. The code, marked in red, was added by the archive (i.e., archive-specific content).

As described in Chapter 1.5.1, web archives may transform original web pages differently depending on the replay tools they are using. For example, `archive.is` transforms the

original pages by removing all JavaScript code, so no JavaScript code is executed at replay time. Berlin [84] explains different approaches used by web archives to transform and reply archived web pages.

2.2.6 THE EFFECT OF JAVASCRIPT ON REPLAYED MEMENTOS

Early web pages were relatively static [85]. However, web pages continuously adopt new web technologies and become more interactive. JavaScript, which runs on the client, adds interactivity to web pages and can manipulate and update the representation of web pages without reloading them in the browser. About 54.5% of web pages in 2012 included JavaScript to load embedded resources [86], and by June 2020, the median number of JavaScript files requests per page is 21 [87]. Withee [88] found that when JavaScript is disabled, about 45% of web pages are displayed as a white screen, and most content of about 50% of web pages does not load. When a browser executes JavaScript included in a web page, it might result in adding new resources to the page. Figure 26(a) shows an example where the memento

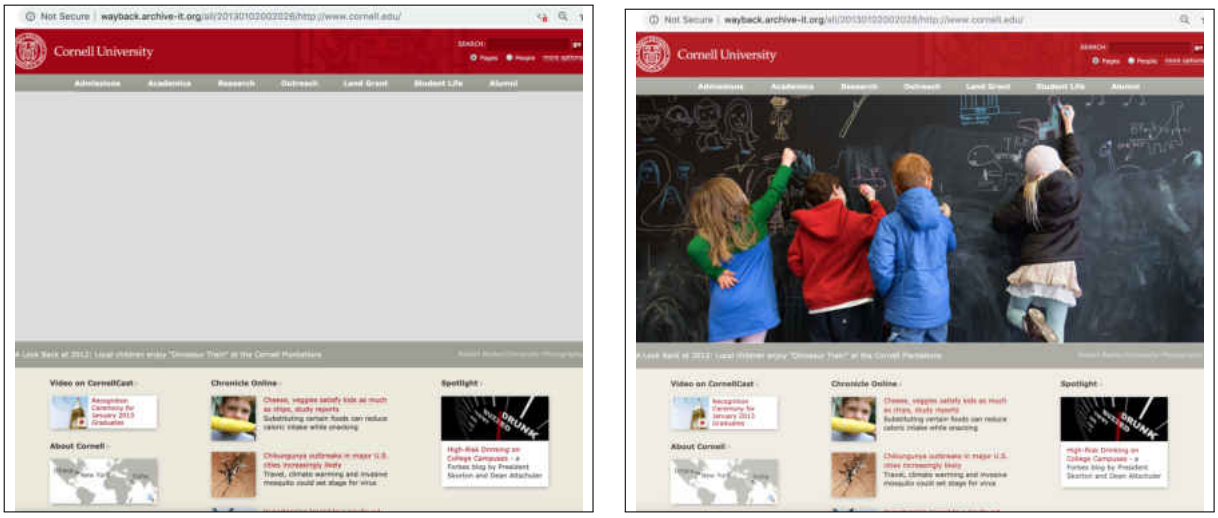
`http://wayback.archive-it.org/all/20130102002028/http://www.cornell.edu/`

is reloaded in the browser with JavaScript disabled (i.e., the browser does not execute JavaScript code), which affects the representation as the background image is not displayed. When we reload the same memento with JavaScript enabled, the background image appears on the page (Figure 26(b)).

URI-Ms for embedded resources can be generated randomly by JavaScript. This impacts the generation of repeatable fixity information. Figure 27 shows an example where each time the same memento from Figure 26 is reloaded in the browser, it results in a different background image. The reason for observing different resources at replay time for this memento is that values in links to the background images are generated randomly on the client side by JavaScript.

The other problem caused by JavaScript is that when JavaScript is executed, it dynamically generates URIs. These URIs may not be correctly rewritten by the web archive (Chapter 4.1.9), so resources specified by such URIs will be retrieved from the live web, not from the archive [84].

2.2.7 RAW MEMENTOS



(a) JavaScript disabled. (b) JavaScript enabled.

Fig. 26: An example shown an additional resource (the background image) loaded by JavaScript.

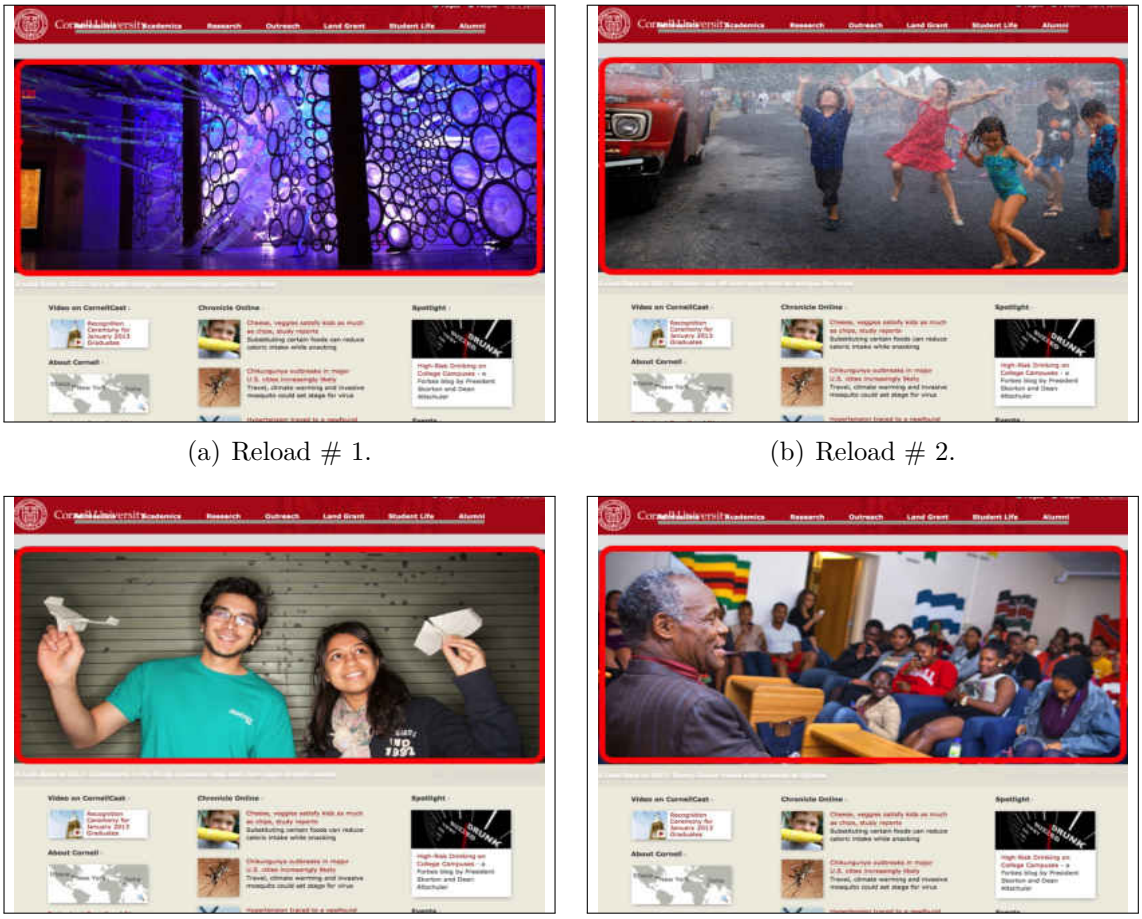
In addition to the rewritten content, many archives allow accessing unaltered, or raw, archived content (i.e., retrieving the archived original content without any type of transformation by the archive). The most common mechanism, used by many Wayback Machine implementations, to retrieve the raw mementos is by adding `id_` [89,90] after the timestamp in the requested URI-M as Figure 28 shows. As expected, Figure 29 indicates that the content of the raw memento (e.g., Figure 28)

`https://web.archive.org/web/20190725212938id_/https://maturban.github.io/playground/index.html`

is the same as the content of its original page (Figure 18)

`https://maturban.github.io/playground/index.html`

as the resulting hash values computed on both resources are identical. This reinforces our intuition that we are always able to generate repeatable hashes on raw mementos, but this is not always the case for two reasons. First, we are interested in computing hashes on composite raw memento, not only the base HTML file. Second, some archives respond to raw memento requests with an altered (or a custom) HTML base file (difficulties in generating repeatable hashes are explained in Chapter 4 and Chapter 6). Other archives allow accessing the raw content through different APIs. For instance, the archive `internetmemory.org`



(a) Reload # 1. (b) Reload # 2.

(c) Reload # 3. (d) Reload # 4.

Fig. 27: Because of JavaScript, reloading the memento `http://wayback.archive-it.org/all/20130102002028/http://www.cornell.edu/` multiple times result in different background images.

responds with the raw content if `raw/` is placed before the timestamp in a URI-M. Jones et al. [91–93] explore the transformation of original content performed by different web archives and propose several rules for acquiring mementos. The proposals have not been adopted, and raw mementos access remains unstandardized and ad-hoc.

2.3 MEMENTO FRAMEWORK MECHANICS

In this section, we show multiple TimeGate and TimeMap examples and explain a Memento aggregator. We also introduce an abstract example that illustrates why web archives do not always give the requested mementos.

```
curl https://web.archive.org/web/20190725212938id_/https://maturban.
    github.io/playground/index.html
<html>
<head>
    <link rel="stylesheet" href="styles.css">
    <title>Example Web Page</title>
</head>
<body>
    <p> An example web page with a link to <a href="https://ws-dl.cs.
        odu.edu/"> Old Dominion University</a></p>
    <p> The university logo is <img src="https://www.odu.edu/images/
        logo-university.png" >
</body>
</html>
```

Fig. 28: The raw HTML from requesting the memento `https://web.archive.org/web/20190725212938id_/https://maturban.github.io/playground/index.html`. The code is the same as the code illustrated in Figure 18.

```
$ curl -s https://maturban.github.io/playground/index.html | sha256sum
014da5ab8ef445f32c414690c53e9abe88ba353abf72a132c06d71217450b1a8  -

$ curl -s https://web.archive.org/web/20190725212938id_/https://
    maturban.github.io/playground/index.html | sha256sum
014da5ab8ef445f32c414690c53e9abe88ba353abf72a132c06d71217450b1a8  -
```

Fig. 29: The hash value of the live web page (Figure 18) is identical to the hash of one of its raw mementos (Figure 28).

2.3.1 TIMEGATE AND TIMEMAP EXAMPLES

Figure 30 shows an example of content negotiation based on time. The client sends an HTTP request to a TimeGate at the URI-G

`http://web.archive.org/web/https://climate.nasa.gov/vital-signs/carbon-dioxide/`

requesting a memento (URI-M) captured on April 24, 2016 at midnight GMT for the original resource (URI-R)

`https://climate.nasa.gov/vital-signs/carbon-dioxide/`.

The datetime is sent to the server via the HTTP header `Accept-Datetime`. Even though the server (or the TimeGate) cannot find a memento captured at the given time, it responds with the HTTP status code `302` to the URI-M closest to the given `Accept-Datetime`. The URI-M is included in the HTTP response header `Location`, and as the URI-M indicates, the memento is captured on April 26, 2016 at 23:24:25. Figure 31 shows that the client receives `200 OK` after sending an the HTTP request to the URI-M

```
http://web.archive.org/web/20160426232425/http://climate.nasa.gov/vital
-signs/carbon-dioxide
```

The value of the HTTP response header `Memento-Datetime` indicates the datetime at which the memento was captured by the archive. The response header `Link` is a standard HTTP header [94, 95]. Links included in the `Link` header are separated by commas. This `Link` header is used by the Memento framework to includes links to: an original resource (URI-R), a TimeMap (URI-T) of the original resource, and a TimeGate (URI-G). It also contains links to the next memento, the previous memento, the first memento made for the original resource, and the last memento. The HTTP response headers that start with `X-Archive-Orig-` are the original headers returned by the server from which the original page is captured. Some web archives prepend the string `X-Archive-Orig-` to the original HTTP response headers so that the client can differentiate between the original response headers and the response headers that are added by the archive (e.g., `Memento-Datetime`).

Some archives develop APIs that consider the timestamp include in a requested URI-M as the value of the HTTP request header `Accept-Datetime`. Thus, even though a client does not include the `Accept-Datetime` header in an HTTP request, the archive will respond with the URI-M closest to the timestamp included in the requested URI-M. For example, the two HTTP requests below will result in a `302` redirect to the same URI-M:

```
curl -I -H "Accept-Datetime:  Sun, 24 Apr 2016 00:00:00 GMT" http://web.
archive.org/web/https://climate.nasa.gov/vital-signs/carbon-dioxide/
```

and

```
curl -I http://web.archive.org/web/20160424000000/https://climate.nasa.g
ov/vital-signs/carbon-dioxide/
```

Both requests will redirect to the same URI-M

```
http://web.archive.org/web/20160426232425/https://climate.nasa.gov/vital-s
igns/carbon-dioxide/
```

```
curl -v -H "Accept-Datetime: Sun, 24 Apr 2016 00:00:00 GMT" http://web.
    archive.org/web/https://climate.nasa.gov/vital-signs/carbon-dioxide/
*   Trying 207.241.233.214...
* Connected to web.archive.org (207.241.233.214) port 80 (#0)
> GET /web/https://climate.nasa.gov/vital-signs/carbon-dioxide/ HTTP
    /1.1
> Host: web.archive.org
> User-Agent: curl/7.54.0
> Accept: */*
> Accept-Datetime: Sun, 24 Apr 2016 00:00:00 GMT
>
< HTTP/1.1 302 FOUND
< Date: Mon, 29 Jul 2019 05:55:01 GMT
< Content-Type: text/plain; charset=utf-8
< X-Archive-Redirect-Reason: found capture at 20160426232425
< Location: http://web.archive.org/web/20160426232425/http://climate.
    nasa.gov/vital-signs/carbon-dioxide
< Link: <https://climate.nasa.gov/vital-signs/carbon-dioxide/>; rel="
    original", <http://web.archive.org/web/20160426232425/http://climate
    .nasa.gov/vital-signs/carbon-dioxide>; rel="memento"; datetime="Tue,
     26 Apr 2016 23:24:25 GMT", <http://web.archive.org/web/timemap/link
    /https://climate.nasa.gov/vital-signs/carbon-dioxide/>; rel="timemap
    "; type="application/link-format"
<
```

Fig. 30: Memento framework allows content negotiations based on time. A client sends HTTP request with the header `Accept-Datetime` to a TimeGate requesting a memento. The TimeGate returns a URI-M that is closest to the value of `Accept-Datetime`.

Figure 32 shows the TimeMap of the original resource (URI-R)

`http://climate.nasa.gov/vital-signs/carbon-dioxide`.

The TimeMap contains the set of all mementos (URI-Ms) available in the archive for this URI-R. There are 4,706 mementos captured between October 2014 and July 2019. Only 10 mementos are shown, two per year. In addition to the HTTP entity body, the `--include` option (or `-i`) will include the HTTP response headers in the output.

2.3.2 MEMENTO AGGREGATORS

A Memento aggregator can be used to retrieve TimeMaps aggregated from multiple web archives. The Memento aggregator from Los Alamos National Laboratory (LANL) [96] is one implementation of a Memento aggregator that provides TimeMaps aggregated from different web archives both with (a) native support of the Memento protocol and (b) by

```
$ curl -I http://web.archive.org/web/20160426232425/http://climate.nasa
    .gov/vital-signs/carbon-dioxide
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Mon, 29 Jul 2019 06:42:48 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 125894
Connection: keep-alive
X-Archive-Orig-status: 200 OK
X-Archive-Orig-content-length: 21162
X-Archive-Orig-server: nginx/1.4.6 (Ubuntu)
X-Archive-Orig-etag: "f7bf2ceecf8fed0fa794552186cfd772-gzip"
X-Archive-Orig-date: Tue, 26 Apr 2016 23:24:25 GMT
X-Archive-Guessed-Content-Type: text/html
X-Archive-Guessed-Charset: utf-8
Memento-Datetime: Tue, 26 Apr 2016 23:24:25 GMT
Link: <http://climate.nasa.gov:80/vital-signs/carbon-dioxide/>; rel="
    original", <http://web.archive.org/web/timemap/link/http://climate.
    nasa.gov:80/vital-signs/carbon-dioxide/>; rel="timemap"; type="
    application/link-format", <http://web.archive.org/web/http://climate
    .nasa.gov:80/vital-signs/carbon-dioxide/>; rel="timegate", <http://
    web.archive.org/web/20141010072816/http://climate.nasa.gov:80/vital-
    signs/carbon-dioxide>; rel="first memento"; datetime="Fri, 10 Oct
    2014 07:28:16 GMT", <http://web.archive.org/web/20160419081121/http
    ://climate.nasa.gov:80/vital-signs/carbon-dioxide/>; rel="prev
    memento"; datetime="Tue, 19 Apr 2016 08:11:21 GMT", <http://web.
    archive.org/web/20160426232425/http://climate.nasa.gov:80/vital-
    signs/carbon-dioxide/>; rel="memento"; datetime="Tue, 26 Apr 2016
    23:24:25 GMT", <http://web.archive.org/web/20160429203645/http://
    climate.nasa.gov/vital-signs/carbon-dioxide/>; rel="next memento";
    datetime="Fri, 29 Apr 2016 20:36:45 GMT", <http://web.archive.org/
    web/20190728173445/https://climate.nasa.gov/vital-signs/carbon-
    dioxide/>; rel="last memento"; datetime="Sun, 28 Jul 2019 17:34:45
    GMT"
Content-Security-Policy: default-src 'self' 'unsafe-eval' 'unsafe-
    inline' data: blob: archive.org web.archive.org analytics.archive.
    org pragma.archivelab.org
```

Fig. 31: The HTTP request to the URI-M returns `200 OK`. The HTTP response header `Memento-Datetime` is included in the response. As defined by the Memento framework, this header indicates the datetime at which the memento was created. The `link` header contains links to the original resource (URI-R marked in blue), the TimeMap (URI-T) of the original resource (green), the TimeGate (URI-G in orange), and the links remaining are for the first memento, the previous memento, the next memento, and the last memento.

proxy support of the Memento protocol. MemGator [97, 98] is another implementation of a Memento aggregator and an open source project that provides a variety of customization options, such as allowing users to specify a list of web archives to retrieve TimeMaps from,

```
$ curl -include http://web.archive.org/web/timemap/link/https://climate
    .nasa.gov/vital-signs/carbon-dioxide/
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Mon, 29 Jul 2019 21:46:55 GMT
Content-Type: application/link-format

<http://climate.nasa.gov:80/vital-signs/carbon-dioxide>; rel="original
    ",
<http://web.archive.org/web/timemap/link/https://climate.nasa.gov/vital
    -signs/carbon-dioxide/>; rel="self"; type="application/link-format";
     from="Fri, 10 Oct 2014 07:28:16 GMT",
<http://web.archive.org>; rel="timegate",
<http://web.archive.org/web/20141010072816/http://climate.nasa.gov:80/
    vital-signs/carbon-dioxide>; rel="first memento"; datetime="Fri, 10
    Oct 2014 07:28:16 GMT",
<http://web.archive.org/web/20150811233214/http://climate.nasa.gov:80/
    vital-signs/carbon-dioxide>; rel="memento"; datetime="Tue, 11 Aug
    2015 23:32:14 GMT",
<http://web.archive.org/web/20160304214529/http://climate.nasa.gov:80/
    vital-signs/carbon-dioxide/>; rel="memento"; datetime="Fri, 04 Mar
    2016 21:45:29 GMT",
<http://web.archive.org/web/20161111001959/http://climate.nasa.gov:80/
    vital-signs/carbon-dioxide>; rel="memento"; datetime="Fri, 11 Nov
    2016 00:19:59 GMT",
<http://web.archive.org/web/20171218165218/https://climate.nasa.gov/
    vital-signs/carbon-dioxide/>; rel="memento"; datetime="Mon, 18 Dec
    2017 16:52:18 GMT",
<http://web.archive.org/web/20170526004412/https://climate.nasa.gov/
    vital-signs/carbon-dioxide>; rel="memento"; datetime="Fri, 26 May
    2017 00:44:12 GMT",
<http://web.archive.org/web/20180612130914/https://climate.nasa.gov/
    vital-signs/carbon-dioxide/>; rel="memento"; datetime="Tue, 12 Jun
    2018 13:09:14 GMT",
<http://web.archive.org/web/20181214074023/https://climate.nasa.gov/
    vital-signs/carbon-dioxide/>; rel="memento"; datetime="Fri, 14 Dec
    2018 07:40:23 GMT",
<http://web.archive.org/web/20190223180808/https://climate.nasa.gov/
    vital-signs/carbon-dioxide/>; rel="memento"; datetime="Sat, 23 Feb
    2019 18:08:08 GMT",
<http://web.archive.org/web/20190723114611/https://climate.nasa.gov/
    vital-signs/carbon-dioxide/>; rel="memento"; datetime="Tue, 23 Jul
    2019 11:46:11 GMT",
...
```

Fig. 32: The TimeMap of the URI-R `http://climate.nasa.gov/vital-signs/carbon-dioxide`. The TimeMap contains 4,706 mementos (only 10 are shown)

but it only aggregates TimeMaps from archives that natively support the Memento protocol. The UK Web Archive also runs a Memento aggregator [99] that allows users to submit a URI-R and return a UI page with discovered mementos from multiple archives.

Figure 33 shows the TimeMap of the original resource

`http://climate.nasa.gov/vital-signs/carbon-dioxide`.

The TimeMap is aggregated from six different public web archives (marked in red). The total number of aggregated mementos in the TimeMap is 4,816, though only eight are shown. Most of the mementos (4706) are from the Internet Archive. The second archive that contributes the second-most to this TimeMap is `archive-it.org` (91 mementos). The 19 mementos remaining are from `archive.is` (13), `arquivo.pt` (3), and `perma-archives.org` (3). AlSum et al. study the impact on aggregated TimeMaps so that requests of TimeMaps are only sent to web archives that likely preserve the requested archived pages [100, 101].

2.3.3 ARCHIVES DO NOT ALWAYS GIVE THE CORRECT COMPOSITE MEMENTOS

As defined in Chapter 1.4, an archived HTML page is a composite memento. It usually consists of the base HTML file and other embedded resources, such as CSS, JavaScript files and images. The mechanism that some web archives use to crawl the web (Chapter 2.2.1) results in a composite memento where the resources comprising the memento have different `Memento-Datetime`s (cf. web pages that consist of one resource like PDFs). Ainsworth et al present an example (Figure 34) where the root HTML page and embedded resources are captured by the archive at different times which may result in a composite memento that never existed on the live web [2, 49].

Figure 35 shows an abstract example where the archives serve a composite memento that has never existed on the live web. At t_0, a web page that consists of the base, or the root, HTML file *index.html* and the embedded image *foo.jpeg* is published on the live web. The archive captures the root page at t_1 and the embedded image at t_3. The archive tries to create another memento for the web page. It captures the root page at t_7, but before it captures the embedded image at t_9, the live web page including the base HTML file and the image have been changed (or updated) at t_8.

If at t_{10}, a client requests the composite memento created at t_1, the archive will return the memento *index.html* created at t_1 and the archived *foo.jpeg* closest to t_1 which is the memento (image) created at t_3. The composite memento "*index.html* of t_1 + *foo.jpeg* at t_3"

```
curl -i https://memgator.cs.odu.edu/timemap/link/https://climate.nasa.
    gov/vital-signs/carbon-dioxide/

HTTP/2 200
access-control-allow-origin: *
access-control-expose-headers: Link, Location, X-Memento-Count, X-
    Generator
content-type: application/link-format
date: Mon, 29 Jul 2019 22:42:51 GMT
x-generator: MemGator:1.0-rc7
x-memento-count: 4816

<https://climate.nasa.gov/vital-signs/carbon-dioxide/>; rel="original",
<https://memgator.cs.odu.edu/timemap/link/https://climate.nasa.gov/
    vital-signs/carbon-dioxide/>; rel="self"; type="application/link-
    format",
<http://web.archive.org/web/20141010072816/http://climate.nasa.gov:80/
    vital-signs/carbon-dioxide/>; rel="first memento"; datetime="Fri, 10
     Oct 2014 07:28:16 GMT",
<http://web.archive.org/web/20160304214529/http://climate.nasa.gov:80/
    vital-signs/carbon-dioxide>; rel="memento"; datetime="Fri, 04 Mar
    2016 21:45:29 GMT",
<http://archive.is/20141101160433/http://climate.nasa.gov/vital-signs/
    carbon-dioxide/>; rel="memento"; datetime="Sat, 01 Nov 2014 16:04:33
     GMT",
<http://archive.is/20170124235942/http://climate.nasa.gov/vital-signs/
    carbon-dioxide/>; rel="memento"; datetime="Tue, 24 Jan 2017 23:59:42
     GMT",
<http://wayback.archive-it.org/all/20150319152254/http://climate.nasa.
    gov/vital-signs/carbon-dioxide/>; rel="memento"; datetime="Thu, 19
    Mar 2015 15:22:54 GMT",
<http://wayback.archive-it.org/all/20180504082215/https://climate.nasa.
    gov/vital-signs/carbon-dioxide/>; rel="memento"; datetime="Fri, 04
    May 2018 08:22:15 GMT",
<https://arquivo.pt/wayback/20160515150329/http://climate.nasa.gov/
    vital-signs/carbon-dioxide//>; rel="memento"; datetime="Sun, 15 May
    2016 15:03:29 GMT",
<https://perma-archives.org/warc/20170419235215/http://climate.nasa.gov
    /vital-signs/carbon-dioxide/>; rel="memento"; datetime="Wed, 19 Apr
    2017 23:52:15 GMT",
...
```

Fig. 33: Retrieving the TimeMap of the URI-R `http://climate.nasa.gov/vital-signs/carbon-dioxide` using Memgator. The total TimeMap is aggregated from six different public web archives contains 4,706 mementos (only eight are shown)

has existed on the live web, so the archive returns the correct composite memento. However, if at t_{10}, the client requests the composite memento created at t_7, the archive will return

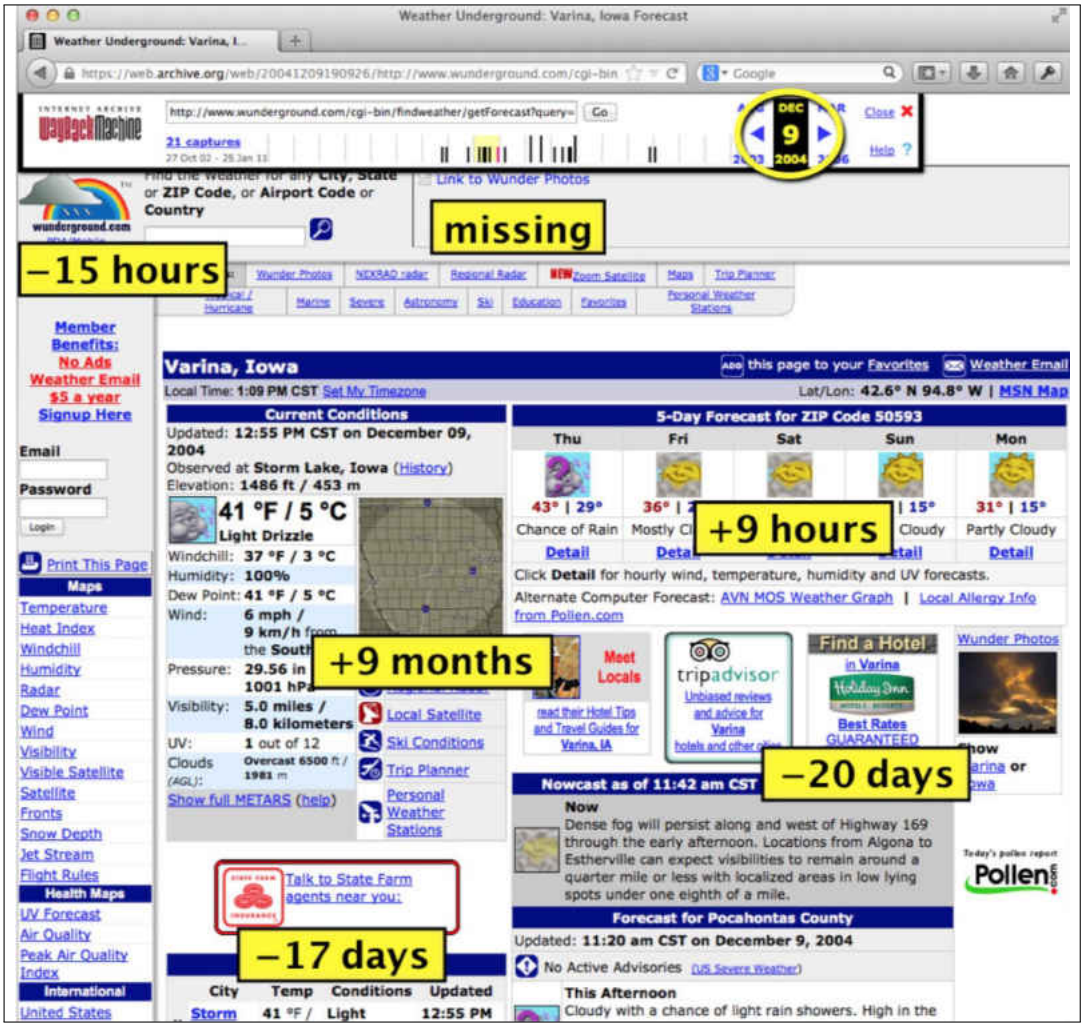


Fig. 34: A composite memento where the `Memento-Datetime` of the root page (December 09, 2004 19:09:26 GMT) is different from the `Memento-Datetime` of several embedded resources. (From [2]).

the memento *index.html* created at t_7 and the archived *foo.jpeg* closest to t_7 which is the memento created at t_9. The composite memento "*index.html* of t_7 + *foo.jpeg* at t_9" has never existed on the live web, so the archive does not return the correct composite memento.

It is common that some mementos frequently become unavailable in the archive because the indexes of the WARC files containing these mementos are temporarily unavailable. We can consider this as changes in TimeMaps since archives respond, at different times, with different TimeMaps of the same original resource *URI-R*. Although the TimeMap inconsistency seems to be unacceptable, archives behave this way for convincing reasons related to improving performance and responding quickly to clients' requests. The TimeMap

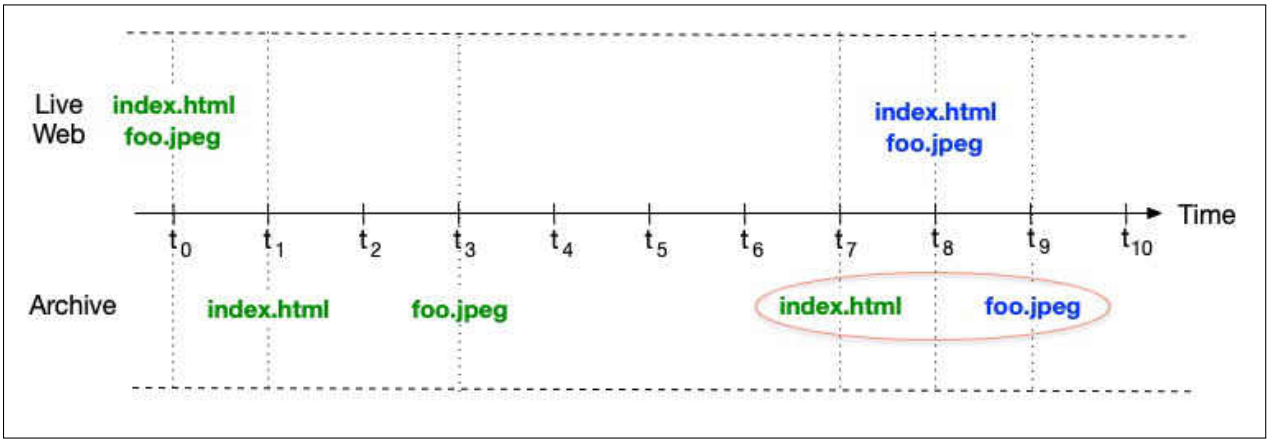


Fig. 35: Illustrating how web archive can serve composite mementos that never existed on the live web (e.g., the composite memento highlighted in red).

inconsistency may also occur for reasons including deduplication and redaction techniques of mementos, archival restructuring, and transient errors [102]. The TimeMap change may cause requests to the same memento to redirect to different URI-Ms. For example, in Figure 35, if the image with the `Memento-Datetime` of t_9 is not available at a particular time, the request might be redirected to the image archived at t_3 (if available).

2.4 VERIFYING THE FIXITY IS IMPORTANT TO ESTABLISH TRUSTED WEB ARCHIVES

The Trusted Repositories Audit & Certification report (TRAC) [103] by the Task Force on Archiving of Digital Information has introduced 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) according to Sections B2.9 and B4.4. The TRAC report recommends that preserved content is stored separately from its 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. In addition, the number of public and private web archives is increasing [14, 15], and we may not have the same level of trust in all of these archives.

2.5 CONVENTIONAL CRYPTOGRAPHIC HASHING ALGORITHMS VERSUS SIMILARITY ESTIMATION TECHNIQUES

As described in Chapter 1.4, common cryptographic hashing algorithms [104] such as MD5 and SHA256 produce entirely different outputs for even near-duplicate documents. This is useful for purposes like verifying fixity, password verification, finding duplicate records, and other security objectives. Figure 36 shows an example of how the hashing algorithm SHA256 produces two different hash values on near-duplicate paragraphs (i.e., one word has been changed from "one" to "two"). These hash algorithms demonstrate the fact that if there is a small change in the input, there would be a large change in the output.

```
1 $ echo "Klein et al. [31] conducted a study on over one million
    references from scientific articles and found that one in five
    articles suffers from Reference Rot, referring to links to web
    resources that no longer exist or that have significantly modified
    content." | shasum -a 256
2 b4948b7e7c300e3c836aea10f2864713703d09a549f720df7307e15c8a357ad4  -
3
4 $ echo "Klein et al. [31] conducted a study on over one million
    references from scientific articles and found that two in five
    articles suffers from Reference Rot, referring to links to web
    resources that no longer exist or that have significantly modified
    content." | shasum -a 256
5 c4b403726594a86d72c50d98f09f2e3a7c96920535b59daab4061b837097782b  -
```

Fig. 36: The hashing algorithm SHA256 produce entirely different hash values (lines 2 and 5) on near-duplicate documents.

The Merkle tree was introduced by Ralph Merkle in 1979 [146], and it is used in many applications, such as Blockchain-based networks (e.g., Bitcoin [137, 138]), to generate a hash value (root hash) on the content of large data structures. As shown in Figure 37, each leaf node in the Merkle tree is the hash of a block of data, while any non-leaf node (or intermediate node) is the hash of its child nodes.

Figure 38 shows an example of how the hash of the intermediate node `Hash 0` (from Figure 37) is computed. First, the hash values of the leaf nodes `L1` and `L2` are computed (as shown in lines 4 and 6). This produces two hash values `Hash 0-0` and `Hash 0-1`. The two hash values are then concatenated to each other and hashed, which produces the hash value of the intermediate node `Hash 0` (line 12). The hash value of any intermediate node in the Merkle tree is computed by following the same steps as described for `Hash 0`.

In contrast to MD5 and SHA256 functions, the hashing function SimHash [106,107] works

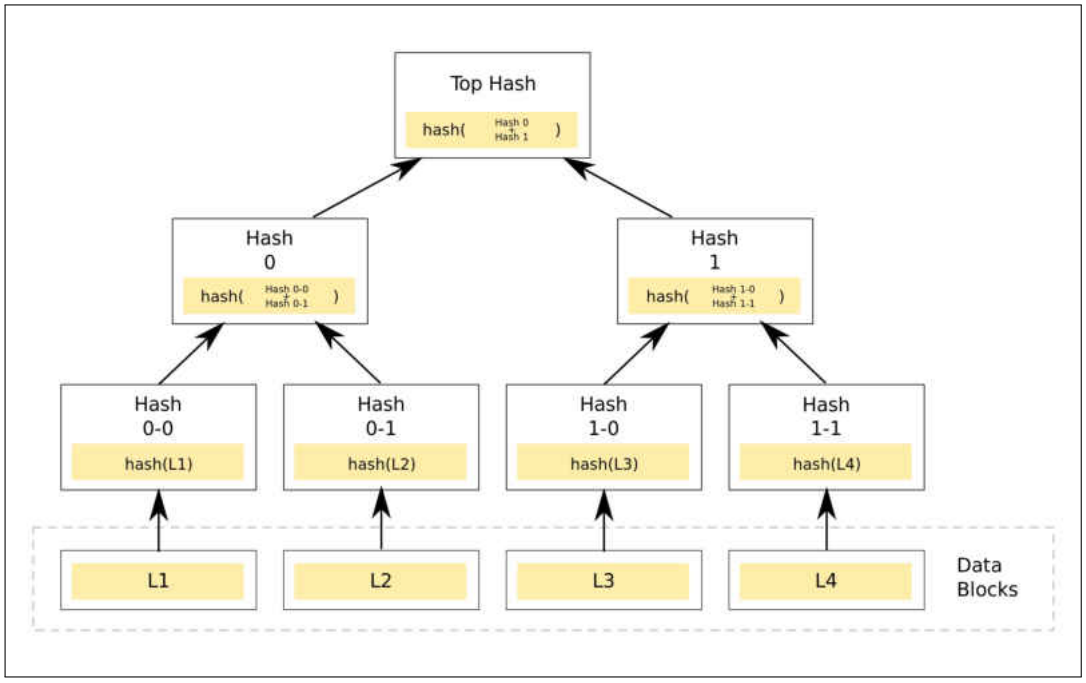


Fig. 37: Merkle tree for generating a root hash value (from [105]).

```
1  L1="foo"
2  L2="bar"
3
4  % echo "$L1" | md5sum
5  d3b07384d113edec49eaa6238ad5ff00  -
6  % echo "$L2" | md5sum
7  c157a79031e1c40f85931829bc5fc552  -
8
9  Hash0_0=d3b07384d113edec49eaa6238ad5ff00
10 Hash0_1=c157a79031e1c40f85931829bc5fc552
11
12 % echo  "$Hash0_0$Hash0_1" | md5sum
13 2d25c7b0aae4ec49c0a551d8d48ab9d7  -
14
15 Hash0=2d25c7b0aae4ec49c0a551d8d48ab9d7
```

Fig. 38: An example of how the intermediate node `Hash 0` (from Figure 37) is computed.

so that it produces similar hash values for similar documents. For instance, if there is a small difference between two documents, SimHash will generate hash values that are slightly different. SimHash is one of the document similarity techniques used by Google crawlers to

detect near-duplicate web pages for reducing redundancy in search results. Figure 39 shows the resulting hash values by applying SimHash (a Python module [108]) on the same input used in Figure 36.

```
1  $ python
2  >>> import re
3  >>> from simhash import Simhash
4  >>> def get_features(s):
5  ...     width = 3
6  ...     s = s.lower()
7  ...     s = re.sub(r'[^\w]+', '', s)
8  ...     return [s[i:i + width] for i in range(max(len(s) - width + 1,
       1))]
9  ...
10 >>> print '%x' % Simhash(get_features('Klein et al. [31] conducted a
       study on over one million references from scientific articles and
       found that @one@ in five articles suffers from Reference Rot,
       referring to links to web resources that no longer exist or that
       have significantly modified content.')).value
11 668d8ccdd966a785
12 >>>
13 >>>
14 >>>
15 >>> print '%x' % Simhash(get_features('Klein et al. [31] conducted a
       study on over one million references from scientific articles and
       found that @two@ in five articles suffers from Reference Rot,
       referring to links to web resources that no longer exist or that
       have significantly modified content.')).value
16 668d8cddd9e6a685
```

Fig. 39: The hashing algorithm SimHash produces slightly different hash values (lines 11 and 16) on near-duplicate documents. The characters marked in red indicate how different the two documents are.

There are other similarity estimation techniques, such Minhash [109]. SimHash and MinHash are widely used, and both algorithms are based on Locality Sensitive Hashing (LSH) [110]. Choosing one of these hashing techniques over the other depends on the application in hand. For example, in 2007, Google used SimHash for web crawling to detect duplicate web pages and used Minhash for Google News personalization [107, 111, 112]. To measure similarity between two documents, MinHash produces multiple hash values per document, while SimHash produces a single hash per document. Thus, SimHash requires less memory than MinHash. However, MinHash outperforms SimHash for documents with high similarity [113].

The SimHash and Minhash algorithms are originally designed for text-based near-duplicate detection. Other algorithms (e.g., pHash [114, 115]) are introduced to detect near-duplicate

images. The pHash algorithm generates 64-bit integer (hash) on an image through multiple steps that include: reducing the size of the image to 32x32 pixels, converting it to a grayscale, and constructing the final hash value. If the 64-bit hash values of two images are the same, then both images are very similar (not necessarily identical). The similarity score of two images is calculated by counting the number of different corresponding bits. Therefore, the similarity score of two images is between zero and 64: zero indicates very similar images, and 64 indicates that both images are very different.

2.6 CHAPTER SUMMARY

In this chapter we presented some basic concepts related to our research. We briefly explained the web architecture and how live web pages are rendered. Also, we described how archives crawl the web and replay mementos. We introduced some issues related to the replay of mementos including the effect of JavaScript on replayed mementos and why archives sometime do not give the correct mementos. We presented the WARC file format. Then, we briefly explained the Memento framework and Memento aggregators. We described some fixity-related approaches including cryptographic hashing algorithms and SimHash.

CHAPTER 3

RELATED WORK

In this chapter, we describe some threats and attacks against web archives that can affect the representation of replayed mementos. We explain how cryptographic hashes in URIs (e.g., using trusty URIs and Multihash) can help to verify the fixity of published web resources. We point to some tools for trusted timestamping mementos using blockchain-based network. We briefly explain how the opinion poll protocol works in the LOCKSS system, which emphasizes the importance of replication.

3.1 WEB ARCHIVING SECURITY ISSUES

In this section we describe multiple security threats against web archives, which emphasizes the importance of verifying the fixity of archived pages. Several vulnerabilities were discovered in the Internet Archive's Wayback Machine by Lerner et al. [3] that can be leveraged by attackers to modify a user's view at the time when a memento is rendered in a browser:

- **Archive-Escapes**: Archived web pages are supposed to be retrieved from the archive. However, there may be links generated by JavaScript on the client (at replay time) that are not rewritten by the archive (Chapter 2.2.6). Such links will be loaded from the live web. Brunelle et al. [5] also provide some examples that illustrate the effect of live resources linked from archived pages. Therefore, if attackers can own and control the original server that hosts the live resource linked from the archive, then they can inject malicious code (e.g., JavaScript) to change the client's rendered view of the archived page.

- **Same-Origin Escapes**: Before or at the time of archiving a web page (i.e., at crawl time), the attacker may able to inject the malicious code in a third-party `<iframe>` which cannot modify the main HTML file because of the Same-Origin Policy in the live web, but once the page is archived, both the main HTML file and the `<iframe>` will be loaded from archive.org at replay time, and the Same-Origin Policy becomes ineffective.

- **Archive-Escapes + Same-Origin Escapes**: The attacker can use `Same-Origin Escapes` to create the first attack `Archive-Escapes` by injecting malicious JavaScript code, which can control the representation of the replayed archived page.

- **Anachronism-Injection**: If a composite memento contains at least one resource that has never been captured by the archive, then the attacker, who has access to the original server hosting that never-archived resource, can publish a malicious version of that resource, and then submit it to the archive. The archive will capture the malicious resource which will be embedded in the composite memento.

We describe one example of such attacks. Figure 40 shows how the presentation of a memento can be changed by the Archive-Escapes attack. Lerner et al. [3] developed a proof-of-concept implementation of the Archive-Escapes attack. They found that the memento (root URI-M)

`http://web.archive.org/web/20110901233330/reuters.com`

has the live web embedded resource

`http://cdn.projecthaile.com/js/trb-1.js`

The URI-R to the embedded resource was not rewritten by the archive because it is generated on the client-side by JavaScript (as explained in Chapter 2.2.5). Using `whois` [116], the authors found that the domain name `projecthaile.com` was not owned. They purchased this domain name and injected malicious code in `trb-1.js`. The browser will try to load all resources embedded in the page. Thus, when loading `trb-1.js`, the malicious code in the JavaScript will change the representation of the memento. The Archive-Escapes attack causes the user's view to be completely controlled by the malicious code without even the need to compromise the Internet Archive from which the memento is delivered.

Lerner et al. suggested some defenses that could be deployed by either web archives or web publishers to prevent the exploit of these vulnerabilities. As a result, the Internet Archive mostly solved this issue by using the `Content-Security-Policy` HTTP header [43]. The `Content-Security-Policy` response header notifies a user-agent (e.g., a web browser) to not load resources except from specified domains (e.g., an archive's domain).

The problem of having live web resources linked from archived pages may still occur in the Internet Archive and other archives because there are several methods that can load live web resources into archived pages as explained by Nelson [38, 42].

(a) Before the Archive-Escapes attack.

(b) After the Archive-Escapes attack.

Fig. 40: A proof of concept of Archive-Escapes attack. The main image in the memento `https://web.archive.org/web/20110901233330/reuters.com` has changed after inserting malicious code (from Lerner et al. [3]).

Cushman and Kreymer created a shared repository [117] in May 2017 to describe potential threats [36, 118] in web archives. They described seven different threats (some of which overlap with those from Lerner et al. [3]):

1. **Archiving local server files**: When crawling a web page, the archive may find links to local resources in the archive itself (e.g., via `http://localhost:8090/<file-path>`), and these resources should not be captured and should not be publicly available.
2. **Hacking the headless browser**: Web archives use web crawlers, such as the Internet Archive's Heritrix [72], to collect web pages. However, several web archives including `archive.is` and `Webrecorder` (now named Conifer) [119] have started to employ more modern headless browsers (e.g., PhantomJS [120]) which all have known vulnerabilities.
3. **Stealing user secrets during capture**: After crawling a composite page by a user-driven crawler, the standard cross-domain policies will not be effective at replay time, which may result in stealing the user's sensitive information. (This vulnerability is related to the `Same-Origin Escapes` attack.)
4. **Cross-site scripting to steal archive logins**: If the domain name used to let users log in to the archives is the same as the domain name used by the archive to replay archived pages, then the admin of an archived page may be able to steal login information of the archive users (e.g., login cookie) because, similar to the `Same-Origin Escapes` attack, the cross-domain policies will be ineffective.
5. **Live web leakage on playback**: As explained in Chapter 4.1.9, JavaScript may create links to live web resources that are not rewritten by the archive causing the `Archive-Escapes` attack.
6. **Showing different page contents when archived**: It is possible that the original server identifies that a web page is being crawled by the archive, the server then can reply with content that is different from the normal content of the page. It is also possible that the page is designed so that it knows it is being replayed from the archive, so it shows different content.
7. **Banner spoofing**: Malicious code can also be used to change the appearance of the archive's banner. This is similar to the `Archive-Escapes`.

Some of the attacks, described above, may not have direct impact on generating repeatable fixity information. For example, by stealing a user's login, the attacker may not be able to change the content of archived pages. However, other attacks may directly change the appearance of replayed web pages causing different fixity information. Therefore, we need a technique that allows users to verify the fixity of archived pages. The attacks and vulnerabilities that may prevent us from generating repeatable fixity information on the playback of mementos include:

- **Archive-Escapes**
- **Same-Origin Escapes**
- **Archive-Escapes + Same-Origin Escapes**
- **Anachronism-Injection**
- **Live web leakage on playback**
- **Banner spoofing**

The authors provide recommendations on how to avoid such threats. For example, the first threat "Archiving local server files" occurs when a web crawler finds links that point to local files (e.g., `http://localhost:8080/path/to/file`), which should not be public. To protect local files, a web crawler should only use Hypertext Transfer Protocol Secure (HTTPS) [121, 122] when collecting web pages. The other solution is to run a web crawler in a virtual machine (VM) [123], which adds another layer of protection by preventing the running crawler from accessing local files.

Rosenthal et al. [37] 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 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.

3.2 IDENTITY DERIVED FROM CONTENT (HASHES IN URIS FOR FIXITY)

There have been approaches for embedding fixity information (e.g., hash values) in the URI of a web resource. In this case, the URI has two purposes. First, it is used to retrieve, via an HTTP request, the resource from the server. Second, the URI is used to verify that the delivered content is actually the requested content.

In this section we describe two approaches, the trusty URI and Multihash, which are based on including a hash value in a URI. The hash value is calculated on the content that the URI identifies.

3.2.1 TRUSTY URIS

Kuhn et al. [124, 125] define a trusty URI as a URI that contains a cryptographic hash value of the content it identifies. The authors introduced this technique of using trusty URIs to make digital artifacts, especially those related to scholarly publications, immutable, verifiable, and permanent. 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 (e.g., trusty URIs to prevent man-in-the-middle attacks [126]). In addition, trusty URIs can verify the content even if it is no longer found at the original URI but still can be retrieved from other locations, such as Google's cache and web archives.

In their second paper [125], 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 module R, the hash is calculated on RDF graphs. Kuhn et al. introduce the module R to indicate that there should be different guidelines for generating hash values on different types of content instead of having only one byte-level-based technique. For example, users should use the module R on RDF files. This allows producing the same hash value for a single RDF graph even if the RDF graph is serialized in different RDF file formats, which is not possible when using the module F. While calculating the hash value on content of type F (byte-level file content) is a straightforward task, multiple steps are required to calculate the hash value on content of type R (RDF graphs). This includes such as converting any serialization (e.g., N-Quads [127] or TriG [128]) into RDF triples [129,130], sorting of RDF triples lexicographically, serializing the graph into a single string, replacing newline characters with "\n", and dealing with self-references and empty nodes.

Figure 41 illustrates the general structure of trusty URIs. The artifact code, everything after r1, is the part that makes this URI a trusty URI. The first character in this code (R) is to identify the module. R, in the example in Figure 41, indicates that this trusty URI was generated on a RDF graph. The second character (A) is to specify the version of this module. The remaining characters from "5" to "0" represent the hash value calculated on the content. All hash values are generated by the SHA256 algorithm.

There are multiple trusty URI implementations in different languages including Python, Java, and Perl [131–133]. For example, the Python implementation of trusty URIs *Trustyuri-python* computes (via the script *ProcessFile.py*) a hash value, or artifact code, on the content of a file and renames the file based on the resulting hash. The library also can verify (via *CheckFile.py*) a trusty URI by comparing a hash value included in the trusty URI with a current hash value calculated on the content obtained after dereferencing the trusty URI.

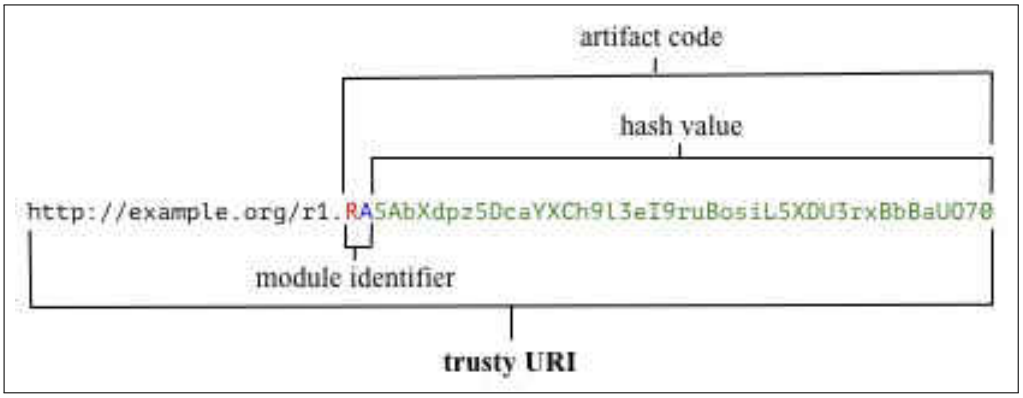


Fig. 41: The general structure of trusty URIs. The character in red is the module type, the character in blue is the module version, and the characters in green are the hash value.

To illustrate how trusty URIs work, we use *Trustyuri-python*. First, we download one of our research papers [134] using `cURL` as shown in Figure 42 (line 1). Second, we run the Python script *ProcessFile.py* which will generate a hash value on the content of the PDF file *tpdl-2015.pdf* and rename it based on the resulting hash value to `tpdl-2015.FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh9gqFVao.pdf`. Third, we publish the trusty file on the web at the trusty URI `http://www.cs.odu.edu maturban/pubs/tpdl-2015.FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh9gqFVao.pdf`. Anyone with this trusty URI can verify the fixity of the content using *CheckFile.py*. As shown in Figure 43, the output `Correct hash: FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh-9gqFVao` indicates that the hash value in the trusty URI is identical to the hash value of the content obtained from dereferencing the trusty URI. Thus, the trusty URI is verified.

```
1 $ curl --silent  https://www.cs.odu.edu/~maturban/pubs/tpdl-2015.pdf --
    output tpdl-2015.pdf
2 $ python ProcessFile.py tpdl-2015.pdf
```

Fig. 42: The `cURL` downloads one of our research paper. The downloaded file will be named *tpdl-2015.pdf*. The Python script `ProcessFile.py` computes a hash value on the content of the file *tpdl-2015.pdf* and renames *tpdl-2015.pdf* based on the resulting hash value that ends with `Gm1vTOcCqrsWLKeeICh9gqFVao.pdf`).

To see how the library detects any change in the original content, we replace all occurrences of the number "61" in the original paper *tpdl-2015.pdf* with the number "71" via the commands shown in Figure 44. Figure 45 shows part of the first page of the paper before and after changes. The library detects that the original resource has been changed as shown

```
python CheckFile.py http://www.cs.odu.edu/~maturban/pubs/tpdl-2015.
    FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh9gqFVao.pdf

Correct hash: FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh9gqFVao
```

Fig. 43: Verify a trusty URI using the Python script *CheckFile.py*.

in Figure 46 using the Python script *CheckFile.py*.

```
$ pdftk tpdl-2015.FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh9gqFVao.pdf
    output tmp.pdf uncompress
$ sed -i "s/61/71/g" tmp.pdf
$ pdftk tmp.pdf output tpdl-2015.
    FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh9gqFVao.pdf compress
```

Fig. 44: Modify the content of the trusty file by replaying all occurrences of "61" with "71".

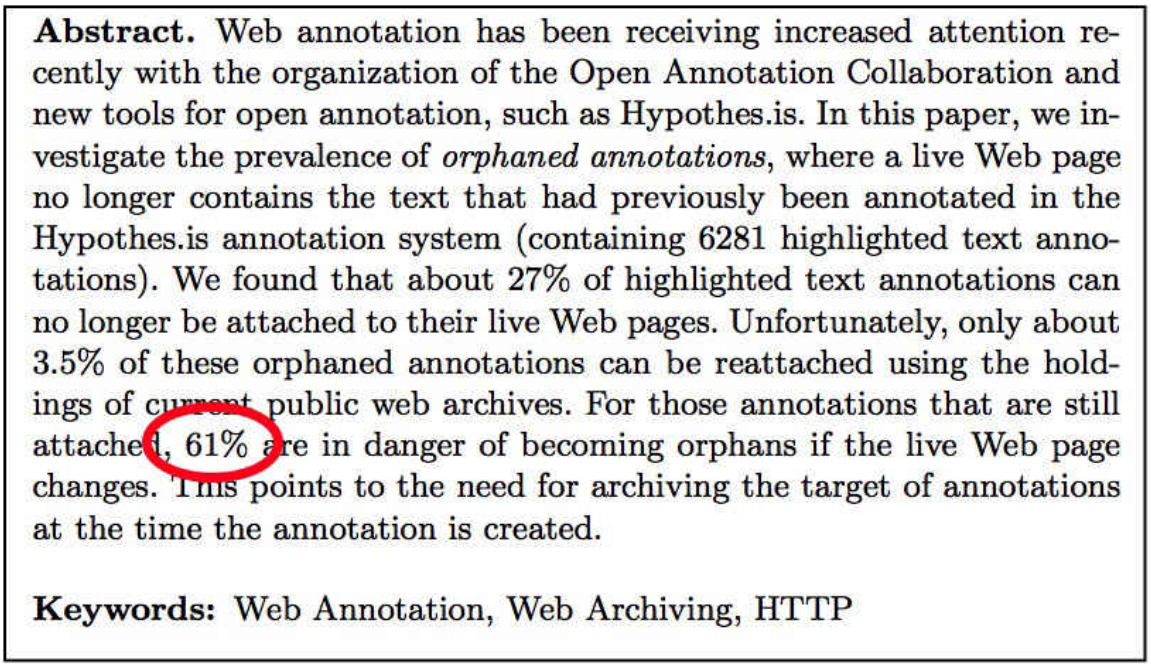
Even though trusty URIs can be used to detect altered documents, there are some limitations. First, trusty URIs can be generated on only two types of content, RDF graphs and byte-level content (i.e., no modules have been introduced for HTML documents). Second, one hash function SHA-256 is used to generate the hash value, which might not be suitable for some use cases or when the algorithm becomes vulnerable.

3.2.2 MULTIHASH

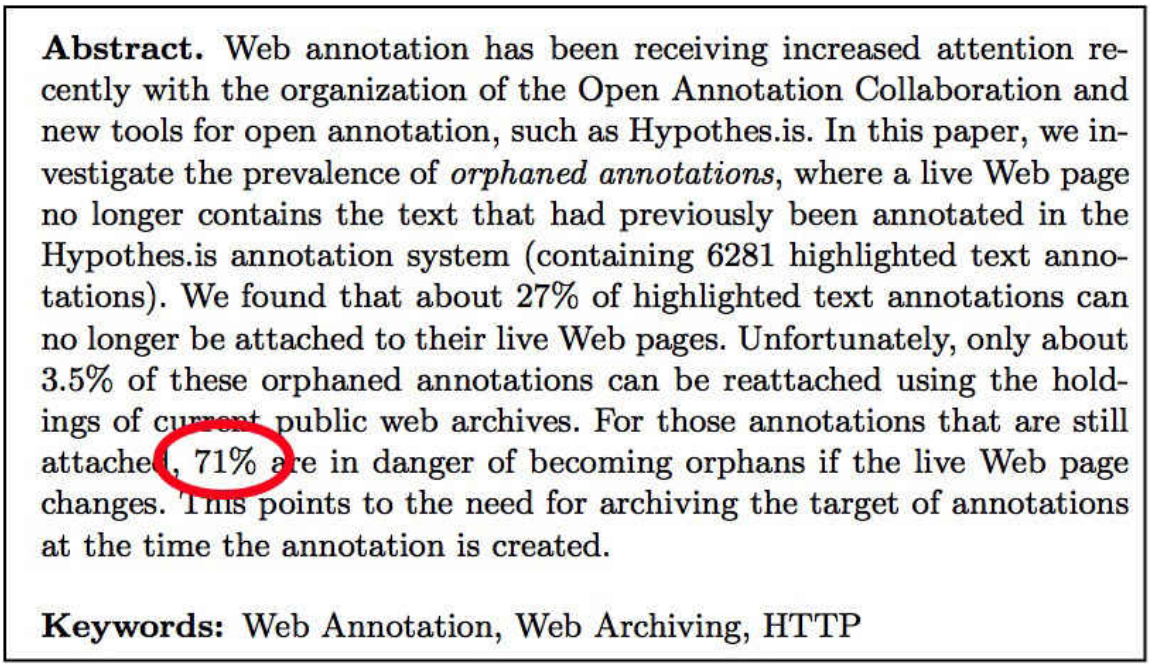
Multihash [135] is a protocol introduced by Juan Benet mainly to create self identifying hashes for IPFS content [136], but the protocol can be utilized to identify other content like regular web pages. Figure 47 shows the structure of self-described hashes by Multihash [135]. The elements of this structure include the following:

- **Hash function code**: This is an integer to indicate the hash function used to generate the hash value. For example, `0x11`, `0x12`, and `0x13` identify SHA1, SHA2-256, and SHA2-512, respectively. The default codes are available on Github[1].
- **Digest size in bytes**: This an integer representing the number of bytes in the hash value.

[1] `https://github.com/multiformats/multihash/blob/master/hashtable.csv`



(a) The paper before the change is made.



(b) The paper after the change is made.

Fig. 45: A small change is made to the paper

- **Hash function output**: This is the actual hash value.

```
1 $ python CheckFile.py http://www.cs.odu.edu/~maturban/pubs/tpdl-2015.
    FAofcNax1YMDFakhRQvGm1vTOcCqrsWLKeeICh9gqFVao.pdf
2 $ *** INCORRECT HASH ***
```

Fig. 46: Verifying a trusty URI using the Python script *CheckFile.py*.

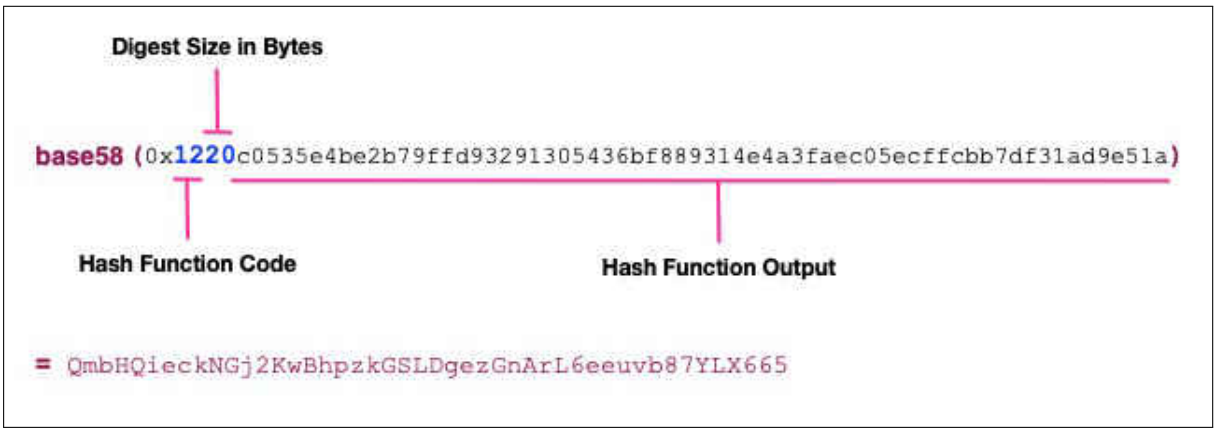


Fig. 47: The structure of self-described hashes by Multihash.

The InterPlanetary File System (IPFS) uses Multihash as a mechanism to generate hashes calculated on the content of files and directories on the IPFS network. For instance, the first two characters "`Qm`" in the IPFS address

`/ipfs/QmZTR5bcpQD7cFgTorqxZDYaew1Wqgfbd2ud9QqGPAkK2V`

have the information about the hash function (i.e., SHA256) used to generate the hash value `ZTR5bcpQD7cFgTorqxZDYaew1Wqgfbd2ud9QqGPAkK2V` and the size of the hash value, or digest.

IPFS files can be accessed through the public gateway `https://ipfs.io/ipfs/<hash>` where `<hash>` is a Multihash hash of a file or a directory. For example, to download the file identified by the hash

`QmZTR5bcpQD7cFgTorqxZDYaew1Wqgfbd2ud9QqGPAkK2V`

from the IPFS network, we can use `cURL`:

```
$ curl https://ipfs.io/ipfs/QmZTR5bcpQD7cFgTorqxZDYaew1Wqgfbd2ud9QqGPAkK2V
```

There are some differences between the trusty URI described in Chapter 3.2.1 and Multihash:

- Trusty URIs are generated using SHA256 only while Multihash provides the ability to use any popular hash algorithm.
- Hashes generated by Multihash can be in any configured size while in trusty URIs only a hash output of size 256 bits is used
- Trusty URIs support self-references (i.e., when trusty URIs are included in the content).
- Well-defined modules are introduced to create trusty URIs on byte-level content and RDF graphs while in Multihash only byte-level content is supported

3.3 TRUSTED TIMESTAMPING

Timestamping is recording the date and time of when an event occurs. For example, the HTTP Response headers `Date` and `Last-Modified` are examples of timestamps referring to different events (i.e., `Date` indicates when a server generated a response message, while `Last-Modified` is the datetime of when the resource was last modified). A "trusted" timestamp is a timestamp initially created and verified by a third-party trustworthy service. Blockchain-based networks (e.g., Bitcoin [137,138]) have been receiving increased attention recently as trustworthy systems for initiating and validating timestamps of digital documents. Once a file is timestamped in the blockchain, anyone should be able to prove the existence of the file at a particular point in time.

Generating a hash value on the content of a memento is one of the crucial parts in the process of timestamping the memento. As shown in Figure 48, the memento is not directly timestamped in the blockchain. Instead, a hash value calculated on the content of the memento is the data to be timestamped. Thus, it is important to be able to reproduce the same hash of a particular memento over time. The difficulties of generating repeatable hashes is discussed in Chapter 6.

The Bitcoin blockchain [137] is a peer-to-peer electronic cash system built using the Blockchain technology [138]. A ledger that contains all transactions in Bitcoin is duplicated across all nodes in the network (i.e., there is no central agency). The timestamp associated with each transaction indicates when the transaction is accepted in the Bitcoin. Services,

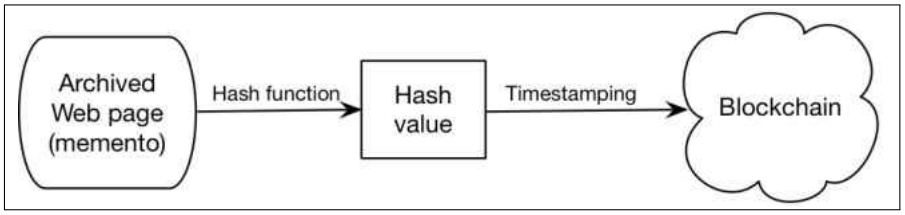


Fig. 48: Timestamping a hash value that summarizes a memento in the blockchain.

such as OriginStamp [139], Chainpoint [140,141], and OpenTimestamps [142,143], generate trusted timestamps in Bitcoin for digital documents. Even though timestamping steps might vary from one service to another, they follow a common procedure:

1. Receiving a file, a hash, or plain text from a user
2. Generating a hash value of received content
3. Converting the hash to a Bitcoin address
4. Issuing a Bitcoin transaction using the Bitcoin address as a money sender or receiver

To verify timestamps in Bitcoin at any point in the future, the first three steps mentioned above are performed. The fourth step then would include issuing a query through the Bitcoin API to obtain information about any transactions on the given Bitcoin address. We consider the timestamp associated with the Bitcoin transaction as a trusted timestamp. Being incorruptible is the key characteristic of Bitcoin as any change in a transaction or a block requires computational power that exceeds the entire network, which is theoretically possible but unlikely to occur practically. The other important feature of Bitcoin is the decentralization of a distributed ledger which contains all transactions ever made in Bitcoin (i.e., the ledger is duplicated across all nodes).

Tools have been developed to generate trusted timestamps in blockchain-based networks. OriginStamp [144] allows users to submit plain text, a hash value, or any file format (e.g., PDF/PNG files). The data is not sent to the OriginStamp's server. Instead, it is hashed in the user's browser and only the resulting hash is transmitted to the 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 (i.e., the source or destination of a transaction in Bitcoin). The timestamp associated with the transaction is considered

a trusted timestamp. OriginStamp provides an instant timestamping in the Bitcoin if a user is willing to pay a Bitcoin transaction fee. A user can verify a timestamp through OriginStamp's API or by visiting their website. The server first receives a hash from a user, then OriginStamp converts the hash to a Bitcoin address and sends a query to Bitcoin's API. If any transaction involved the given address is returned, the timestamp associated with the transaction can be used as a proof of existence. In addition to the process of verifying timestamps through OriginStamp's website, users may verify timestamps directly in Bitcoin.

Other services, such as Chainpoint, Proof of Existence [145], and OpenTimestamps, are based on the same concept of using blockchain (e.g., Bitcoin) to timestamp digital documents. Some differences between these tools include:

- *Cost* - The OriginStamp service can be used with no charge unless users want an instant submission to the blockchain. Users of Proof of Existence, on the other hand, have to pay some fees for the service.

- *Generation of aggregated hashes* - In OriginStamp, an aggregated hash is computed by storing all hashes received within a day (i.e., 24 hours) in a file, which then will be hashed to generate a single aggregated hash. Chainpoint and OpenTimestamps use a Merkle Tree [146] to generate one aggregated hash (i.e., root hash).

- *The number of Bitcoin transactions v. hashes* - Services like OriginStamp, Chain-Point, and OpenTimestamps support issuing either one Bitcoin transaction per submitted hash or one transaction per aggregated hash. Other tools, such as Proof of Existence, create one Bitcoin transaction per hash.

- *Use* - OriginStamp and Proof of Existence provide online services through their websites that allow users to create or verify trusted timestamps using a web browser. Chainpoint and OpenTimestamps require installing client software in order to use the timestamping service.

- *Blockchain-based network* - Bitcoin is commonly used by all of these services to generate trusted timestamps. In addition, Chainpoint can create timestamps using other blockchain networks like Ethereum [147].

Even though users of the tools mentioned above can pass data by value, such as plain text, any file format, or a hash value, they are not allowed to submit data by reference (i.e.,

passing a URI of a web page). In other words, these services are not directly timestamping web pages. The only exception is an additional service [4] established by OriginStamp. The service works by receiving a URI from a client and hashing the content of the web page identified by the URI before submitting to the blockchain. Figure 49 (from [4]) shows the UI of this service where users can search for timestamped web pages by entering a URI. There are two disadvantages of this additional service. First, the service is no longer available on the live web[2]. Second, 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 [4]. As illustrated in Chapter 1.4, not including embedded resources in hash calculation may leave the page vulnerable to undetected changes.

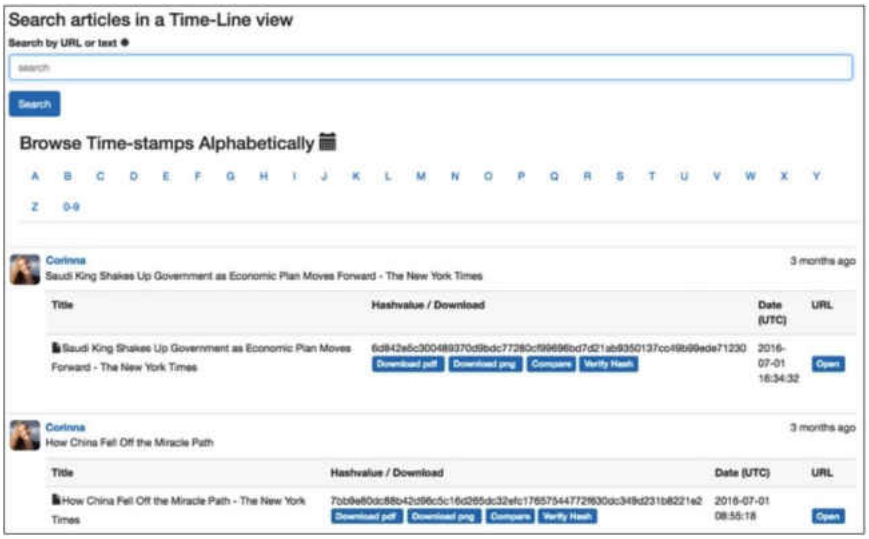


Fig. 49: A list of timestamped web pages. Users can search for a particular web page by typing a URI or text (from [4]).

Collomosse et al. introduced ARCHANGEL [148]. The main difference between other timestamping services and ARCHANGEL is that it was developed with the objective to timestamp mainly archived pages at time of content deposition using blockchain. The authors introduced a prototype implemented using Ethereum, and they plan to investigate different ways to generate hash values on different format, such as PDFs, images, and videos. The authors did not describe whether the approach can work with composite mementos.

Branwen [1] wrote a shell script to calculate a hash value by considering all resources constructing a web page (e.g., images and stylesheets) in addition to the base HTML file.

[2] `https://www.isg.uni-konstanz.de/web-time-stamps/`

This seems to be a reasonable solution for timestamping web resources, but without considering other factors, such as the effect of JavaScript on replayed web pages, transient errors, and content added by a third-party services, it is difficult to produce a repeatable hash for the same web page over time.

In summary, timestamping in blockchain-based networks can be used to generate trusted timestamps, but regarding timestamping archived web pages we showed that state-of-the-art timestamping services in blockchain-based networks do not allow users to submit URIs (i.e., not accepting data by reference). Second, it is hard to generate repeatable hash values on the content of archived web page as will be explained in Chapter 6.

3.4 THE IMPORTANCE OF REPLICATION

Preserving only a single copy of a digital resource makes it subject to threats described in Chapter 3.1. To maintain archived resources and make them accessible for a long period of time, multiple copies of the same resource should be preserved [37]. For example, the Internet Archive makes a complete copy of the Wayback Machine (all mementos captured between 1996 and 2002), in addition to thousands of other digital resources including films and television broadcast, available at Bibliotheca Alexandrina [149]. The Internet Archive has other backup collections available in Amsterdam, the Netherlands, and University of Toronto and University of Alberta in Canada [150, 151].

The Lots of Copies Keep Stuff Safe (LOCKSS) system [152] is a peer-to-peer digital preservation system used by libraries worldwide to preserve published scholarly content. The system allows each subscribed library to have its own copy of a document if the library has access permission from a publisher. Thus, the number of copies of a document in LOCKSS network depends on the number of libraries that have access to the document and collect it. The auditing mechanism in LOCKSS allows libraries, or nodes, in the network to compare same content with each other through a particular voting protocol, the opinion poll protocol. If a node identifies a corrupted document (i.e., it does not agree with the majority), it can request a valid copy from an other node (that agrees with the majority). The node that receives a repair request will send a valid copy only if it is aware that the requester used to maintain an uncorrupted copy. Here are the simplified steps that a node in LOCKSS follows for verifying the fixity of an archival resource, or an archival unit (AU), and repairing it if damaged [153]:

1. A LOCKSS peer (or poller) starts a poll for verifying the fixity of the content of an AU.

2. Because each node has a list of a small number of peers that maintain the same AU, it invites these peers to the poll.

3. Each of these peers (or voters) can accept to participate in the poll or reject if it is busy with other tasks.

4. Each voter computes a hash value on the content of its own version of the AU and sends it back (i.e., the vote) to the poller.

5. Based on the votes received, the poller makes one of the three decisions (voters do not need to be informed about the poll's results):

   (a) If the majority of voters agree with the poller's version of the AU, the fixity of the AU is verified and it waits for the next regular poll.

   (b) If the majority of voters disagree with the poller's version of the AU, the fixity of the AU fails and the poller will randomly select one of the voters that disagrees with it to repair the damaged AU. The selected voter sends its version of the AU to the poller, and the poller then starts a poll on the received AU.

   (c) If the majority of votes do not agree nor disagree with the poller's version of the AU, human interaction is required.

6. If the poller does not have enough votes (e.g., less than a certain threshold) to start a poll, the voters can invite other peers to the poll.

These are several reasons why the LOCKSS opinion poll protocol cannot be used to verify the fixity of mementos preserved in public web archives:

- LOCKSS peers communicate with each other, for example, to repair damaged archival units, invite other peers to a poll, and other activities. On the other hand, web archives generally do not communicate with each other, and they may use different set of tools and APIs.

- In general, the number of peers in LOCKSS (more than 200 peers) is greater than the number of public web archives (fewer than 40 independent web archives [154]). Moreover, the number of web archives that support on-demand archiving is around six archives.

- Each LOCKSS peer is considered independent and allowed to participate in polls. However, not all web archives are independent. For example, `archive-it.org` is a subscription-based service developed by the Internet Archive. Several European web archives might be managed by the same organization (e.g., European Union).
- Some web archives might be independent but not "trusted" (e.g., *Michael's Evil Wayback* introduced in Chapter 1) resulting in even smaller margins.
- Discovering copies of an AU in LOCKSS is faster than discovering mementos in web archives. In LOCCKS, each peer keeps a list of peers that have copies of an AU. In web archives, we have to use tools like Memento aggregators to discover mementos.

3.5 COMPUTING FIXITY INFORMATION USING WEB PACKAGING

Web packaging [155, 156] is an emerging standard. It has been introduced with the objective to distribute web pages packaged in a way that allows receivers (e.g., a user-agent) to verify that the content is from a particular origin and to view the content offline. The web packaging specification defines three related layers:

1. **Signed HTTP exchanges** [157]: This allows an origin to sign an HTTP Exchange (i.e., an HTTP request and corresponding response) with its cryptographic private key.
2. **Bundled HTTP exchanges** [158]: Multiple signed (or unsigned) HTTP exchanges are aggregated into a single "bundle" along with metadata to indicate how the bundle should be interpreted.
3. **Loading Signed Exchanges** [159]: This specifies several algorithms for validating digital signatures.

In addition to the WARC format, a bundle in web packaging can be viewed as another way of aggregating web resources in a single file. Web packaging can also be considered as replacement of WARC format in the future [160]. Alam et al. [161] suggest some improvements to the current web packaging specification so that it meets the need of the web archiving community, such as extending Loading Signed Exchanges to support time-based content negotiation.

Related to our work of verifying the fixity of composite mementos, web packaging theoretically can be used to download a composite memento, packaged in a bundle, with a single HTTP request. This should reduce playback-related changes, such as transient errors and TimeMap changes. However, we are not able to use web packaging (i.e., bundles) to deliver composite mementos because web packaging currently is not supported by any web archive even though web packaging might become an Internet standard in the near future [162]. Browsers that support web packaging in the future can render composite mementos, received in bundles, without the need to load more resources from the archive.

3.6 CHAPTER SUMMARY

In this chapter, we presented some threats and attacks against web archives that can affect the archived content at ingest time, in the archive, or at the replay time. We explained two mechanisms that use cryptographic hash values in URIs (i.e., trusty URIs and Multihash). These hash values are computed on the content of resources that the URIs identify, so they can be used to verify the fixity of resources. We point to some tools (e.g., OriginStamp, Chainpoint, and OpenTimestamps) for trusted timestamping mementos using blockchain-based network. We briefly explain how the opinion poll protocol works in LOCKSS system, and why we cannot use this protocol to verify the fixity of memento in web archives.

CHAPTER 4

DEFINING AN ARCHIVE-AWARE HASHING FUNCTION

This chapter contributes toward addressing RQ1 and RQ2:

RQ1: Can we identify and quantify the types of changes on the playback of mementos that prevent generating repeatable fixity information?

RQ2: Given the types of changes identified in the playback of mementos, can we define the final guidelines for generating repeatable fixity information (defining an archive-aware hashing function)?

This chapter describes our archive-aware hashing function for generating fixity information on the playback of mementos. Developing an archive-aware hashing function that works on archived web pages is important as conventional hashing functions (e.g., `MD5`) are not suitable for replayed archived web pages as described in Chapter 1.4 and 1.5, and in details in Chapter 6. We, initially, introduce several guidelines that the hashing function should follow for generating fixity information (Chapter 4.1). We notice that even if these predefined guidelines are considered, our archive-aware hashing function still produces different aggregated hash values on the same mementos over time. For that, we conduct a study (Chapter 6) to understand, identify, and quantify changes on the playback of mementos causing different fixity information. The study is conducted on a dataset of 16,627 composite mementos selected from 17 public web archives. (Chapter 5 describes multiple methods we used to collect the dataset.) We download each composite memento several times and compute their fixity information using our initial archive-aware hashing function. The results of the study highlight the need to add additional guidelines (Chapter 4.2) for generating fixity information. The new guidelines include the use of multiple hash values representing each composite memento. These hash values can help us detect which resource in a composite memento has changed over time.

Chapter 7 describes our framework that we use to store and discover fixity information. The framework also explains how we can verify the fixity of archived pages using fixity information previously generated and pushed into web archives.

4.1 INITIAL GUIDELINES FOR GENERATING FIXITY INFORMATION ON THE PLAYBACK OF MEMENTOS

In this section, we describe several guidelines we should follow to generate fixity information on the playback of mementos.

4.1.1 FIXITY INFORMATION SHOULD BE GENERATED ON A COMPOSITE MEMENTO

When generating fixity information on a composite memento, it is important that we include all resources comprising the composite memento. If we consider only the base HTML file, changes in embedded resources (e.g., an image) will not be detected. In other words, we will get false negative results indicating that the composite memento has not been changed as illustrated in the following scenario.

We first pushed the web page (Figure 50(a))

`https://www.bea.gov/news/2020/us-international-trade-goods-and-services-april-2020`

into a private web archive, *Evil Wayback*[1] (`evil-wayback.github.io`), on June 15, 2020. The URI-M of the memento is

`https://evil-wayback.github.io/web/20200615114013/www.bea.gov/news/2020/us-international-trade-goods-and-services-april-2020/`

We replayed the memento at two different times, on June 15, 2020 and June 17, 2020, which resulted in two slightly different representations as shown in Figures 50(b) and 50(c).

Figure 51 shows an example of calculating the hash on only the base HTML file. The `cURL` command downloads the HTML of the memento, and then the hashing function `sha256sum` generates a `SHA-256` hash on the resulting HTML. As expected, this simple technique does not detect changes in the embedded image as it produces the same hash value that ends with `221a` on June 15, 2020 and on June 17, 2020.

Figure 52, on the other hand, shows a shell script that generates one aggregated hash on a composite memento by including all embedded resources in the hash calculation through the following steps:

1. Download a composite memento, including the base HTML file and all embedded resources, using Wget [51] (line 1 in Figure 52). The output files will be stored locally in a directory hierarchy related to the paths of the files in the server.

[1] We created this archive to demonstrate different scenarios

(a) The live web page on June 15, 2020.

(b) Replaying the archived page on June 15, 2020.

(c) Replaying the archived page on June 17, 2020.

Fig. 50: Replaying a memento at two different times. An image in the memento has changed, and without including the image in the hash calculation, we will not be able to detect these changes.

```
1 $date
2 Mon Jun 15 17:20:37 EDT 2020
3 $ curl -s https://evil-wayback.github.io/web/20200615114013/www.bea.gov
    /news/2020/us-international-trade-goods-and-services-april-2020/ |
    shasum -a 256
4
5 01c8cd5606e22bb69799ba567cb41aae3d9873aea6215dba19acdbb0b56e221a  -
6
7 $date
8 Wed Jun 17 19:01:14 EDT 2020
9 $ curl -s https://evil-wayback.github.io/web/20200615114013/www.bea.gov
    /news/2020/us-international-trade-goods-and-services-april-2020/ |
    shasum -a 256
10 01c8cd5606e22bb69799ba567cb41aae3d9873aea6215dba19acdbb0b56e221a  -
```

Fig. 51: cURL command to generate a SHA-256 hash value on only the HTML content. It produces the same hash value as there are no changes in the HTML of the archived page.

2. Generate a SHA-256 hash on the entity body of each resource. The set of resulting hash values will be aggregated and stored in the file *allhashes.txt* (lines 5-10).

3. Read the hash values from *allhashes.txt*, use them as input to the hashing command `shasum -a 256` which will generate one aggregated SHA-256 hash value that represent the content of the composite memento (line 12).

We used the shell script from Figure 52 to generate a single aggregated hash value on June 15, 2020 and June 17, 2020. As shown in Figure 53, the resulting aggregated hash values (marked in red) are different because of changes in the image *trad0420.png* (marked in red in Figures 50(b) and 50(c)). The resulting hash values of the image on June 15th and June 17th are marked in green in Figure 53. The example shown in Figure 53 indicates that it is important to include all resources comprising a composite memento in the hash calculation.

4.1.2 EXCLUDE ARCHIVE-SPECIFIC CONTENT

Including all resources embedded in a composite memento in the hash calculation should help identify changes over time. However, we need to identify embedded resources added by web archives. As explained in Chapter 1.5.1 and Chapter 2.2.5, web archives add archive-specific resources to the original content and also transform the original content to appropriately replay them in the browser. Including archive-specific content in the hash calculation

```
1 FILE=$(nice -n 20 wget --continue --unlink --page-requisites --
      timestamping -e robots=off -k --user-agent="Firefox 6.4" "$1" 2>&1 |
       egrep 'Saving to: .*' | sed -e 's/Saving to: âĂŸ//' | tr -d 'âĂŹ')
2
3 for TARGET in $FILE; do
4 if [ -f "$TARGET" ]; then
5 CONT=$(cat $TARGET)
6 HASH=$(echo "$CONT" | shasum -a 256 | awk '{print $1}')
7 echo "$HASH" >> "allhashes.txt"
8 echo "$HASH $TARGET"
9 fi
10 done
11
12 FINAL_HASH=$(cat "allhashes.txt" | shasum -a 256 | awk '{print $1}')
13
14 echo "Final hash: $FINAL_HASH"
```

Fig. 52: The shell script *aggregated_hash.sh* for generating a single hash on the content of a composite memento by aggregating all hash values of the embedded resources in a single temporary file and hashing the file. This shell script is a modified version of the original script written by Branwen [1].

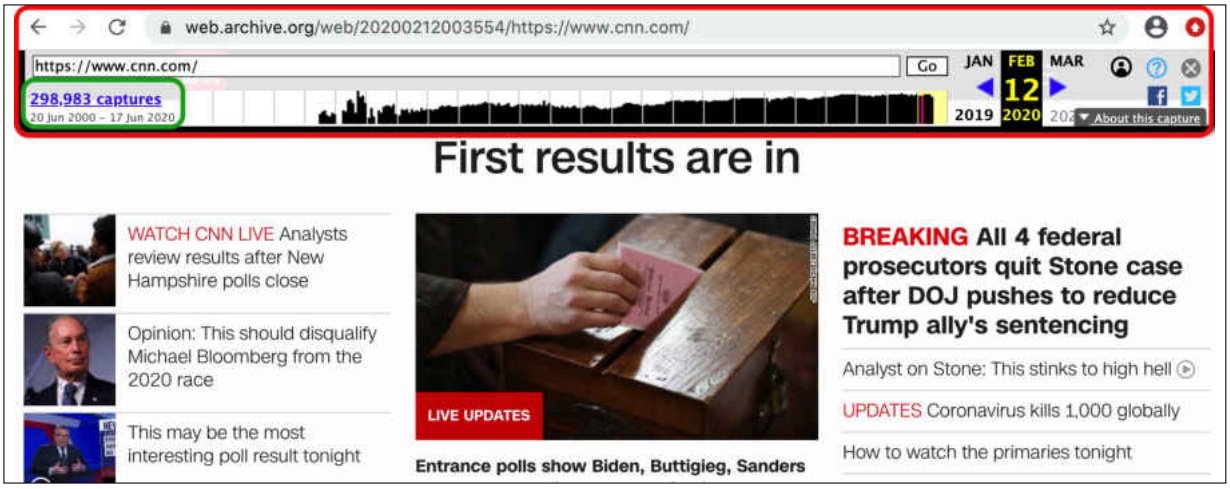
may prevent generating repeatable hash values. For example, Figure 54 shows the Internet Archive's banner (marked in red), which is used by the archive to convey information to users about the archived page. For instance, in Figure 54(a), the archive indicates that the displayed page is an archived page captured on February 12, 2020 from the original page `https://www.cnn.com`. The banner also shows the number of available mementos for the original page (marked in green) in the archive. As shown in Figure 54, we replayed the archived page at two different times on June 17, 2020 at 12:04 PM GMT and at 12:55 PM GMT. Within about 50 minutes, the number of available mementos in the archive for `https://www.cnn.com` has increased from 298,983 to 298,986. Thus, if we include such information (e.g., from the archive's banner) in the hash calculation, we may get different aggregated hash values.

There are various techniques that we can use to identify archive-specific resources:

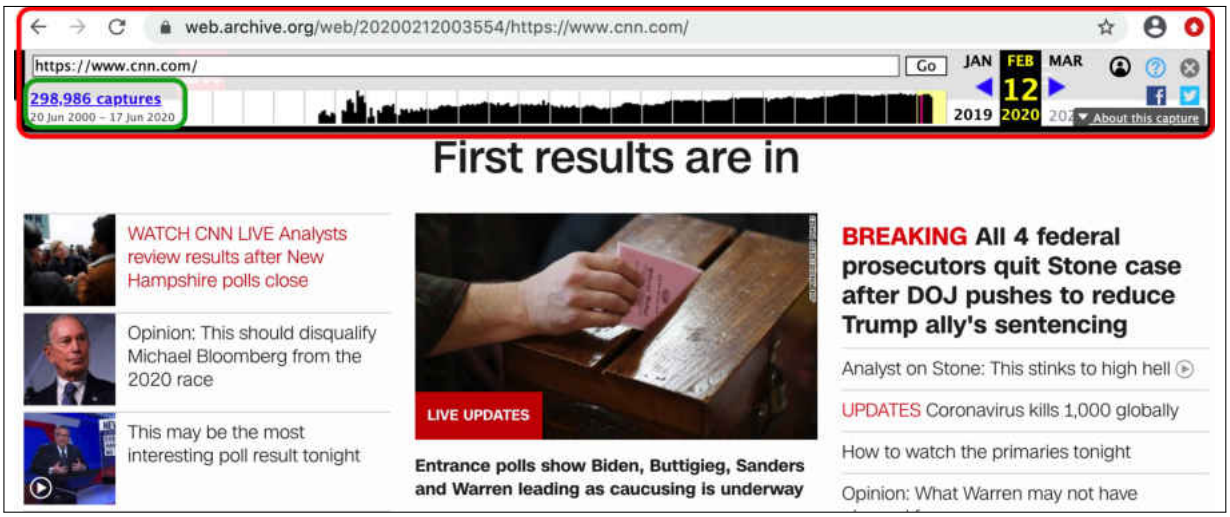
1. Archives that support the Memento protocol should respond with the HTTP `Link` response header containing `http://mementoweb.org/terms/donotnegotiate` and also `rel="type"` to requests of resources that are not mementos and are excluded from HTTP content negotiation based on the time dimension. For example, we should exclude the file *wayback-toolbar-logo.png* as it is not a memento, as shown in Figure 55.

```
$date
Mon Jun 15 17:21:44 EDT 2020
$ ./aggregated_hash.sh https://evil-wayback.github.io/web/20200615114013/www.bea.gov/news/2020/us-international-trade-goods-and-services-april-2020/
7a668dbe498d7046ba766693a0e561e63b4a5ceb97366acecd472e8e4671a8fe
4fd813dcad954ee96cc67e8dbcb673225cd81ded72714767f0bb2f7e1ba5fb9b
286a9ea032f2a85faffe8dc10d1ddf06ee00dc2f2ec2a84d48994c563eadef1c
27cedca55e7af115fd53d8f2a061ddb5f3cf322e897ea360f6c4e109f295063a
cc23b95896b7ee1cb65006420556be1d1d717a34ae5c86d0f10295b0d99de414
92c62d8f7b8b4ae8e367e43411c69bb21f10adc65da7d51f68e63f97178c1f57
3c5baccc073961fd80bed859da00ed951ae532b50dfa5275a445e9754b8252c2
2e37e31388b6e2412746cda0de1449bb17e7f29d21ba498673981e06b2b6ac9b
2e29ff9e101dda6c6504ca27def02b2da8d191f3249c27430f33092379e9dfb2
0871005c7ef65ea64f7a5026a40683193bd2bc716c87aa292de113465b738163
17722aac22890908db94b09f09ebe0f18730044afc5d0957eacc8cd4d81a9900
69d37ea3e3a74715e620a13e111eefba7446ed847f3c9fccfb39badec18cccac
26f2c5d869478020bc147bcc0526e927962f13f46a24259a13d0953286a5323b
f0de0d9117a65ed2e23361a45107dde2a41be855f4ffd03747fd8276b0421139
Final hash:
 a5139aaf4b84cf28fed50e06f5316fdf777f2c5bc202ad4165d665b7fd55d942

 $date
Wed Jun 17 19:03:45 EDT 2020
$ ./aggregated_hash.sh https://evil-wayback.github.io/web/20200615114013/www.bea.gov/news/2020/us-international-trade-goods-and-services-april-2020/
7a668dbe498d7046ba766693a0e561e63b4a5ceb97366acecd472e8e4671a8fe
4fd813dcad954ee96cc67e8dbcb673225cd81ded72714767f0bb2f7e1ba5fb9b
286a9ea032f2a85faffe8dc10d1ddf06ee00dc2f2ec2a84d48994c563eadef1c
27cedca55e7af115fd53d8f2a061ddb5f3cf322e897ea360f6c4e109f295063a
cc23b95896b7ee1cb65006420556be1d1d717a34ae5c86d0f10295b0d99de414
92c62d8f7b8b4ae8e367e43411c69bb21f10adc65da7d51f68e63f97178c1f57
3c5baccc073961fd80bed859da00ed951ae532b50dfa5275a445e9754b8252c2
2e37e31388b6e2412746cda0de1449bb17e7f29d21ba498673981e06b2b6ac9b
2e29ff9e101dda6c6504ca27def02b2da8d191f3249c27430f33092379e9dfb2
0871005c7ef65ea64f7a5026a40683193bd2bc716c87aa292de113465b738163
953d33b0afa80df6670bc2a1ad36e9227a62f61e12f27efcd1b17be624478653
69d37ea3e3a74715e620a13e111eefba7446ed847f3c9fccfb39badec18cccac
26f2c5d869478020bc147bcc0526e927962f13f46a24259a13d0953286a5323b
f0de0d9117a65ed2e23361a45107dde2a41be855f4ffd03747fd8276b0421139
Final hash:
 8f0b0807dde5bc89008ff39eb33b6e239627b0e0a1936b9db6ab27fadfcb3241
```

Fig. 53: Using the shell script *aggregated_hash.sh* (Figure 52) to generate an aggregated hash on a composite memento. The hash of the image *trad0420.png* is maked in green. The aggregated hash value is marked red. The resulting aggregated hash values are different because of changes in the image *trad0420.png*.



(a) Replaying the archived page on June 17, 2020 at 12:04 PM GMT.



(b) Replaying the archived page on June 17, 2020 at 12:55 PM GMT.

Fig. 54: Replaying an archived CNN page from the Internet Archive on June 17 at 12:04 PM GMT and at 12:55 PM GMT. The archive's banner shows that the number of available mementos in the archived (marked in green) has increased from 298,983 to 298,986. Thus, including archive-specific resources, such as the archive's banner, in hash calculation will affect the process of generating repeatable fixity information.

2. Archives that support the Memento protocol also include the HTTP response header Memento-Datetime for mementos. Those responses without this header are not mementos and are likely archive-specific resources. For example, as shown in Figure 55, there is no `Memento-Datetime` response header, so the image *wayback-toolbar-logo.png* is not a memento. The image is added by the archive as a part of the archive's banner.

```
curl -I https://www.webarchive.org.uk/wayback/archive/images/toolbar/
    wayback-toolbar-logo.png

HTTP/1.1 200 OK
Date: Mon, 06 Nov 2017 22:35:53 GMT
Server: Apache-Coyote/1.1
Link: <http://mementoweb.org/terms/donotnegotiate>; rel="type"
Accept-Ranges: bytes
ETag: W/"4549-1486118270000"
Last-Modified: Fri, 03 Feb 2017 10:37:50 GMT
Content-Type: image/png
Content-Length: 4549
Content-Language: en
```

Fig. 55: One way to identify archive-specific resources is to look at the HTTP Response header "Link" that contains `http://mementoweb.org/terms/donotnegotiate`.

3. The third technique for identifying archive-specific resources, especially for archives that do not support the Memento protocol, is by looking at the URI of a resource retrieved from the archive. Archives usually use a particular URI-M structure (described in Chapter 2.2.4). If the URI of a resource does not match the archive's URI-M structure, then the resource is not a memento, and it should be excluded from the hash calculation. For example, as explained in Chapter 2.2.4 (Figure 23), the URI-M structure used by the Internet Archive consists of the domain name, archival collection identifier, timestamp, and URI-R. For instance, the URI-M

   `https://web.archive.org/web/19961120150251/http://www.usnews.com:80/`

   is a URI to a memento, while the URI

   `https://web.archive.org/_static/images/toolbar/wayback-toolbar-logo.png`

   is not a URI to a memento (e.g., no timestamp included in the URI).

We want to avoid including archive-specific content in hash calculations for two reasons. First, as mentioned, this type of content does not belong to the original page. Second, the archive-specific resources, such as the Wayback Machine's banner, are expected to change over time due to, for example, updates in the archival replay system (e.g., the Wayback Machine software). In addition, archive-specific resources may carry dynamically-generated

information corresponding to the current state of an archive. Thus, we need to avoid including these archive-specific resources when calculating fixity information.

4.1.3 USE THE ORIGINAL HTTP ENTITY BODIES AND HEADERS, IF AVAILABLE

At replay time, web archives transform the content of an original web page by rewriting all links in the page (Chapter 2.2.5) so that all resources comprising the page (e.g., links to images) are retrieved from the archive, not from the live web. In addition to rewriting links, archives may add HTML comment tags to indicate that the page is being retrieved from the archive as shown in Figure 56. These HTML comment tags always contain current date and time resulting in an aggregated hash value that is different on each replay. Therefore, we should retrieve original content from web archives instead of the transformed and modified content.

```
<!--
     FILE ARCHIVED ON 06:01:32 Feb 12, 2020 AND RETRIEVED FROM THE
     INTERNET ARCHIVE ON 08:45:42 Jun 17, 2020.
     JAVASCRIPT APPENDED BY WAYBACK MACHINE, COPYRIGHT INTERNET ARCHIVE
         .

     ALL OTHER CONTENT MAY ALSO BE PROTECTED BY COPYRIGHT (17 U.S.C.
     SECTION 108(a)(3)).
-->
<!--
playback timings (ms):
  PetaboxLoader3.resolve: 66.907 (2)
  RedisCDXSource: 150.069 (7)
  esindex: 0.144 (7)
  LoadShardBlock: 1288.393 (24)
  PetaboxLoader3.datanode: 490.328 (25)
  exclusion.robots: 2.797 (7)
  load_resource: 180.865
  exclusion.robots.policy: 2.64 (7)
  CDXLines.iter: 90.396 (12)
-->
```

Fig. 56: The Internet Archives inserts HTML comment tags to convey information, such as the memento creation/retrieval date and time.

Most web archives allow users to access the original pages without rewriting their content (i.e., accessing raw mementos as described in Chapter 2.2.7). The most common mechanism to retrieve the raw mementos is by adding `id_` [89,90] after the timestamp in the requested URI-M. Retrieving and using raw mementos in generating fixity information will help us get

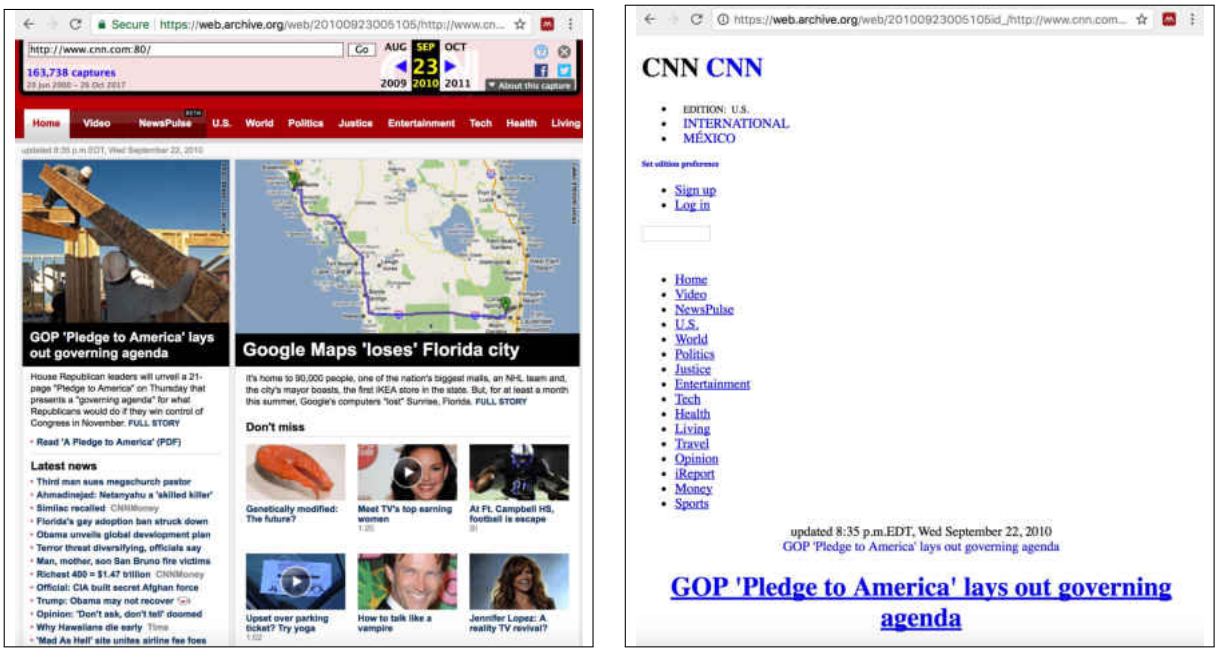
more consistent and repeatable hash values. For example, Figure 57 shows that generating fixity information on the base HTML file of the memento

`https://web.archive.org/web/19961120150251/http://www.usnews.com:80/`

results in the same hash value when `id_` is used to request the memento from the archive. Otherwise, we get a different hash value each time the memento is retrieved. Figure 58 shows an example of replaying the memento with and without the `id_` option. The raw memento is not replayed properly (Figure 58(b)) because all URIs in the base HTML file are not rewritten to be relative to the archive.

```
1 $ date
2 Wed Jun 17 05:33:08 EDT 2020
3 $ curl -s https://web.archive.org/web/20100923005105id_/http://www.cnn.
      com:80/ | shasum -a 256
4 166056cfe7c161d9669e75c8a6208e6ad705f3c330f96664ded3370fb85b8b15  -
5 $ date
6 Wed Jun 17 05:51:23 EDT 2020
7 $ curl -s https://web.archive.org/web/20100923005105id_/http://www.cnn.
      com:80/ | shasum -a 256
8 166056cfe7c161d9669e75c8a6208e6ad705f3c330f96664ded3370fb85b8b15  -
9 $ date
10 Wed Jun 17 06:01:32 EDT 2020
11 $ curl -s https://web.archive.org/web/20100923005105id_/http://www.cnn.
      com:80/ | shasum -a 256
12 166056cfe7c161d9669e75c8a6208e6ad705f3c330f96664ded3370fb85b8b15
13
14 $ date
15 Wed Jun 17 05:35:00 EDT 2020
16 $ curl -s https://web.archive.org/web/20100923005105/http://www.cnn.com
      :80/ | shasum -a 256
17 7fffc863900ce2389013cc856eb0bb08c8867fd71a755627e5160b58bb503fcf  -
18 $ date
19 Wed Jun 17 05:51:35 EDT 2020
20 $ curl -s https://web.archive.org/web/20100923005105/http://www.cnn.com
      :80/ | shasum -a 256
21 c7f8463570907308266934a518eff6cc3716aefa9a5400d9d6b46be94ca14a85  -
22 $ date
23 Wed Jun 17 06:01:42 EDT 2020
24 $ curl -s https://web.archive.org/web/20100923005105/http://www.cnn.com
      :80/ | shasum -a 256
25 63fcf404221d001e71cc8f830d38dd926bc6ee274daba5f62fa9c53370c6b533  -
```

Fig. 57: An example of considering a raw memento (downloaded using id_) in the hash calculation, which always results in the same hash value (the first three HTTP requests). In contrast, we get a different hash value each time the rewritten memento is used in the hash calculation (the last HTTP requests).



(a) The rewritten memento `https://web.archive.org/web/20100923005105/http://www.cnn.com:80/`.

(b) Requesting the raw memento using the `id_` option `https://web.archive.org/web/20100923005105id_/http://www.cnn.com:80/`.

Fig. 58: Rewritten mementos vs raw mementos.

4.1.4 IF RAW MEMENTOS ARE NOT AVAILABLE, WE MAY EXTRACT ORIGINAL CONTENT FROM REWRITTEN MEMENTOS

Some web archives, such as `webcitation.org`, do not serve raw mementos. In order to generate repeatable fixity information, we need to extract the original content by removing any code added by web archives. Jones et al. [91–93] explore how several web archives transform original content and introduce several rules for acquiring mementos. Web archives may provide mementos in multiple file formats. For example, `archive.is` does not allow accessing the raw content. However, it serves a composite memento packaged in a ZIP file. The content stored in the ZIP file is transformed by the archive, but the content that usually changes (e.g., the banner) is not included in the ZIP file. In December 2019, the `archive.is` had a major update in its service, including the use of a real web browser to capture web pages, instead of the PhantomJS headless browser [163]. As a result, the archive no longer provides the ZIP file for mementos created after December 1, 2019.

For some archives that do not provide access to raw mementos, it may be difficult, or

even impossible in some cases, to extract the content of the original page from a transformed/rewritten archived page. Archive.is, for example, transforms the original page by removing any JavaScript code and HTML attributes, and converting style sheets to inline properties [84], which is different from how the Wayback Machine transform the original content. Berlin [84] describes different techniques used by web archives to transform archived web pages.

In general, if we do not have access to raw mementos, we should exclude archive-specific content, that we can identify, from being included in the hash calculation.

4.1.5 VERIFY THAT ARCHIVES REPLY WITH THE ACTUAL ORIGINAL CONTENT IN RESPONSE TO REQUESTS FOR RAW MEMENTOS

Web archives may reply with the HTML `30x Redirect` status codes to requests of mementos for two reasons:

1. **Archival 30x Redirect**: At crawl time, web archives capture web resources with `30x Redirect` responses (i.e., the original resources reply with the HTTP `302 Found` or `301 Moved Permanently` status codes). Therefore, at replay time, archives should respond with the same original status code (i.e., `30x Redirect`) to requests of such archived resources.

2. **Non-archival 30x Redirect**: Archives may respond with `30x Redirect` to requests of mementos that are not available or cannot be served at the time of the request (i.e., the request redirects to the closest available memento).

In general, archives commonly return responses that include rewritten HTTP entity bodies and headers to requests for rewritten mementos regardless of the original HTTP status codes. For example, one of the archived pages of `http://www.carper.senate.gov/` (this resource returns multiple HTTP redirects in the live web as shown in Figure 59) is available at

`https://www.webharvest.gov/congress115th/20181221213254/http://www.carper.senate.gov/`

When requesting this memento, as expected, the archive replies with the rewritten HTTP entity body and headers so that all links are relative to the archive as shown in Figure 60. However, we notice different behavior by archives in response to requests for either

raw mementos whose original HTTP status codes are `30x redirect` or raw mementos that should return non-archival `HTTP 30x redirect`. We explain below how the three archives `vefsafn.is`, `webharvest.gov`, and `archive.org` react differently to requests for such raw mementos:

- **`vefsafn.is`**: The archive returns a custom HTML page with `200 OK` for requests for raw mementos. This occurs especially when the HTTP status code of the original response is `302 Redirect` (i.e., archival `302`). When the archive returns `200` for requests for raw mementos, it is an indication that the archive has successfully processed the request and returned the raw content, but this is not always the case as Figure 61(a) illustrates. The returned page by the Icelandic Web Archive `vefsafn.is` is not the raw version. It contains links pointing to the closest memento that satisfies the request. Such behavior might be applied by the archive to prevent redirects to the live web, but the rewritten content, most likely, affects the hash calculation, resulting in different hash values.

- **`webharvest.gov`**: The archive simply returns the HTTP status code of the original resource, which actually may be `30x Redirect` to the live web as Figure 61(b) (**webharvest.gov**) shows. Unlike `vefsafn.is`, the archive `webharvest.gov` does not use a custom HTML page to handle requests for raw mementos.

- **`archive.org`**: As shown in Figure 61 (**archive.org**), the archive returns `302 Redirect` with a rewritten HTML page (marked in blue in Figure 61(c)), which is different from how `vefsafn.is` handles the raw memento requests. Unlike `webharvest.gov`, `archive.org` does not respond with `302 Redirect` that redirects to the live web.

We do not consider any resources retrieved from the live web in hash calculation. Thus, the `302 Redirect`s to the live web by `webharvest.gov` do not affect generating repeatable hashes. Similarly, the technique that `archive.org` uses to respond to raw requests does not affect hash calculation because the archive returns a `302 Redirect` to the closest raw memento. Although this `302 Redirect` response unexpectedly has a rewritten HTML page, its content is fixed (i.e., the content is not expected to change over time). However, the custom HTML page returned by `vefsafn.is` might prevent generating repeatable hashes because the returned HTML page has content that is expected to change (e.g., the banner).

4.1.6 AVOID MEMENTOS SERVED FROM CACHE

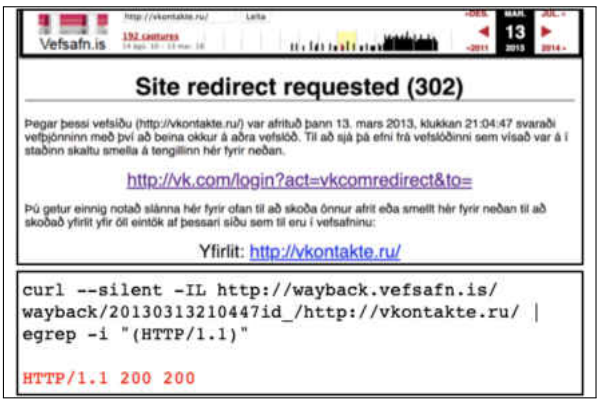
```
1 $ curl --silent -IL http://www.carper.senate.gov/ | egrep -i "(HTTP
    /1.1|Date:|^Location:)"
2
3 HTTP/1.1 301 Moved Permanently
4 Location: https://www.carper.senate.gov/
5 Date: Fri, 03 Jul 2020 10:24:38 GMT
6 HTTP/1.1 302 Moved Temporarily
7 Location: http://www.carper.senate.gov/public/
8 Date: Fri, 03 Jul 2020 10:24:38 GMT
9 HTTP/1.1 301 Moved Permanently
10 Location: https://www.carper.senate.gov/public/
11 Date: Fri, 03 Jul 2020 10:24:38 GMT
12 HTTP/1.1 200 OK
13 Date: Fri, 03 Jul 2020 10:24:38 GMT
```

Fig. 59: The HTTP request to the live web page `http://www.carper.senate.gov/` resulted in multiple HTTP redirects.

```
1 $ curl --silent -IL https://www.webharvest.gov/congress115th
    /20181221213254/http://www.carper.senate.gov/ | egrep -i "(HTTP
    /1.1|^Location:)"
2
3 HTTP/1.1 302 Found
4 Location: /congress115th/20181221213254/http://www.carper.senate.gov/
    public/
5 HTTP/1.1 302 Found
6 Location: /congress115th/20181221213257/http://www.carper.senate.gov/
    public/
7 HTTP/1.1 301 Moved Permanently
8 Location: /congress115th/20181221213257/https://www.carper.senate.gov/
    public/
9 HTTP/1.1 302 Found
10 Location: /congress115th/20181221213240/https://www.carper.senate.gov/
    public/
11 HTTP/1.1 200 OK
```

Fig. 60: The HTTP request to the archived web page `https://www.webharvest.gov/congress115th/20181221213254/http://www.carper.senate.gov/`.

Web archives may use web caching for performance purposes to speed up subsequent requests. For example, the Wayback Machine's HTTP Response header `X-Page-Cache` indicates whether the returned content is delivered from cache (`X-Page-Cache: HIT`) or not (`X-Page-Cache: MISS`). Although caching has powerful benefits, the returned content may not reflect what is actually in the archive. Figure 62 shows content that is not served



(a) `vefsafn.is`.

(b) `webharvest.gov`.

(c) `archive.org`.

Fig. 61: Three archives react differently to requests for raw mementos. The archive `vesafn.is` returns a custom HTML page with `200 OK` which might cause different hashes. The archive `webharvest.gov` issues `302 Redirect` to the live web, while `archive.org` returns `302 Redirect` (with a rewritten "unexpected to change" HTML page—marked in blue) to the closest raw memento that satisfies the request. The way that `webharvest.gov` and `archive.org` react to requests for raw mementos does not affect the hash calculation until they install a new version of Wayback and configuration changes.

from the cache (i.e., `X-Page-Cache: MISS`). The issue is that cache hits produce a risk of calculating the same hash even if the archived page has changed. For example, we issued different HTTP requests for the same memento (Figure 63). The first response was actually not from the cache with computed MD5 hash ending in `7afd3`, while the two responses that follow were from the cache. The MD5 hash value calculated on the content of each of the two responses were identical to the first hash value, because the content served from the cache is an exact copy of the content returned upon the first request. Now, because the content on the cache is only stored for a "short" period of time (depending on how the caching system is configured) before it is discarded (or updated), the fourth response was

not a cache hit. The archived page seems to be changed since we obtain a different MD5 hash value ending in `b1059`. Thus, we should not consider content served from the cache when computing a memento's fixity information.

```
curl -i http://web.archive.org/web/20130724144801/http://www.cnn.com/

HTTP/1.1 200 OK
Server: Tengine/2.1.0
Date: Mon, 02 Oct 2017 11:10:15 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 147311
Connection: keep-alive
X-Archive-Orig-expires: Wed, 24 Jul 2013 14:48:55 GMT
X-Archive-Orig-vary: Accept-Encoding
X-Archive-Orig-last-modified: Wed, 24 Jul 2013 14:47:16 GMT
X-Archive-Orig-cache-control: max-age=60, private
X-Archive-Orig-date: Wed, 24 Jul 2013 14:46:36 GMT
Memento-Datetime: Wed, 24 Jul 2013 14:48:01 GMT
Link: <http://www.cnn.com/>; rel="original", <http://web.archi
ve.org/web/timemap/link/http://www.cnn.com/>; rel="timemap"; ...
X-App-Server: wwwb-app23
X-ts: ----
X-Archive-Playback: 0
X-location: All
X-Page-Cache: MISS

...
```

Fig. 62: The memento is not served from the cache as the HTTP Response header `X-Page-Cache:MISS` indicates.

4.1.7 EXCLUDE RESOURCES WITH INCORRECT HTTP STATUS CODES

Web archives may respond with inaccurate HTTP status codes, which affects generating repeatable fixity information. For example, for resources that are not found in the archive (e.g., non-archival `404`), the archive should respond with the HTTP `404` status code. However, we have encountered cases in which the archive responds with `200 OK` (instead of `404`). In this case, the archive replies with `soft 404`—responding with `200 OK` status code and an HTTP entity indicating that the resource was not found in the archive [164]. For instance, the request for the URI-M

`http://www.collectionscanada.gc.ca/webarchives/20061026030606/http://www.regulations.gov/fdmspublic/component/main`

```
% date
Mon Oct  2 01:15:18 EDT 2017
% curl --silent http://web.archive.org/web/20130724144801/http://www.
cnn.com/ | md5
477b6d923cbb7bf9675a0d2feb37afd3

% date
Mon Oct  2 01:16:29 EDT 2017
% curl --silent http://web.archive.org/web/20130724144801/http://www.
cnn.com/ | md5
477b6d923cbb7bf9675a0d2feb37afd3

% date
Mon Oct  2 01:19:31 EDT 2017
% curl --silent http://web.archive.org/web/20130724144801/http://www.
cnn.com/ | md5
477b6d923cbb7bf9675a0d2feb37afd3

% date
Mon Oct  2 02:10:24 EDT 2017
% curl --silent http://web.archive.org/web/20130724144801/http://www
.cnn.com/ | md5
dda6a9bf091d412cbdc2226ce3eb1059
```

Fig. 63: The first `cURL` request was not served from the cache (i.e., `X-Page-Cache: MISS`) while the second and third request were cache HITs. After about an hour, the fourth request was a cache MISS and produces a different hash. This example shows that cache HITs produce the same hash even though the memento might have changed.

returned a `200` status code as shown in Figures 64 and 65 with an entity body indicating the resource is not available in the archive. Most web archives, such the Internet Archive, do not use or return `soft 404` unless it is how the original server responds at crawl time. In general, if we are able to detect responses with `soft 4xx/5xx`, they should not be part of the hash calculation.

4.1.8 INCLUDE SELECTED HTTP RESPONSE HEADERS IN HASH CALCULATION

We should include values of important HTTP response headers in hash computation. For instance, by hashing the HTTP header `Location`, we can identify if mementos are served from different locations, and we want to keep track of such behavior. Another important header is `Content-Type` through which we can identify if the format of the resource has changed. For example, an archive may start serving a video in the MPEG-4 format [165]

```
WARC/1.0
WARC-Type: response
WARC-Target-URI: http://www.collectionscanada.gc.ca/webarchives
    /20061026030606/http://www.regulations.gov/fdmspublic/component/main
WARC-Date: 2017-12-28T01:13:26Z
WARC-Record-ID: <urn:uuid:4e530b30-eb6c-11e7-8926-079a54998c73>
Content-Type: application/http; msgtype=response
Content-Length: 11037

HTTP/1.1 200 OK
Date: Thu, 28 Dec 2017 01:13:15 GMT
Server: Apache-Coyote/1.1
ETag: "pv4e2543f6d7ff101f765de8e5085241b1"
Vary: Accept-Encoding
Content-Language: en-US
Content-Type: text/html;charset=UTF-8
Content-Length: 10604

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>
...
<p>
<strong>URL:</strong> <span class="nowrap">http://www.regulations.gov/
    fdmspublic/component/main</span><br /><br /></p>
<!--<h2></h2>-->
<p><strong>Sorry, no documents with the given url were found in this
    archive.</strong></p>
...
</body>
</html>
```

Fig. 64: Our WARC records show that the Canadian archive replies with `soft 404`—returning `200 Ok` with an entity body indicating that the resource is not available in the archive.

that was previously served using a different format, such as MPEG-2 [166]. Rosenthal et al. [167] implemented a proof-of-concept system that demonstrated how an archive can use HTTP content negotiation to transparently migrate resources from one MIME type to another (e.g., `image/gif` to `image/png`). This can be useful if a format type becomes obsolete (as recently happened with Adobe Flash [168]) or otherwise legally encumbered (as happened with GIF [169]). Other HTTP response headers that should be included in hash calculation are the original headers that begin with `X-Archive-orig-`.

Fig. 65: The Canadian archive responds with `soft 404` to the request of a memento that is not available in the archive (i.e., returning `200 OK` with an HTTP entity indicating that the resource is not in the archive).

4.1.9 EXCLUDE ANY RESOURCES FROM THE LIVE WEB

At replay time, a web archive rewrites all URIs found in an original web page so they point to the archive, not to the live web. However, there are URIs, generated by client-side JavaScript code, that will not be rewritten by the archive. Therefore, the web resources specified by such URIs will be retrieved from the live web. Because live web resources often change or disappear over time, we do not want to include such resources when computing fixity information.

Brunelle et al. [5] introduced multiple examples of mementos that when replayed in the browser have embedded resources retrieved from the live web, not from the archive. For example, the memento

`http://web.archive.org/web/20080903204222/http://www.cnn.com/`

is from 2008. When replayed in 2012, the memento included an embedded resource (loaded by JavaScript) that was retrieved from the live web, which was about the 2012 presidential election between Barack Obama and Mitt Romney (Figure 66), not about the 2008 presidential election between Barack Obama and John McCain.

The Internet Archive uses the HTTP header `Content-Security-Policy` (explained in

Chapter 3.1) to avoid loading live web resources linked from archives. However, the issue may still occur in the Internet Archive and other archives.

Fig. 66: An archived CNN web page from 2008. When it is replayed in 2012, the advertisement image was about the 2012 presidential election, not about the 2008 presidential election (from [5]).

4.1.10 FIXITY INFORMATION CANNOT BE COMPUTED FOR MEMENTOS WITH TRANSIENT ERRORS

Before generating fixity information on a composite mementos, we should verify that the base HTML file and all resources embedded in the composite memento are retrieved without any transient errors. A transient error is a temporary error that is expected to disappear after a short period of time. The HTTP `500 Server Error` is an example of a transient error. It indicates that the server (or the archive) has experienced an unexpected condition preventing it from completing the request. Figures 67 and 68 show an example of a transient error where the memento

`https://webharvest.gov/congress111th/20110111060640/http://house.gov/`

is replayed/downloaded 20 times. There was a transient error in download 11 since the archive replied with `HTTP 503 Service Unavailable`. We do not see this transient error in the other downloads.

Fig. 67: An example of a transient error. Replaying a memento 20 times. The archive replies with `HTTP 503 Service Unavailable` at download 11.

```
1  WARC/1.0
2  WARC-Type: response
3  WARC-Target-URI: https://www.webharvest.gov/congress111th/2011011106064
4  0/http://house.gov//
5  WARC-Date: 2018-01-31T00:27:29Z
6  WARC-Record-ID: <urn:uuid:856f9220-061d-11e8-b159-57a39c6ec8eb>
7  Content-Type: application/http; msgtype=response
8  Content-Length: 307
9
10 HTTP/1.1 503 Service Unavailable
11 Date: Wed, 31 Jan 2018 00:27:19 GMT
```

Fig. 68: A transient error example of `HTTP 503 Service Unavailable` returned by the archive on January 31, 2018.

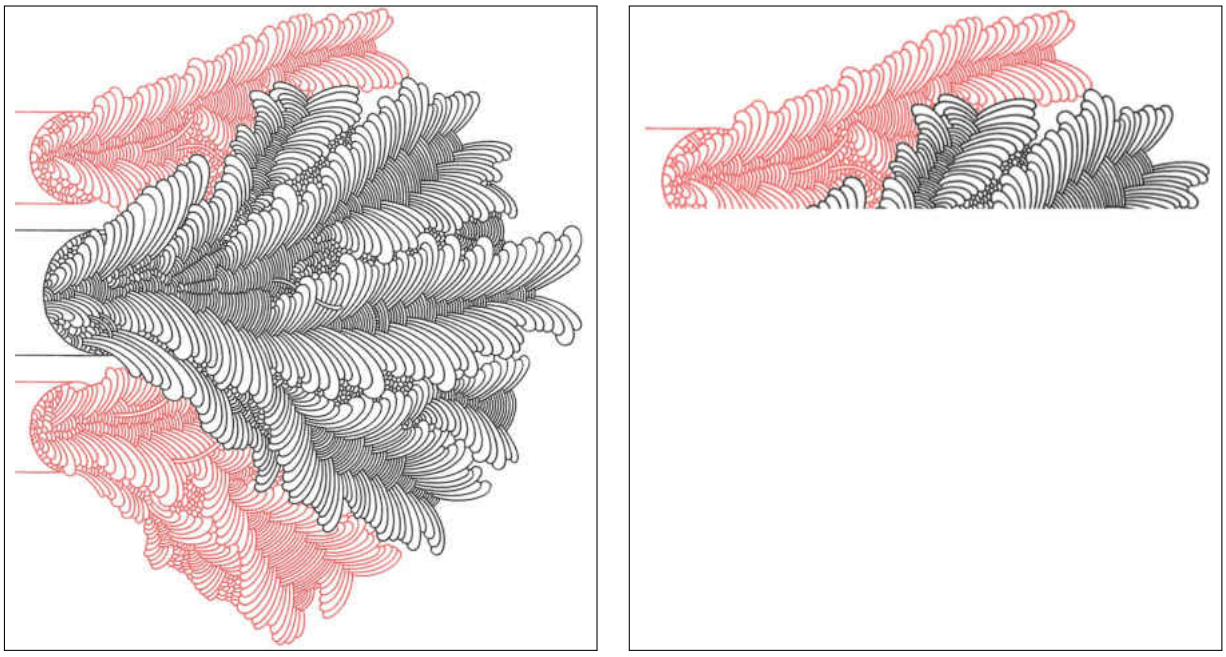
Transients errors may affect either the base HTML file (i.e., the root URI-M) as shown in Figure 67 or any resource embedded in the composite memento. For example, the image

`http://webarchive.nationalarchives.gov.uk/20170303010736id_/https://cereals.ahdb.org.uk/media/1157842/corporate-strategy-1.jpg`

which is embedded in the memento

`http://webarchive.nationalarchives.gov.uk/20170302155612/http://cereals.ahdb.org.uk/`

was requested at two different times. The HTTP entity of the image was successfully delivered on December 01, 2017 (Figure 69(a)). However, because of a transient error, only part of the HTTP entity body was delivered on December 07, 2017 (Figure 69(b)). We can identify this type of transient error by comparing the `Content-Length` response header of the resource with the actual size of the HTTP entity body. If the actual size of the entity is smaller than the value of the `Content-Length` response header, then the entity body is not successfully delivered. For instance, Figure 70 shows the actual size (459,640 bytes) of the HTTP entity body and headers of the image illustrated in Figure 69(b), which is smaller than the actual size of the entity body (642,336 bytes).



(a) The image was successfully delivered on December 01, 2017.

(b) The image was not successfully delivered on December 07, 2017.

Fig. 69: The image is downloaded two time. Because of an transient error, only part of the HTTP entity body of the image was delivered on December 07, 2017.

```
WARC/1.0
WARC-Type: response
WARC-Record-ID: <urn:uuid:fd24929e-d651-11e7-8f56-e839354e74f7>
WARC-Target-URI: http://webarchive.nationalarchives.gov.uk
    /20170303010736id_/https://cereals.ahdb.org.uk/media/1157842/
    corporate-strategy-1.jpg
WARC-Date: 2017-12-01T04:42:08Z
Content-Type: application/http; msgtype=response
Content-Length: 643398

HTTP/1.0 200
Content-Type: image/jpeg
Content-Length: 642336
Date: Fri, 01 Dec 2017 04:42:07 GMT

================================================================================

WARC/1.0
WARC-Type: response
WARC-Record-ID: <urn:uuid:fc89d63d-db35-11e7-9c4a-e839354e74f7>
WARC-Target-URI: http://webarchive.nationalarchives.gov.uk
    /20170303010736id_/https://cereals.ahdb.org.uk/media/1157842/
    corporate-strategy-1.jpg
WARC-Date: 2017-12-07T10:04:18Z
Content-Type: application/http; msgtype=response
Content-Length: 459640

HTTP/1.0 200
Content-Type: image/jpeg
Content-Length: 642336
Date: Thu, 07 Dec 2017 10:04:18 GMT
```

Fig. 70: A transient error example of an incomplete entity body. We can identify this type of error by comparing the value of `Content-Length` with the actual size of the entity body. In this example, the entity body is incomplete on December 07, 2017 because the actual size of the entity (459,640 bytes) is smaller than the value of `Content-Length` (642,336 bytes).

Transient errors also include situations where there is no response from the archive to HTTP requests for mementos. For example, Figures 71 shows that there is a "timeout" error to the request of a memento from the Canadian archive `webarchive.bac-lac.gc.ca` on March 26, 2020, and Figure 72 shows that the HTTP request to the homepage of the WebCite archive `www.webcitation.org` resulted in the transient error "curl: (6) Could not resolve host". They are transient errors because they disappeared after a short time.

```
1 $ date
2 Thu Mar 26 02:31:40 EDT 2020
3 $ curl -I http://webarchive.bac-lac.gc.ca:8080/wayback/20060127185447/
      http://www.boaa.gc.ca/
4 curl: (7) Failed to connect to webarchive.bac-lac.gc.ca port 8080:
      Operation timed out
```

Fig. 71: A "timeout" transient error example.

```
1 $ date
2 Mon Jul 01 01:33:15 EDT 2019
3 $ curl -I www.webcitation.org
4 curl: (6) Could not resolve host: www.webcitation.org
```

Fig. 72: A transient error example showing that the host does not resolve.

4.1.11 ARCHIVED WEB PAGES MAY BE IN FLUX

On the playback of a composite memento, there may be one of more embedded resources that have not been captured by the archive (e.g., `404 Not Found`). Some web archives simply return the HTTP `404 Not Found` to requests for these resources without any further actions. However, other web archives behave differently by trying to capture these resources from the live web, but they will not make these resources available before the next request of the composite memento. Therefore, replaying mementos at least once before computing fixity information can trigger services in the archives to capture embedded resources that have not been archived. For example, the image

`https://web.archive.org/web/20141209193553im_/http://wac.450F.edgecastcdn.net/80450F/noisecreep.com/files/2009/06/aaron_a042209eb_200.jpg`

which is embedded in the memento

`https://web.archive.org/web/20141209193553/http://noisecreep.com/aaron-harris-of-isis-talks-twitter/`

returns `404 Not Found` on November 17, 2017, and it becomes `200 OK` on November 18, 2017, as shown in Figure 73. We notice that the creation date (`Memento-Datetime`) of the image is November 18, 2017, which indicates that archives may change during our observations.

4.2 ADDITIONAL GUIDELINES FOR GENERATING FIXITY INFORMATION

```
WARC/1.0
WARC-Type: response
WARC-Target-URI: https://web.archive.org/web/20141209193553im_/http://
    wac.450F.edgecastcdn.net/80450F/noisecreep.com/files/2009/06/
    aaron_a042209eb_200.jpg
WARC-Date: 2017-11-17T00:34:17Z
WARC-Record-ID: <urn:uuid:0bc5bfe0-cb2f-11e7-8c10-b775c1a97b16>
Content-Type: application/http; msgtype=response
Content-Length: 241

HTTP/1.1 404 NOT FOUND
X-ts: ----
Server: Tengine/2.1.0
X-App-Server: wwwb-app23
Date: Fri, 17 Nov 2017 00:33:54 GMT
Content-Type: text/html; charset=utf-8

================================================================================

WARC/1.0
WARC-Type: response
WARC-Target-URI: https://web.archive.org/web/20141209193553im_/http://
    wac.450F.edgecastcdn.net/80450F/noisecreep.com/files/2009/06/
    aaron_a042209eb_200.jpg
WARC-Date: 2017-11-18T10:33:14Z
Content-Length: 406

HTTP/1.1 302 FOUND
Date: Sat, 18 Nov 2017 10:32:49 GMT
Location: https://web.archive.org/save/_embed/http://wac.450F.
    edgecastcdn.net/80450F/noisecreep.com/files/2009/06/
    aaron_a042209eb_200.jpg
âĂę
WARC/1.0
WARC-Type: response
WARC-Target-URI: https://web.archive.org/save/_embed/http://wac.450F.
    edgecastcdn.net/80450F/noisecreep.com/files/2009/06/
    aaron_a042209eb_200.jpg
WARC-Date: 2017-11-18T10:33:14Z
âĂę
HTTP/1.1 200 OK
Date: Sat, 18 Nov 2017 10:32:51 GMT
Content-Type: image/jpeg
Content-Location: /web/20171118103250/http://wac.450F.edgecastcdn.net
    /80450F/noisecreep.com/files/2009/06/aaron_a042209eb_200.jpg
```

Fig. 73: Replaying a composite memento on November 17, 2017. One of the embedded images returns `404 Not Found` on November 17, 2017 before it becomes `200 OK` the next day.

Based on these initial guidelines for generating fixity information on the playback of mementos, we conducted a study (Chapter 6) to determine if these guidelines can produce repeatable fixity information. We used 16,627 composite mementos selected from 17 public web archives. We downloaded each composite memento 39 times within 14 months. After each download of a composite memento, we generated an aggregated hash value that represented content of the composite memento. We expected the resulting 39 aggregated hash values of each composite memento to be identical. However, the result of our study indicates that 88.45% of mementos produced multiple unique aggregated hash values even after considering the initial guidelines.

One of the main reasons for getting different aggregated hash values on the same composite memento over time is JavaScript. When JavaScript code is executed (e.g., in a web browser), it may load additional resources in the composite memento. These resources may be selected randomly by JavaScript, which would result in different aggregated hash values. For example, as shown in Figure 74, each time the memento

`https://wayback.archive-it.org/all/20190102192824/https://www.rochester.edu/`

is replayed, a different background image is loaded by JavaScript. Figure 75 shows the corresponding JavaScript code. The function `Math.random()` is used to select an image randomly from a pool of nine images.

The other reason for getting different aggregated hash values on the same composite memento over time is changes in TimeMaps as explained in Chapter 2.3.3. Web archives may not be able to serve some mementos because, for example, their index files are temporarily unavailable. Therefore, web archives may respond at different times with different TimeMaps.

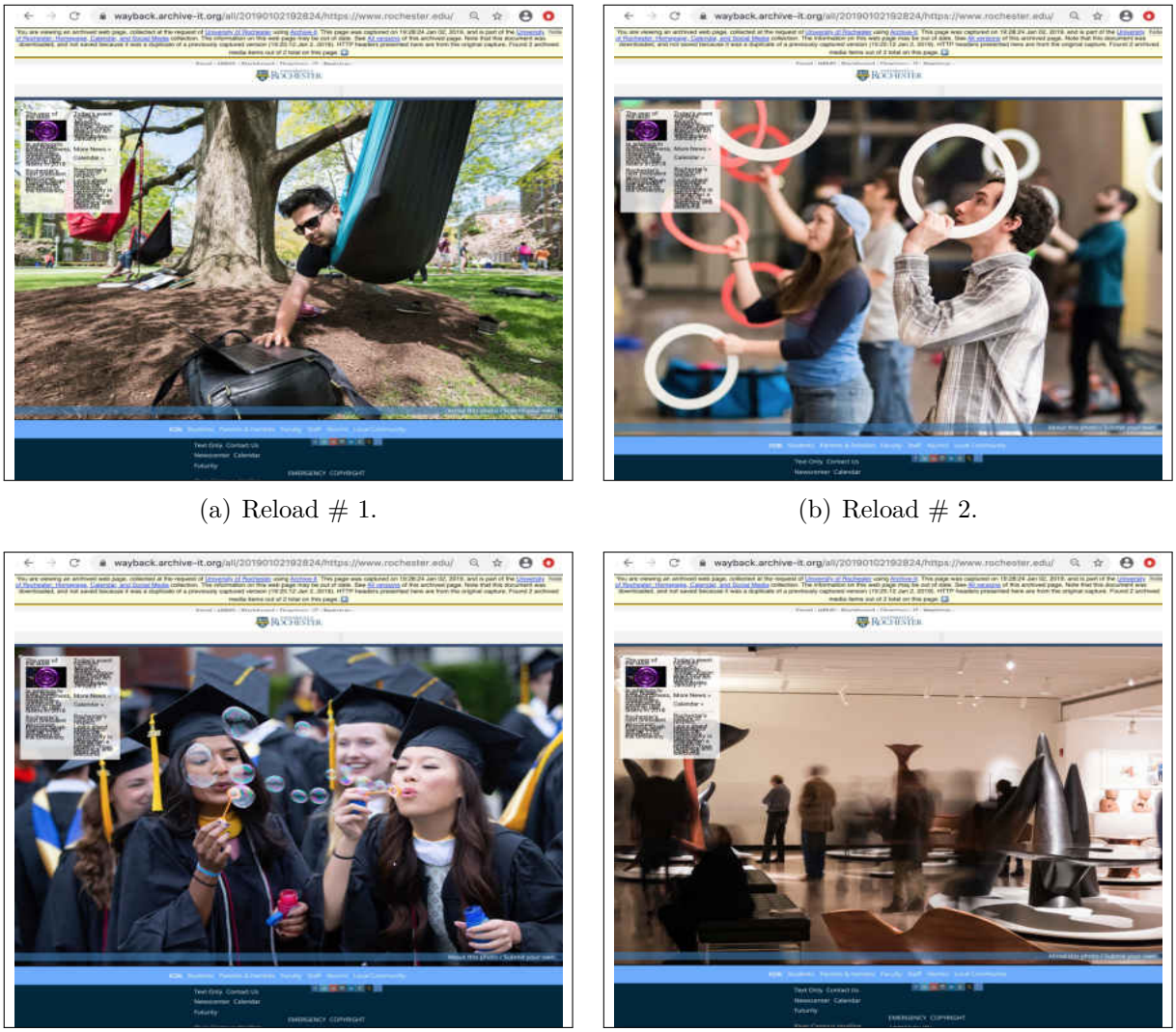
For example, the JavaScript file

`https://web.archive.org/web/20120306231221/http://an.yandex.ru/system/context.js`

which is embedded in the memento

`https://web.archive.org/web/20120306231221/http://www.echo.msk.ru:80/blog/milov/865094-echo`

is requested on January 31, 2018. The redirect points to



(a) Reload # 1.

(b) Reload # 2.

(c) Reload # 3.

(d) Reload # 4.

Fig. 74: Each time the memento `http://wayback.archive-it.org/all/20130102002028/http://www.cornell.edu/` is replayed, JavaScript loads a different background image.

`https://web.archive.org/web/20120308004741/http://an.yandex.ru/system/context.js`

before it redirects to

`https://web.archive.org/web/20120305025731/http://an.yandex.ru/system/context.js`

which returns `200 OK`. When requesting the same JavaScript file

```
var random_images_array = [
            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/seven.jpg" .../div>',

            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/eleven.jpg" .../div>',

            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/hero_three2x.jpg" .../div>',

            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/homepage-quad.jpg" .../div>',

            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/homepage-student-smiling.jpg"
               .../div>',

            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/homepage-student-lab.jpg" .../
               div>',

            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/homepage-research-faculty.jpg"
               .../div>',

            '<div class="hero-site yellow"><div class="hero-image"><img
                src="assets/hero_images/homepage-eastman.jpg" .../div
               >',
        ];
function getRandomImage(imgAr) {
            var num = Math.floor( Math.random() * imgAr.length );
            var img = imgAr[ num ];
            var imgStr = img;console.log(imgStr);
            document.write(imgStr); document.close();
        }
getRandomImage(random_images_array, '/images/');
```

Fig. 75: Because of the function `Math.random()`, each time the JavaScript code is executed, an image will be selected randomly (out of 9 images).

`https://web.archive.org/web/20120306231221/http://an.yandex.ru/system/context.js`

on February 06, 2018, it redirects to

`https://web.archive.org/web/20120308004741/http://an.yandex.ru/system/context.js`

which returns `200 OK`. Therefore, the same entity is served from two different URI-Ms over time.

Although the inconsistency in the returned TimeMaps may be unacceptable, it is beneficial as it makes web archives respond quickly to memento requests.

As a result of our study, we introduce additional guidelines:

- Because of the effect of JavaScript and changes in TimeMaps, there can be more than one correct aggregated hash value calculated on the playback of a composite memento over time.
- When generating fixity information on the playback of a composite memento, we can generate multiple aggregated hash values that are calculated using different hashing techniques:
    1. URI-M-based hashing (Chapter 6.7.1): Only the URI-Ms of resources embedded in a composite memento are included in the hash calculation.
    2. Entity-based hashing (Chapter 6.7.2): Only the HTTP entity bodies of resources embedded in a composite memento are included in the hash calculation.
    3. Complete hashing (Chapter 6.1): The HTTP entity bodies and headers, URI-Ms, and the HTTP status codes of resources embedded in a composite memento are included in the hash calculation.
- The fixity information generated on a composite memento should include not only the multiple aggregated hash values, but also fixity information generated on each resource embedded in the composite memento.

We introduce an archive-aware hashing function (Figure 76) that consists of multiple steps:

- It follows the guidelines defined in Sections 4.1 and 4.2 to generate fixity information on the playback of archived web pages. The archive-aware hashing function should produce multiple aggregated hash values on the playback of a composite memento using multiple hashing techniques (Chapter 6.7.2). It should also produce fixity information for each resource comprising the composite memento.
- The resulting aggregated hash values should be stored in a specific file, or manifest, in any standard serialization format (e.g., JSON) along with other metadata about

the memento. This file then is published at a well-known web location (e.g., at the Archival Fixity server in Figure 76) before it is pushed into multiple web archives (Chapter 7).

- The manifest file then is disseminated to web archives using one of the two approaches, *Atomic* and *Block* (Chapter 7).

In general, we want web archives to monitor web archives where a memento is preserved in a particular archive (e.g., `archive.org`) and its fixity information are preserved by other web archives, such as `archive.is` and `perma.cc`. The evaluation of our framework is described in Chapter 7.

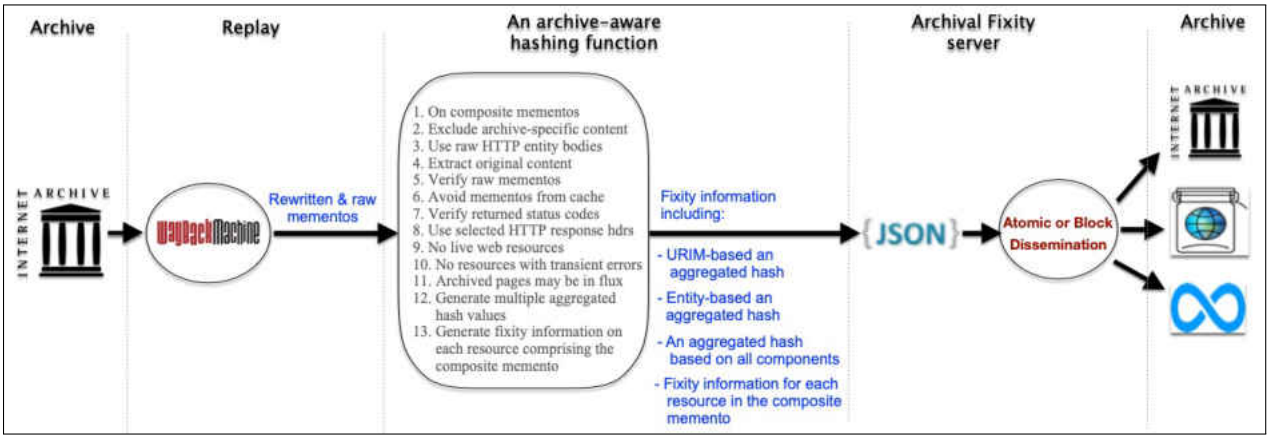


Fig. 76: The archive-aware hashing function.

4.3 CHAPTER SUMMARY

This chapter describes our archive-aware hashing function for generating fixity information on the playback of mementos. We start by introducing several initial guidelines that our archive-aware hashing function should follow for generating fixity information. We also provide examples and explanation of reasons that prevent generating repeatable fixity information including the impact of transient errors, caching servers, archive-specific resources, live web leakage, the rewriting process at replay, reconstructing composite mementos, changes in HTTP response headers and entities, and handling raw mementos. Next, we introduce additional guidelines based on results from our 14-month study on 16,627 composite mementos. We introduce these additional guidelines because of the effect of JavaScript and changes in TimeMaps on producing repeatable fixity information.

CHAPTER 5

COLLECTING A LARGE DATASET OF MEMENTOS

Even though web archives hold billions of archived web pages [69], obtaining a sample of mementos can be difficult. This chapter describes four methods used to create a dataset of 16,627 URI-Ms of 3,698 URI-Rs from 17 public web archives [170]. We discovered many more mementos, but we explain why we applied filters and set conditions to reduce the number of selected mementos. We use this collection in our study of identifying changes in the replay of mementos over time (Chapters 6 and 7)

To obtain a memento, lookup by URI-R is widely supported by most web archives, but this requires a user to know the URI of an original page. Table 2 shows a list of 17 public web archives which we classify in the following categories:

- **General**: Archives preserve any web page discovered through large-scale web crawlers.
- **On-demand**: In general, only web pages (URI-Rs) submitted by users are captured, but the archive might also create archived collections or obtain a copy of collections captured by other archives.
- **Subscription-based**: Each subscriber makes a separate collection of URI-Rs (i.e., seed URI-Rs) and can indicate how frequently these URI-Rs should be crawled in addition to other options. The archive usually crawl more pages by following links found in the already crawled web pages identified by the seed URI-Rs.
- **National**: Archives preserve a government or country's web content. They might capture web pages with one or more specific Top-Level Domains (TLDs).
- **Organizational**: Archives preserve web pages that are about specific organizations, such as the European Union. This category also includes archives maintained by universities and state governments to preserve their own websites.

Table 3 shows the actual number of collected mementos (URI-Ms) per archive and also the distribution of selected mementos through time. We explain in the rest of this section how we obtained this set of 16,627 mementos.

During the process of collecting mementos, we obtained many archived pages, but because of the following requirements for our target study, we only selected 16,627 mementos:

TABLE 2: A set of 17 public web archives.

Archive URI	Archive Name	Purpose
swap.stanford.edu	Stanford Web Archive Portal	General
web.archive.org	The Internet Archive	General and on-demand
archive.bibalex.org	Bibliotheca Alexandrina's Internet Archive	National
arquivo.pt	The Portuguese Web Archive	National
collectionscanada.gc.ca	Library and Archives Canada	National
digar.ee	The Estonian Web Archive	National
nationalarchives.gov.uk	The National Archives	National
vefsafn.is	The Icelandic Web Archive	National
webarchive.loc.gov	Library of Congress Web Archives	National
webarchive.org.uk	The UK Web Archive (UKWA)	National
webarchive.proni.gov.uk	Public Record Office of Northern Ireland (PRONI)	National
webharvest.gov	Congressional & Federal Government Web Harvests	National
archive-it.org	Archive-It - Web Archiving Services for Libraries and Archives	Subscription-based
archive.is	Archive.is	On-demand
perma.cc	Perma.cc	On-demand
webcitation.org	WebCite	On-demand
europarchive.org	The European Archive	Organizational

1. Downloading mementos is a slow operation and since the bottleneck is the archives themselves, parallelization will not help. We chose a target of completing the download of all mementos from all the archives within 40 hours. We also planned to do no more than two such downloads per week in order to limit the load on the archives.

2. Since we want to study changes in the playback of mementos over time, we chose 200 as the minimum number of URI-Rs per archive.

3. The number of selected mementos from each web archive should not exceed 1,600. This condition should help reducing the difference between large archives and small archives in terms of the number of sampled mementos.

5.1 COLLECTING THE INITIAL SET OF URI-RS

TABLE 3: URI-Ms per archive per year. The data is available in CSV format at `github.com/oduwsdl/mementos-fixity/blob/master/urims-per-year.csv`.

Archive	URI-Ms	1996	97	98	99	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
webarchive.loc.gov	1,594	-	1	1	1	4	100	100	100	99	100	100	100	100	98	99	99	99	98	98	99	98	-
vefsafn.is	1,589	6	8	10	11	11	13	13	14	42	46	74	71	70	85	102	116	140	153	152	152	150	150
webcitation.org	1,585	-	-	-	-	-	-	-	-	-	28	89	85	70	119	156	156	157	156	155	130	127	157
arquivo.pt	1,569	30	14	14	15	15	-	-	-	-	1	1	-	163	167	166	163	162	167	165	164	162	-
web.archive.org	1,566	73	73	73	69	71	71	72	73	72	73	72	72	72	72	70	69	69	67	70	71	72	70
archive.is	1,396	11	10	9	12	10	12	14	13	18	14	20	33	25	29	28	59	12	214	214	214	213	212
archive-it.org	1,383	17	15	2	1	3	1	1	-	1	51	109	107	108	105	109	107	106	109	107	107	109	108
swap.stanford.edu	1,222	-	-	-	-	-	-	-	-	-	-	-	21	77	185	166	119	135	164	180	140	21	14
nationalarchives.gov.uk	994	-	-	-	-	-	-	1	2	25	12	50	40	97	117	106	110	104	94	83	59	54	40
europarchive.org	979	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	120	219	72	172	146	213	37
webharvest.gov	712	-	-	-	-	-	-	-	-	128	-	126	-	91	-	129	2	127	59	38	12	-	-
digar.ee	488	-	-	-	-	-	-	-	-	-	-	-	-	-	-	36	95	69	89	69	74	56	-
webarchive.proni.gov.uk	469	-	-	-	-	-	-	-	-	-	-	-	-	-	-	17	94	19	75	75	78	59	52
collectionscanada.gc.ca	351	-	-	-	-	-	-	-	-	-	40	173	138	-	-	-	-	-	-	-	-	-	-
webarchive.org.uk	349	-	-	-	-	-	-	-	-	-	6	9	10	31	34	31	34	34	30	34	29	34	33
archive.bibalex.org	199	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	99	98	-	-	-	-	-
perma.cc	182	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	23	53	53	53
Total	**16,627**	137	121	110	109	114	197	201	202	385	371	824	677	904	1011	1215	1442	1550	1547	1635	1528	1421	926

TABLE 4: URI-R source count.

NAME		**Access time**	**URI-Rs**	**URI-Rs after removing duplicates**
MOZ		2017-06-08	500	500
Memento damage		2017-06-08	1,535	1,535
HTTP Archive		2017-04-15	6,657,856	6,657,856
WAHR	#climatemarch	2017-04-19 – 2017-05-03	175,278	41,674
	#MarchForScience	2017-04-12 – 2017-04-26	299,124	90,318
	#porteouverte #paris #Bataclan #parisattacks	2015-12-11	5,561,037	857,490
	#WomensMarch	2017-01-12 – 2017-01-28	2,403,637	526,903
	#YMMfire	2016-08-20	335,276	45,327
Total			**15,434,243**	**8,220,606**

We collected URI-Rs from four different sources. The first 500 URI-Rs were from Moz [171], which provides a list of the top 500 domains on the web. The second set consists of 1,535 URI-Rs from a previous study [172] about investigating memento damage. The third set contains 6,657,856 URI-Rs that are publicly available in the HTTP Archive [173]. The final set of URI-Rs (8,774,352) is from the Web Archives for Historical Research group (WAHR) [174].

We included the first two sources even though they are relatively small compared to the other two sources because (1) we wanted our final selected set of URI-Rs to have some top/well-known web pages (i.e., URI-Rs from Moz), and (2) the URI-Rs from the study of memento damage contains a mixture of URI-Rs with different path lengths (e.g., `www.example.com/path/to/file.html`). The main characteristic of URI-Rs that belong to the first and third source (i.e., Moz and HTTP Archive) is that the URI-R consists of a domain name only (e.g., `www.example.com`). The URI-Rs from WAHR are extracted from tweets about the hashtags `#climatemarch`, `#MarchForScience`, `#porteouverte`, `#paris`, `#Bataclan`, `#parisattacks`, `#WomensMarch`, and `#YMMfire` between December 11, 2015 and May 3, 2017. Table 4 shows the number of collected URI-Rs including the number of URI-Rs by hashtag from the WAHR source. The total number of unique URI-Rs from all four sources is 8,220,606.

```
$ python canonicalize.py http://www.example.com
com,example)/

$ python canonicalize.py http://www.example.com:80
com,example)/

$ python canonicalize.py www.EXAMPLE.com
com,example)/
```

Fig. 77: An example showing three different URI-Rs that map to the same URI-R (in SURT format [6]) using the canonicalization function from [7]

We merged all 8,220,606 unique URI-Rs from the four sources into a single list. The order of how URI-Rs are placed on the list is as follows:

1. Moz's URI-Rs were placed on the top of this list followed by URI-Rs from our Memento damage study.
2. We repeatedly selected 10 URI-Rs from HTTP Archive and 10 URI-Rs from WAHR, choosing 10 from a different hashtag each round.

The order of URI-Rs in the list is important because we decided to work with a smaller number of URI-Rs for our study.

5.2 MEETING THE DATASET REQUIREMENTS

From this initial set of 8,220,606 URI-Rs, we used four different methods in sequence to meet the requirements outlined earlier.

5.2.1 METHOD 1: SELECT THE 10,000 URI-RS

To limit the total number of URI-Rs, we chose to start with the first 10,000 unique URI-Rs from our initial dataset of 8,220,606 that met certain conditions. URI-Rs must be canonicalized to determine whether or not a URI-R with a particular domain name and file path length has already been selected. We used the canonicalization function that is part of PyWb [7]. The function indicates that `http://www.example.com`, `http://www.example.com:80`, and `www.EXAMPLE.com` are the same, as shown in Figure 77. The output of this canonicalization function is in Sort-friendly URI Reordering Transform (SURT) format [6].

In addition to the canonicalization function, we issued an HTTP HEAD request to discover if two URI-Rs redirect to the same web resource. As Figure 78 shows, sending a

```
$ curl -sIL fb.com | egrep -i "(HTTP/|^location:)"
HTTP/1.1 301 Moved Permanently
Location: https://www.facebook.com/
HTTP/2 200
```

Fig. 78: An example showing two different URI-Rs that redirect to the same URI-R.

HTTP HEAD request to `www.fb.com` redirects to `https://www.facebook.com/` (which we already had), and we selected only the latter.

Also, the selected URI-Rs must contain a variety of file path lengths that we group into the following five sets, each of which contains 2,000 URI-Rs:

- s_0: Path length of zero

  - `www.example.com`

- s_1: Path length of one

  - `www.example.com/file1.html`

- s_2: Path length of two

  - `www.example.com/1/file2.html`

- s_3: Path length of three

  - `www.example.com/1/2/file3.html`

- s_{4+}: Path length of four or more

  - `www.example.com/1/2/3/file4.html`

The final two conditions for selecting the first 10,000 URI-Rs are as follows:

1. URI-Rs with the same file path length should not have the same domain name. For example, if `www.youtube.com/watch?v=cpPG0bKHYKc` has already been selected, then `www.youtube.com/watch?v=hFhiV5X5QM4` will not be selected. This may help to collect more unique URI-Rs and vary the content we plan to study.

2. The TimeMaps of selected URI-Rs must contain at least one memento, as our further work is to study any change or transformation in the content of mementos over time.

To retrieve TimeMaps, we used the LANL Memento aggregator[1]. Once a TimeMap is downloaded, we reduced the number of mementos in the TimeMap to one memento per year from each archive. TimeMaps returned from LANL's aggregator have more information and metadata than we need for our further study. Therefore, we wrote two Python scripts, available on Github[2]. The script `timemap.py` extracts only URI-Ms and their `Memento-Datetime` from the returned TimeMaps, while the second script `yearly-filter.py` filters TimeMaps by selecting one memento (the first) per year by archive. Figure 79 shows an example of a TimeMap with 64 mementos of the URI-R `http://www.futureofmusic.org/about/positions.cfm`, and Figure 80 shows the corresponding TimeMap after filtering. It contains only 10 mementos.

Table 5 shows the number of selected URI-Rs per source and path length, and Table 6 shows that 13% of the selected URI-Rs currently have either the HTTP status code 4xx or 5xx. Even though these URI-Rs are no longer live, they are archived.

TABLE 5: The initial collected set of URI-Rs per source by path length (results of Method 1).

	Path length					
Source	$\mathbf{s_0}$	$\mathbf{s_1}$	$\mathbf{s_2}$	$\mathbf{s_3}$	$\mathbf{s_{4+}}$	**Total**
MOZ	286	17	3	2	1	**309**
HTTP Archive	1,581	42	70	2	0	**1,695**
Memento Damage	114	63	62	42	60	**341**
#climatemarch	4	74	98	89	99	**364**
#MarchForScience	1	162	173	139	243	**718**
#porteouverte #paris #Bataclan #parisattacks	8	758	716	855	711	**3,048**
#WomensMarch	3	723	734	749	734	**2,943**
#YMMfire	3	161	144	122	152	**582**
Total	**2,000**	**2,000**	**2,000**	**2,000**	**2,000**	**10,000**

Table 7 (column **Method 1**) shows the list of 16 web archives from which the mementos of our 10,000 URI-Rs are collected (there is one archive, `nationalarchives.gov.uk`, that has not been counted yet because it has not contributed any mementos). The total number of URI-Rs in the table exceeds 10,000 because a URI-R often has mementos in multiple archives, resulting in some URI-Rs being counted multiple times, but the total number of

[1] `http://timetravel.mementoweb.org/guide/api/`

[2] `https://github.com/oduwsdl/mementos-fixity`

```
1
2 % python timemap.py http://www.futureofmusic.org/about/positions.cfm >
    full-timemap.txt
3 % cat full-timemap.txt
4 20120328211040 http://www.webcitation.org/66VfNacdz
5 20141021161223 http://archive.is/20141021161223/http://www.
    futureofmusic.org/about/positions.cfm
6 20141021175005 http://archive.is/20141021175005/http://www.
    futureofmusic.org/about/positions.cfm
7 20141021175817 http://archive.is/20141021175817/http://www.
    futureofmusic.org/about/positions.cfm
8 20141106145319 http://archive.is/20141106145319/http://www.
    futureofmusic.org/about/positions.cfm
9 20141106151301 http://archive.is/20141106151301/http://www.
    futureofmusic.org/about/positions.cfm
10 20070114182707 https://web.archive.org/web/20070114182707/http://www.
    futureofmusic.org:80/about/positions.cfm
11 ... <18 mementos from 2007-2008> ...
12 20090122061339 https://web.archive.org/web/20090122061339/http://
    futureofmusic.org:80/about/positions.cfm
13 20090228213737 https://web.archive.org/web/20090228213737/http://
    futureofmusic.org:80/about/positions.cfm
14 20120607045812 https://web.archive.org/web/20120607045812/http://www.
    futureofmusic.org/about/positions.cfm
15 20120607045828 https://web.archive.org/web/20120607045828/http://
    futureofmusic.org/about/positions.cfm
16 20130323010922 https://web.archive.org/web/20130323010922/http://www.
    futureofmusic.org/about/positions.cfm
17 20130323011136 https://web.archive.org/web/20130323011136/http://
    futureofmusic.org/about/positions.cfm
18 20131231022915 https://web.archive.org/web/20131231022915/http://www.
    futureofmusic.org/about/positions.cfm
19 20140819212552 https://web.archive.org/web/20140819212552/http://www.
    futureofmusic.org/about/positions.cfm
20 20150320143837 https://web.archive.org/web/20150320143837/http://www.
    futureofmusic.org/about/positions.cfm
21 20160325184708 https://web.archive.org/web/20160325184708/http://www.
    futureofmusic.org/about/positions.cfm
22 20070114182707 https://web.archive.org/web/20070114182707/http://www.
    futureofmusic.org:80/about/positions.cfm
23 20070209043456 https://web.archive.org/web/20070209043456/http://www.
    futureofmusic.org:80/about/positions.cfm
24 ... <26 duplicate mementos from web.archive.org> ...
```

Fig. 79: The TimeMap of `http://www.futureofmusic.org/about/positions.cfm` contains 64 mementos from three different archives: `https://web.archive.org`, `https://archive.is`, and `https://www.webcitation.org`.

```
% cat full-timemap.txt | python yearly-filter.py > yearly-filter.txt
% cat yearly-filter.txt
20120328211040 http://www.webcitation.org/66VfNacdz
20141021161223 http://archive.is/20141021161223/http://www.
    futureofmusic.org/about/positions.cfm
20070114182707 https://web.archive.org/web/20070114182707/http://www.
    futureofmusic.org:80/about/positions.cfm
20080109053549 https://web.archive.org/web/20080109053549/http://www.
    futureofmusic.org:80/about/positions.cfm#ed
20090122061339 https://web.archive.org/web/20090122061339/http://
    futureofmusic.org:80/about/positions.cfm
20120607045812 https://web.archive.org/web/20120607045812/http://www.
    futureofmusic.org/about/positions.cfm
20130323010922 https://web.archive.org/web/20130323010922/http://www.
    futureofmusic.org/about/positions.cfm
20140819212552 https://web.archive.org/web/20140819212552/http://www.
    futureofmusic.org/about/positions.cfm
20150320143837 https://web.archive.org/web/20150320143837/http://www.
    futureofmusic.org/about/positions.cfm
20160325184708 https://web.archive.org/web/20160325184708/http://www.
    futureofmusic.org/about/positions.cfm
```

Fig. 80: The TimeMap of `https://www.futureofmusic.org/about/positions.cfm` after filtering. It contains only 10 mementos (the first memento per year is selected from each archive).

TABLE 6: The final URI-R HTTP status codes of the initial collected set of URI-Rs (results of Method 1).

	HTTP status code		
Path length	**200**	**4xx/5xx**	**Total**
$\mathbf{s_0}$	1,870	130	**2,000**
$\mathbf{s_1}$	1,651	349	**2,000**
$\mathbf{s_2}$	1,715	285	**2,000**
$\mathbf{s_3}$	1,720	280	**2,000**
$\mathbf{s_{4+}}$	1,731	269	**2,000**
Total	**8,687**	**1,313**	**10,000**

unique URI-Rs is still 10,000. The total number of URI-Ms in all TimeMaps is 12,988,039. This number drops to 48,199 URI-Ms after applying the one memento per year filter.

From Table 7, we notice that several archives have a small number of URI-Rs and URI-Ms. Since we want to study the playback fidelity of the web archives, we chose 200 as the minimum number of URI-Rs per archive. After applying Method 1, we used three additional methods to discover more mementos from web archives that have fewer than 200 URI-Rs.

TABLE 7: The four methods used to collect URI-Rs/URI-Ms. The table indicates (shown in bold) that (1) seven archives satisfy the condition of 200 URI-Rs by Method1 (2) five additional archives satisfy the condition of 200 URI-Rs by Method 2, (3) four other archives satisfy the condition by Method 3, and (4) the last archive that satisfies the condition of 200 URI-Rs by Method 4.

	Method 1		Method 2		Method 3		Method 4	
Archive	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**
web.archive.org	**32,139**	**9,790**	40,258	10,353	45,155	10,924	45,155	10,924
archive.is	**2,322**	**1,284**	3,229	1,526	3,471	1,654	3,471	1,654
archive-it.org	**3,500**	**804**	7,986	1,355	8,994	1,593	8,994	1,593
archive.bibalex.org	**3,363**	**568**	6,176	941	7,286	1,148	7,286	1,148
webarchive.loc.gov	**2,721**	**418**	7,122	934	7,766	1,062	7,766	1,062
arquivo.pt	**1,410**	**324**	3,154	758	3,430	895	3,430	895
webcitation.org	**1,125**	**472**	1,858	725	1,954	775	1,954	775
europarchive.org	407	106	**911**	**287**	992	324	992	324
swap.stanford.edu	609	132	**1,176**	**283**	1,233	304	1,233	304
vefsafn.is	19	7	**1,520**	**246**	1,715	294	1,715	294
webharvest.gov	84	21	**743**	**227**	826	248	826	248
webarchive.org.uk	12	5	27	8	**907**	**228**	907	228
digar.ee	333	129	**513**	**223**	518	228	518	228
webarchive.proni.gov.uk	138	48	316	141	**480**	**213**	480	213
nationalarchives.gov.uk	0	0	0	0	**1,011**	**200**	1,011	200
collectionscanada.gc.ca	8	6	59	50	**359**	**200**	359	200
perma.cc	9	6	101	71	154	111	**290**	**200**
Total	**48,199**	**14,120**	**75,149**	**18,128**	**86,251**	**20,401**	**86,387**	**20,490**

5.2.2 METHOD 2: DISCOVER ADDITIONAL URI-RS FROM THE HTML OF ALREADY COLLECTED MEMENTOS

For each archive that has not satisfied the 200 URI-Rs condition, we downloaded the raw content of already collected mementos from the archive and extracted all URI-Rs found in the HTML. Using the LANL Memento aggregator, we requested the TimeMap of each URI-R that had not already been selected. We applied this method for the following archives:

1. webharvest.gov
2. swap.stanford.edu
3. vefsafn.is
4. webarchive.org.uk
5. webarchive.proni.gov.uk
6. collectionscanada.gc.ca
7. perma.cc

```
1 $ python extract_urirs.py http://wayback.vefsafn.is/wayback
    /20041020191800id\_/http://www.w3.org/
2 http://www.csail.mit.edu/
3 http://www.google.com/
4 http://www.ilog.com/
5 http://www.inria.fr/
6 http://jigsaw.w3.org/css-validator/
7 http://www.w3.org/People/Raggett/tidy/
8 http://validator.w3.org/
9 http://www.w3.org/2004/MWeb/Overview.html
10 http://purl.org/rss/1.0/
11 ...
```

Fig. 81: An example of extracting URI-Rs from the HTML of the memento `http://wayback.vefsafn.is/wayback/20041020191800id_/http://www.w3.org/` (only 9 URI-Rs, out of 138, are shown). Notice that we used the option `id_` in the URI-M to retrieve the archived unaltered, or raw, content of the memento.

Three archives are not included in the list above. The first reason being that Method 2 cannot be applied for `nationalarchives.gov.uk` because the archive has not yet provided any mementos. The second is that the archives `europarchive.org` and `digar.ee` satisfied the condition of 200 URI-Rs after applying Method 2 to `swap.stanford.edu` and `vefsafn.is`, respectively.

As shown in Table 8, any new discovered URI-Rs/URI-Ms with this method caused the information from all archives to be updated even for archives that already had more than 200 URI-Rs. Figure 81 shows an example of URI-Rs extracted from the HTML of the memento

`https://wayback.vefsafn.is/wayback/20041020191800id_/http://www.w3.org/`

The URI-Rs are extracted from the attribute `href` in the `<a>` tags (using the Python script `extract_urirs.py`[3]). We downloaded the TimeMap of the URI-R `www.inria.fr/` which had not previously been selected. As Figure 82 shows, the TimeMap does not only contain mementos from `vefsafn.is` but also mementos from eight other archives: `web.archive.org`, `archive.bibalex.org`, `webcitation.org`, `webarchive.loc.gov`, `archive-it.org`, `archive.is`, `vefsafn.is`, and `digar.ee`.

With **Method 2**, we now have 200 URI-Rs for the following additional archives:

8. `webharvest.gov`

[3] `https://github.com/oduwsdl/mementos-fixity`

TABLE 8: After applying Method 2 to seven archives, five archives satisfy the condition of 200 URI-Rs (shown in bold). Notice that applying this method to one archive may increase the number of URI-Rs in other archives (e.g., applying method 2 for `vefsafn.is` makes both `vefsafn.is` and `digar.ee` satisfy the 200 URI-R condition).

	Method 2							
	webharvest.gov		**swap.stanford .edu**		**vefsafn.is**		**webarchive.org .uk proni.gov.uk collectionscanada .gc.ca perma.cc**	
Archive	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**
`web.archive.org`	34,819	9,968	36,385	10,075	39,092	10,265	40,258	10,353
`archive.is`	2,330	1,289	2,562	1,359	3,093	1,479	3,229	1,526
`archive-it.org`	5,108	979	5,999	1,095	7,373	1,271	7,986	1,355
`archive.bibalex.org`	4,169	675	4,764	762	5,827	891	6,176	941
`webarchive.loc.gov`	4,432	590	5,297	698	6,598	860	7,122	934
`arquivo.pt`	1,762	456	2,158	553	2,935	689	3,154	758
`webcitation.org`	1,290	532	1,415	593	1,737	689	1,858	725
europarchive.org	476	137	**603**	**202**	842	265	911	287
swap.stanford.edu	651	145	**797**	**200**	1,072	261	1,176	283
vefsafn.is	19	7	19	7	**1,270**	**200**	1,520	246
webharvest.gov	**647**	**200**	698	212	711	214	743	227
digar.ee	339	134	362	153	**491**	**212**	513	223
`webarchive.proni.gov.uk`	144	52	156	58	224	84	316	141
`perma.cc`	58	38	74	50	76	52	101	71
`collectionscanada.gc.ca`	26	22	40	35	42	37	59	50
`webarchive.org.uk`	12	5	12	5	12	5	27	8
`nationalarchives.gov.uk`	0	0	0	0	0	0	0	0
Total	**56,282**	**15,229**	**61,341**	**16,057**	**71,395**	**17,474**	**75,149**	**18,128**

9. `swap.stanford.edu`
10. `vefsafn.is`
11. `digar.ee`
12. `europarchive.org`

Table 7 (column **Method 2**) shows the new archives that satisfy the condition of 200 URI-Rs and the four archives which still do not satisfy the condition.

5.2.3 METHOD 3: DISCOVER URI-RS IN ARCHIVES' PUBLISHED LISTS

Some archives make lists of URI-Rs they collect available on the web. Archives may also publish lists of URI-Ms associated with each URI-R. We found these published collections for three archives (Table 9) that had not yet met the 200 URI-R minimum.

We downloaded the published list of URI-Rs only (URI-Ms were not included in this list) from the archive `webarchive.org.uk`. Then, using the LANL Memento aggregator

```
$ python timemap.py http://www.inria.fr/
19961230035541 https://web.archive.org/web/19961230035541/http://www4.
    inria.fr:80/
19961230035541 http://web.archive.bibalex.org:80/web/19961230035541/
    http://www4.inria.fr/
20140729051225 http://www.webcitation.org/6RQAbDGPm
20020808175122 http://webarchive.loc.gov/all/20020808175122/http://www.
    inria.fr/
20100731132417 http://wayback.archive-it.org/all/20100731132417/http://
    www.inria.fr/
19961230035541 http://archive.is/19961230035541/http://www.inria.fr/
19961013190926 https://arquivo.pt/wayback/19961013190926/http://www.
    inria.fr/
20110325131647 http://veebiarhiiv.digar.ee/a/20110325131647/http://www.
    inria.fr/
...
```

Fig. 82: Downloading the TimeMap of the URI-R `http://www.inria.fr/`.

TABLE 9: Archives' published lists of URI-Rs and URI-Ms used in **Method 3**.

Archive	List	URI-Rs	URI-Ms
`webarchive.org.uk`	`data.webarchive.org.uk/opendata/ukwa.ds.1/`	26,910	-
`collectionscanada.gc.ca`	`collectionscanada.gc.ca/webarchives/url-list/index-e.html`	2,613	27,232
`nationalarchives.gov.uk`	`nationalarchives.gov.uk/webarchive/atoz/`	4,956	168,328

we retrieved TimeMaps of the first 192 URI-Rs that contain at least one memento in the UK Web Archive. Table 7 (column **Method 3**) shows that this method helps two archives to reach 200 URI-Rs (i.e., `webarchive.proni.gov.uk` and `webarchive.org.uk`), but at the same time, a new web archive appears in the TimeMaps, `nationalarchives.gov.uk`, raising the total number of archives to 17.

Next, we downloaded lists of URI-Rs and URI-Ms made available by the two web archives `collectionscanada.gc.ca` and `nationalarchives.gov.uk`. We only extracted the number required to reach 200 URI-Rs per archive. With this method, we did not need a Memento aggregator since the archives already provide a list of mementos, but for the sake of consistency, we used LANL Memento aggregator to download TimeMaps, so we can update information for the other archives. Table 7 (column **Method 3**) shows the four new archives

TABLE 10: Applying Method 3 (using archives' published lists) for three archives.

	Method 3					
	webarchive.org.uk		collectionscanada.gc.ca		nationalarchives.gov.uk	
Archive	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**	**URI-Ms**	**URI-Rs**
web.archive.org	41,988	10,572	43,615	10,739	45,155	10,924
archive.is	3,370	1,590	3,426	1,619	3,471	1,654
archive-it.org	8,339	1,431	8,910	1,561	8,994	1,593
archive.bibalex.org	6,630	1,009	7,043	1,091	7,286	1,148
webarchive.loc.gov	7,344	989	7,721	1,039	7,766	1,062
arquivo.pt	3,298	819	3,379	855	3,430	895
webcitation.org	1,892	746	1,944	765	1,954	775
europarchive.org	963	311	986	320	992	324
swap.stanford.edu	1,198	292	1,232	303	1,233	304
vefsafn.is	1,626	270	1,703	289	1,715	294
webharvest.gov	743	227	826	248	826	248
webarchive.org.uk	**812**	**201**	812	201	907	228
digar.ee	515	225	518	228	518	228
webarchive.proni.gov.uk	**460**	**201**	462	203	480	213
nationalarchives.gov.uk	3	1	3	1	**1011**	**200**
collectionscanada.gc.ca	59	50	**359**	**200**	359	200
perma.cc	104	73	154	111	154	111
Total	**79,344**	**83,093**	**19,773**	**19,007**	**86,251**	**20,401**

that have at least 200 URI-Rs:

13. webarchive.org.uk
14. webarchive.proni.gov.uk
15. collectionscanada.gc.ca
16. nationalarchives.gov.uk

Table 7 (column **Method 3**) also shows that for perma.cc we only needed to discover 89 additional URI-Rs to reach 200 URI-Rs. Table 10 shows how the number of URI-Rs/URI-Ms has increased after applying Method 3.

5.2.4 METHOD 4: SEND TIMEMAP REQUESTS DIRECTLY TO AN ARCHIVE

The LANL Memento aggregator may serve cached TimeMaps [96], which may result in TimeMaps that do not contain recently created mementos. For this reason we decided to request TimeMaps for the already selected URI-Rs directly from perma.cc. Figure 83 shows an example of requesting the TimeMap of the URI-R www.whitehouse.gov from perma.cc (the domain perma-archives.org is no longer used in TimeMaps or TimeGates based

on changes on the replay tool made in February 2020 [175]). The TimeMap contains 57 mementos. As shown in Table 7 (column **Method 4**), we were able to obtain the additional 89 URI-Rs for:

17. `perma.cc`

```
$ curl https://perma-archives.org/warc/timemap/*/http://www.whitehouse.gov/
<https://perma-archives.org/warc/timemap/*/http://www.whitehouse.gov/>; rel="self"; type="application/link-format"; from="Thu, 27 Aug 2015 17:14:18 GMT",
<http://www.whitehouse.gov/>; rel="original",
<https://perma-archives.org/warc/timegate/http://www.whitehouse.gov/>; rel="timegate",
<https://perma-archives.org/warc/20150827171418/http://www.whitehouse.gov/>; rel="memento"; datetime="Thu, 27 Aug 2015 17:14:18 GMT",
<https://perma-archives.org/warc/20150827171418/https://www.whitehouse.gov/>; rel="memento"; datetime="Thu, 27 Aug 2015 17:14:18 GMT",
<https://perma-archives.org/warc/20150831171426/https://www.whitehouse.gov/>; rel="memento"; datetime="Mon, 31 Aug 2015 17:14:26 GMT",
...
<https://perma-archives.org/warc/20180302185657/https://whitehouse.gov/>; rel="memento"; datetime="Fri, 02 Mar 2018 18:56:57 GMT",
<https://perma-archives.org/warc/20180302185657/https://www.whitehouse.gov/>; rel="memento"; datetime="Fri, 02 Mar 2018 18:56:57 GMT",
<https://perma-archives.org/warc/20180828214528/https://www.whitehouse.gov/>; rel="memento"; datetime="Tue, 28 Aug 2018 21:45:28 GMT
```

Fig. 83: An example of requesting the TimeMap of `https://www.whitehouse.gov`, which contains 57 mementos (only 6 are shown).

5.3 FILTERING THE FINAL SET OF MEMENTOS

At this point, the selected dataset contained 86,387 URI-Ms, obtained from 20,490 total and 11,222 unique URI-Rs from 17 different web archives. For our preliminary study, we downloaded the rewritten and raw mementos 10 times. We ran 17 parallel processes where each process downloaded mementos from a specific archive. We found that download time varied between web archives. For example, it took about 40 hours to download 733 mementos from `webharvest.gov` and 12 hours to download 1,011 mementos from `nationalarchives.gov.uk`. Thus, we decided to change the number of mementos per archive to what could be downloaded within 40 hours, and the number of mementos must not exceed 1,600 per archive. This produced 18,472 mementos. Unfortunately, we did not check

```
1 $ curl -I http://web.archive.bibalex.org/web/20051026134855/http:/www.
    anchorage.gc.ca/
2 HTTP/1.1 503 Service Unavailable
3 Server: Apache-Coyote/1.1
4 Content-Type: text/html;charset=utf-8
5 Transfer-Encoding: chunked
6 Date: Sun, 31 Mar 2019 13:25:47 GMT
7 Connection: close
```

Fig. 84: Non archival HTTP 503 example. The HTTP response header `Memento-Datetime` is not included in the returned response.

the HTTP status when selecting mementos to make sure they are "200 OK" or archival 4xx/5xx responses (i.e., they have the HTTP response header `Memento-Datetime` for the archives that support the Memento protocol). After selecting the 18,472 mementos, we found that about 10%, 1,975, of these mementos had the HTTP status code of non-archival 4xx or 5xx (1,498 are from `archive.bibalex.org`) as the example in Figure 84 shows. Thus, we removed most of the 4xx/5xx mementos and kept only 130 (out of 1,975). We selected the 130 mementos from multiple archives. Even though these mementos had non-archival 4xx/5xx HTTP status codes, we want to keep track of such mementos in our study (as will be described in Chapter 6) to see if their status codes change. We did not keep all 1,975 mementos because they will impact final results of our study since these mementos do not have any embedded resources (e.g., images). We could not replace those removed mementos because by this time we had already used the selected dataset in our study, and it was not possible to recover any excluded mementos. This resulted in 16,627 mementos remaining.

Table 11 shows the final numbers of selected URI-Rs and URI-Ms per archive (available on GitHub[4]). The table shows that three archives have fewer than 200 URI-Rs for the following reasons:

- `perma.cc`: It took about 40 hours to download 182 mementos from perma.cc, including the raw mementos.
- `archive.bibalex.org`: We removed 1,498 mementos because they returned the "`503 Service Unavailable`" HTTP response code.

Figure 85 shows the distribution of URI-Ms between 1996 and 2017. The main reason for having fewer mementos in years 1996-2005 is because many web archives did not exist during those early years [14, 15]. Figure 86 shows the number of URI-Rs per path length.

[4] `https://github.com/oduwsdl/mementos-fixity/blob/master/final_urims.txt`

TABLE 11: Final numbers in the selected set of URI-Rs and URI-Ms.

Archive	URI-Rs	URI-Ms
web.archive.org	1,566	1,566
archive-it.org	1,338	1,383
archive.is	1,257	1,396
webarchive.loc.gov	1,059	1,594
arquivo.pt	766	1,569
webcitation.org	720	1,585
europarchive.org	321	979
swap.stanford.edu	302	1,222
vefsafn.is	290	1,589
webharvest.gov	247	712
digar.ee	225	488
webarchive.org.uk	221	349
webarchive.proni.gov.uk	209	469
nationalarchives.gov.uk	200	994
collectionscanada.gc.ca	198	351
perma.cc	175	182
archive.bibalex.org	168	199
Total	**9,262**	**16,627**

The number of distinct URI-Rs is 3,698, and of those, 1,996 (54%) have a path length of zero and the remaining 1,702 URIs (46%) have a path length greater than or equal to one.

Figure 87 shows the median number of resources (e.g., images, CSS/JavaScript files, and iframes) comprising composite mementos in our dataset per year. As expected, the figure indicates that web pages contain fewer resources in early years 1996-2006 compared to recent years. There was a mean of 42 embedded resources per page, with a median of 32. On March 1, 2019, HTTP Archive's report [52] indicates that the median number of requests per page is 75, which is almost double the number that we report from our dataset in Figure 87 (i.e., the median number of embedded resources is about 39 for the recent years 2012-2017). The difference between the median number of resources comprising live web pages (reported by HTTP Archive) and the median number of resources comprising archived web pages (our dataset) can be explained as follows:

- According to the HTTP Archive forum [176], all HTTP requests are included in calculating the median regardless of returned HTTP status codes, while in our median calculation we do not count `30x Redirect` responses.
- At replay time, a composite memento might have missing resources that have not been

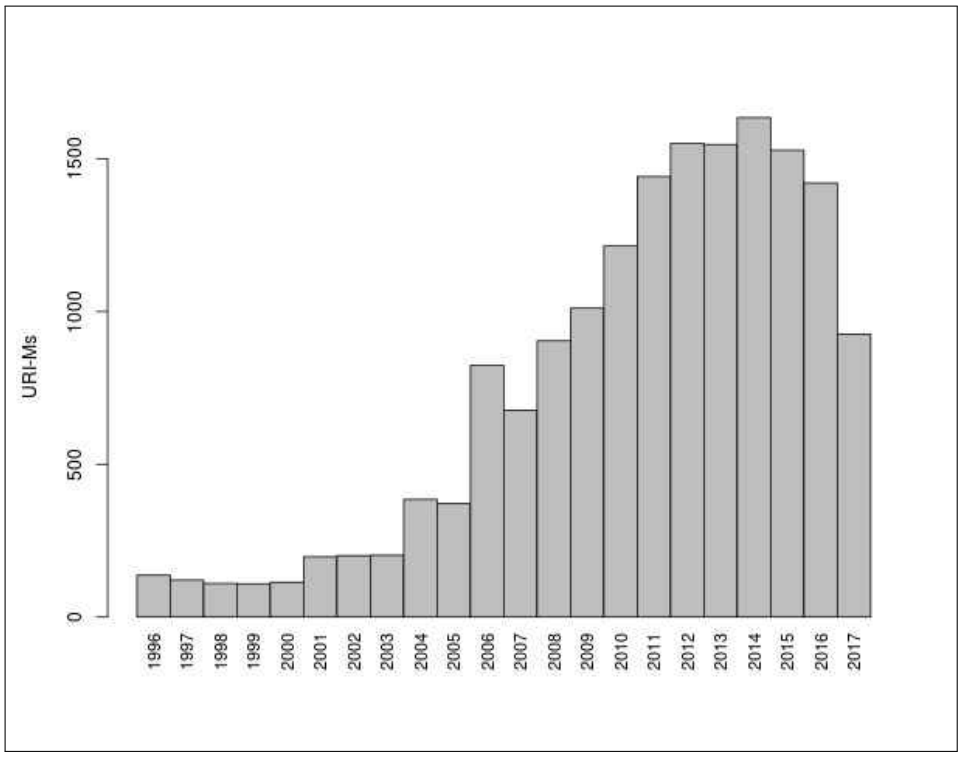


Fig. 85: URI-Ms per year. Note that we collected mementos in November 15, 2017. For this reason, the number of mementos from 2017 is less than the number of mementos in other years, 2010-2016 (i.e., no mementos with a `Memento-Datetime` value after November 15, 2017).

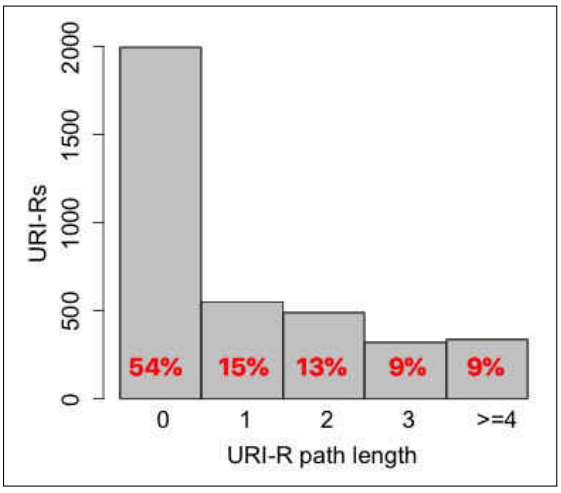


Fig. 86: URI-Rs per path length (54% of URI-Rs are with zero path length).

captured yet. Thus, the archive will not be able to serve those resources at replay time. Furthermore, URI-Rs that are generated on the client-side by JavaScript will not be retrieved from the archive [5] (for not being rewritten) or from the live web (for being blocked for security).

- If archives do not execute JavaScript when crawling web pages, URI-Rs that are generated by JavaScript will not be discovered and captured. This is one of the reasons for observing more embedded resources in live web pages than mementos [84, 177].

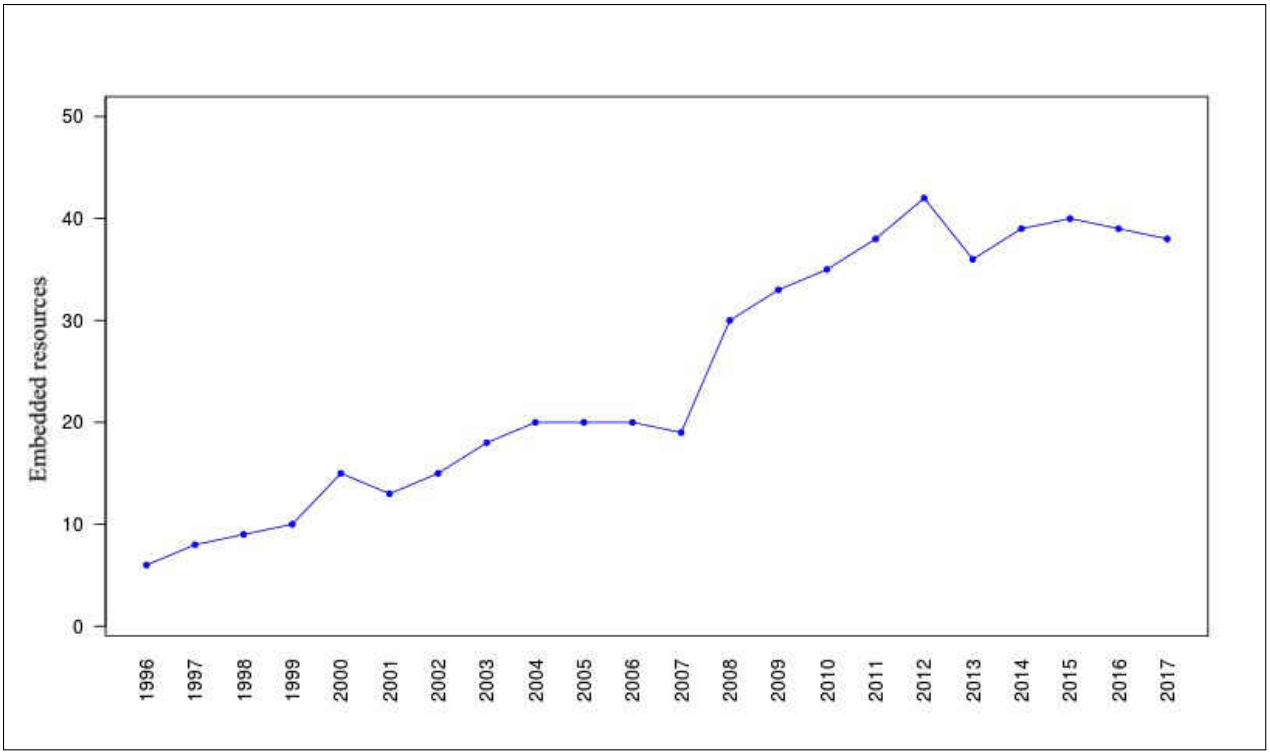


Fig. 87: The median number of embedded resources per memento per year.

5.4 CHAPTER SUMMARY

In this chapter, we explained our four methods for selecting a dataset of 16,627 URI-Ms from 17 public web archives. First, we used the LANL Memento aggregator to download TimeMaps of an initial set of 10,000 URI-Rs. We only selected particular mementos from the downloaded TimeMaps by applying a one memento per year filter. The mementos collected by the first method were selected from 16 public web archives. There were seven archives that satisfied the condition of 200 URI-Rs per archive after applying this first method.

Second, we downloaded the raw content of already collected mementos and extracted all URI-Rs found in the HTML. Similar to the first method, we used the LANL Memento aggregator to download TimeMaps of the discovered URI-Rs. We applied this method to seven archives. It resulted in other five archives satisfying the condition of 200 URI-Rs per archive (at this time we only saw 16 web archives, 12 of them had at least 200 URI-Rs).

Third, we used archives' published lists of URI-Rs/URI-Ms. We applied this method to three archives. This method includes the use of the LANL Memento aggregator to update information (i.e., URI-Rs/URI-Ms) on other archives. A new archive appeared in the downloaded TimeMaps, so the total number of seen archives became 17. This method helped four additional archives to satisfy the 200 URI-Rs condition.

Finally, we sent TimeMap requests of already selected URI-Rs directly to the last archive that did not have 200 URI-Rs. We did not use the LANL Memento aggregator because it might use cached TimeMaps. This method added the final archive to the list of 17 archives that satisfy the 200 URI-Rs condition.

We then filtered the final selected mementos by excluding any memento with the non-archival HTTP status code `4xx/5xx`, which resulted in a final set of 16,627 URI-Ms. We used this dataset in our study of identifying and quantifying the types of changes that cause different fixity information on composite mementos.

CHAPTER 6

CHANGES IN THE PLAYBACK OF MEMENTOS

This chapter explains our study toward addressing RQ1:

RQ1: Can we identify and quantify the types of changes on the playback of mementos that prevent generating repeatable fixity information?

We compute fixity information (hashes) at replay time (client-side) rather than getting fixity information from the archive for two reasons. First, we do not expect hashes generated by archives at crawl time to match those generated on the playback of mementos [178]. Second, if an archive has been compromised then it is likely the corresponding hashes have been also compromised.

In our 14-month study, we observe 16,627 mementos between November 2017 and January 2019. These mementos selected from 17 public web archives. We download each memento 39 times and compute its hash value (for a total of 648,453 hash calculations) to understand and analyze archives' behavior on their replayed archived resources. Even though we avoid archive-specific content in hash calculation when possible and consider only resources that we expect not to change, such as unaltered original content (i.e., raw mementos), we find that 88.45% of mementos produce more than one unique hash value. About 16% of mementos always produce different hashes. Our study also shows that archived collections or archives may migrate to other archives. The migration process of mementos from one archive to other might cause multiple mementos to disappear. This study points to the need for defining a new archive-aware hashing function, as conventional hashing functions are not suitable for replayed archived web pages.

6.1 METHODOLOGY FOR GENERATING FIXITY INFORMATION

We observe the playback of mementos over time and try to understand why it is difficult to generate repeatable fixity information on replayed mementos. Moreover, we want to identify and quantify the types of changes that cause the same mementos to produce different fixity information (e.g., hash values) when replayed at different times.

We used the dataset of 16,627 mementos (described in Chapter 5). Those mementos were downloaded 39 times. We used Merkle trees [146] to generate a single hash for each memento after each download resulting in a total of 39 hashes per memento. This hash value can be viewed as a value that summarizes a memento's content at a particular point in time. The initial assumption of mementos is that they should not change over time, or in other words, for a memento, we should always be able to calculate the same hash each time the memento is downloaded. The next step is to identify the types of changes that cause the same memento to map to different hashes.

Rewritten mementos (*rewritten.warc*)

We used Squidwarc [53, 54] to download mementos and save them as WARC files. Squidwarc uses Google Chrome in headless mode (Headless Chrome) to replay mementos. The headless browsing allows loading an entire web page faster as it runs without the need of the UI. Another powerful feature of Headless Chrome is its ability to execute JavaScript, which leads to the discovering of URIs to embedded resources that otherwise would not be discovered by tools that do not execute JavaScript, such as Wget [51]. We use the term "rewritten" to refer to the content of a memento replayed in Headless Chrome. This content has been rewritten by the archive. Any *rewritten.warc* should contain the HTTP requests and responses (HTTP headers and entity bodies) for all resources comprising a composite memento. Figure 88 illustrates an example that shows the representation of the memento:

`https://web.archive.org/web/19961120150251/http://www.usnews.com:80/`

Table 12 shows the list of all embedded resources including the base HTML file (resource no. 1) involved in constructing the composite memento. The last column "Transformation" in the table indicates the type of modification performed by the archive. The "rewritten" label implies that links to embedded resources are rewritten or that the archive modifies the code by inserting HTML tags, "original" resources are in the original form and have not been modified by the archive, and "archive-specific" refers to resources that are added by the archive.

Raw mementos (*raw.warc*)

For each *rewritten.warc*, we created a corresponding file, *raw.warc*, that contains the WARC records from the raw version of the mementos. The term "raw" refers to the unaltered archived content (described in Chapter 2.2.7). We were not able to create *raw.warc* files for

Fig. 88: The memento `https://web.archive.org/web/19961120150251/http://www.usnews.com:80/`.

mementos from `https://collectionscanada.gc.ca/collectionscanada.gc.ca` (before May 2018) and `webcitation.org` because these archives do not serve the raw content. Based on some changes [179] made by the Library and Archives Canada `collectionscanada.gc.ca` in May 2018, the raw mementos become available. `archive.is` also does not allow accessing the raw content, but it serves mementos in different file formats, including ZIP and PNG. We decided to store the content of the ZIP file in *raw.warc* because even though the content in the ZIP file is transformed, the contents that would change (e.g., banner) are not included. We used the Python module `Requests` to request and download the raw mementos.

Generating Hashes

We have two files, *rewritten.warc* and *raw.warc*, created for each downloaded memento. We used both files to create Merkle Trees [146] and generate one aggregated hash value (i.e., root hash). Any leaf node of a Merkle tree is the hash of data, while any non-leaf node is a hash of its children nodes (i.e., its children's hashes). As Figure 89 and Algorithm 1 show, our technique of generating the root hash of a memento is based on four Merkle

TABLE 12: The embedded resources that comprise the memento `web.archive.org/web/19961120150251/http://www.usnews.com:80/` including the base HTML file. All URIs with a leading slash are relative to the URI web.archive.org.

No	URI	Status	MIME	Transformation
1	/web/19961120150251/http://www.usnews.com:80/	200	text/html	rewritten
2	/static/js/analytics.js?v=1513849547.0	200	application/javascript	archive-specific
3	/static/js/wbhack.js?v=1513849547.0	200	application/javascript	archive-specific
4	/static/js/timestamp.js?v=1513849547.0	200	application/javascript	archive-specific
5	/static/js/graph-calc.js?v=1513849547.0	200	application/javascript	archive-specific
6	/static/js/auto-complete.js?v=1513849547.0	200	application/javascript	archive-specific
7	/static/js/toolbar.js?v=1513849547.0	200	application/javascript	archive-specific
8	/static/images/toolbar/wayback-toolbar-logo.png	200	image/png	archive-specific
9	/static/images/toolbar/wm_tb_prv_off.png	200	image/png	archive-specific
10	/static/images/toolbar/wm_tb_nxt_on.png	200	image/png	archive-specific
11	/static/images/loading.gif	200	image/gif	archive-specific
12	/web/19961120150251im_/http://www.usnews.com:80/usnews/graphics/logo.gif	302		
13	/web/19970725063110im_/http://www.usnews.com:80/usnews/GRAPHICS/logo.gif	200	image/gif	original
14	/static/css/record.css	200	text/css	archive-specific
15	/web/19961120150251im_/http://www.usnews.com:80/usnews/graphics/24hrc15.gif	302 → 404		
16	/web/19961120150251im_/http://www.usnews.com:80/usnews/graphics/netscape.GIF	302 → 404		
17	/web/19961120150251im_/http://www.usnews.com:80/usnews/graphics/infoseek.GIF	302 → 404		
18	/web/19961120150251im_/http://www.usnews.com:80/usnews/graphics/real.GIF	302		
19	/web/19970307190703im_/http://www.usnews.com:80/usnews/GRAPHICS/REAL.GIF	200	image/gif	original
20	archive.org/includes/fonts/Iconochive-Regular.woff?-ccsheb	200	application/octet-stream	archive-specific
21	/__wb/sparkline?output=json&url=http://www.usnews.com:80/&collection=web	200	application/json	archive-specific

trees (marked in different colors), where the output of a Merkle tree becomes the input to another Merkle tree. There are multiple ways to generate a single aggregated hash, but we chose to use the Merkle tree for two main reasons. First, the Merkle tree design is perfectly suited for generating a hash of hashes, and it is used in different well-known technologies, such as Blockchain [138]. Second, the binary hash tree structure of the Merkle tree can be used to quickly detect which resources may cause the same memento to produce different hash values.

For each resource in *rewritten.warc* and *raw.warc*, a Merkle tree (marked in brown in Figure 89) is built on the HTTP response headers of a resource. For instance, the values $\texttt{Hash}_5$ and $\texttt{Hash}_8$ are generated on the HTTP response headers of $\texttt{Resource}_A$ and $\texttt{Resource}_B$, respectively. As described in Chapter 4.1.8, it is important to include some HTTP response headers in hash computation.

Next, another Merkle tree (marked in blue in Figure 89) is used to calculate the hash

TABLE 13: The API used to access the raw content from archives.

Archives	API for retrieving the raw content
archive.org archive-it.org vefsafn.is nationalarchives.gov.uk swap.stanford.edu webarchive.org.uk archive.bibalex.org webarchive.loc.gov arquivo.pt perma-archives.org webharvest.gov veebiarhiiv.digar.ee	/{timestamp}id_/
webarchive.proni.gov.uk europarchive.org	/raw/{timestamp}
archive.is	ZIP file: no raw content; we used this file instead
webcitation.org collectionscanada.gc.ca	unavailable

of each resource. The input to this Merkle tree includes: (1) the resulting hash value on the HTTP response headers generated from the previous step (e.g., $\texttt{Hash}_5$), (2) the hash of the HTTP entity body of the resource (e.g., $\texttt{Hash}_6$), and (3) the hash of the resource's *URI-M* (e.g., $\texttt{Hash}_7$). In some cases the entity body will not be involved in hash calculation when it is not available (e.g., there is often no HTTP entity body for responses with the HTTP status code `302 Redirect`). After this step, we should have a single hash for each resource in *rewritten.warc* and *raw.warc* (e.g., $\texttt{Hash}_3$ of $\texttt{Resource}_A$ and $\texttt{Hash}_4$ of $\texttt{Resource}_B$ in Figure 89).

The next step is to create a Merkle tree (marked in red) that consists of all resources' hashes in each WARC file. This step will result in only two hashes: one hash for *rewritten.warc* (e.g, $\texttt{Hash}_1$) and the other hash for *raw.warc* (e.g, $\texttt{Hash}_2$).

The final step is to calculate the final hash (i.e., `Root Hash`) using a Merkle tree (marked in green in Figure 89) where the input of this tree is the hash of *rewritten.warc* and the hash of *raw.warc*. The resulting hash can be considered as a summary of the content of a memento at a particular time.

We used the Python implementation [180] of Merkle tree to generate one aggregated hash value (root hash) on a composite memento. As the following lines of code show, the

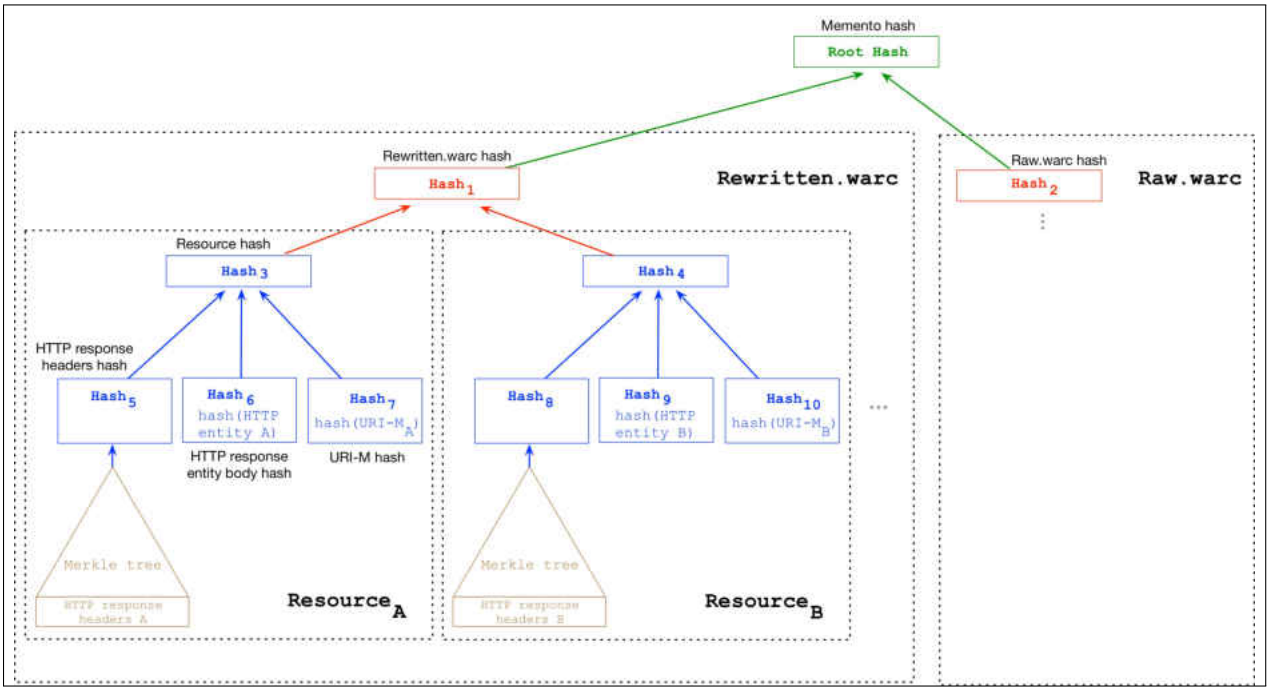


Fig. 89: Generating the root hash of a memento using Merkle trees (marked in different colors) where the output of a Merkle tree becomes input to another Merkle tree. The brown Merkle tree is for generating a hash on HTTP response headers of each resource. The blue Merkle tree generates an overall hash for each resource. The red Merkle tree generates a hash that represents *rewritten.warc* and another hash for *raw.warc*. The root hash is generated by the green Merkle tree.

hash of any intermediate node (non-leaf node) in the Merkle tree is computed by first, combining the hashes of its children as line 9 illustrates (i.e., adding the two hexadecimal representations of the hash values of the children nodes) and then hashing the results as shown in line 3:

```
1               # 3.6 If the current right hash is not a '' <- empty string
2               if current_right != '':
3                   current_right_hash = hashlib.sha256(current_right)
4
5           ...
6
7               # 3.9 Create the new list of transaction
8               if current_right != '':
9                   temp_transaction.append(current_hash.hexdigest() +
                        current_right_hash.hexdigest())
```

The same concept is repeated until we get the root hash. The SHA256 hashing function is used in this Python implementation to calculate hash values.

Algorithm 1 Generate a Root Hash Using Four Merkle Trees

Require: $WARC_{rewritten}$, $WARC_{raw}$
Ensure: $Hash_{root}$

function ROOT_HASH($WARC_{rewritten}$, $WARC_{raw}$)
    $Hash_{rewritten} \leftarrow WARC\_Hash(WARC_{rewritten})$
    $Hash_{raw} \leftarrow WARC\_Hash(WARC_{raw})$
    $Hash_{root} \leftarrow merkleTree(Hash_{reritten}, Hash_{raw})$ ▷ The final root hash
    return $Hash_{root}$
end function

function WARC_HASH($Resources[\,]$)
    $Hash_{resources} \leftarrow [\,]$
    $N \leftarrow length(Resources)$
    for $k \leftarrow 1$ to N **do**
        $Hdrs \leftarrow extractHdrs(Resources_k)$
        $Hash_{Hdrs} \leftarrow merkleTree(Hdrs)$ ▷ The hash on selected headers
        $Entity \leftarrow extractEntity(Resources_k)$
        $Hash_{Entity} \leftarrow hash256(Entity)$
        $URIM \leftarrow extractURIM(Resources_k)$
        $Hash_{URIM} \leftarrow hash256(URIM)$
        $Hash_{rsrc} \leftarrow merkleTree(Hash_{Hdrs}, Hash_{Entity}, Hash_{URIM})$
        ▷ The overall resource hash
        $Hash_{resources}.insert(Hash_{rsrc})$
    end for
    $Hash_{WARC} \leftarrow merkleTree(Hash_{resources})$ ▷ The overall WARC hash
    return $Hash_{WARC}$
end function

In addition to using Merkle trees to generate root hashes, we consider the following rules since not all archives support the Memento protocol or use the same tool for replaying mementos:

- `webcitation.org` and `collectionscanada.gc.ca`: Only resources in *rewritten.warc* are considered in the hash calculation since these archives do not provide the raw

content. But raw mementos became available for `collectionscanada.gc.ca` after download 24, so we included *raw.warc* in the hash calculation after this point.

- `archive.is`: This archive does not provide the raw content, so we only used the content of the ZIP file stored in *raw.warc* in the hash calculation. We do not use *rewritten.warc* in the hash calculation for two reasons. First, it is not used to extract *raw.warc* unlike other archives. Second, both the regular rewritten version stored in *rewritten.warc* and the ZIP file stored in *raw.warc* are actually rewritten, so instead of using two rewritten versions in the hash calculation, we decided to only use *raw.warc* because it has fewer archive-specific resources compared to *rewritten.warc*. For example, `archive.is` does not include the archival banner in the ZIP file.

- The remaining archives: The HTTP entity body of resources in *rewritten.warc* is considered only if it is in the original form and has not been modified by the archive. Without taking this rule into account, it is hard to compute repeatable hashes since archives may insert information that is different each time we request the same resource (e.g., the retrieval time in Figure 25). If the HTTP entity of a resource is modified by the archive, then only selected HTTP response headers are considered in the hash calculation. We expect resources with certain MIME types, such as images, to remain unaltered.

6.2 DEFINING TYPES OF CHANGES

For each memento, we have 39 hash values generated after downloading the mementos 39 times. We compared consecutive hashes; the first hash is compared with the second hash, the second hash is compared with the third hash, and so on. Each time two consecutive hash values were different, we identified one type of change causing different hashes for the same memento. In general, the change could occur on the base HTML file, embedded resources (e.g., images), or HTTP response headers. Before introducing our categorization of changes in mementos, we define, and explain in detail below, the following attributes:

- S: The *set* of all resources (the base HTML file and embedded resources) that comprise a composite memento.

- *URI-M*: A memento of an original resource.

- C: The returned HTML *status code* to a memento request.

- *H*: The set of HTTP response *headers* that we do not expect to change includes `Memento-Datetime`, `Content-Type`, `Location`, and all original response headers that start with `X-Archive-orig-`.
- *R*: The HTTP entity body of a memento.

In general, the attribute *S* is associated with each composite memento, while the attributes *URI-M*, *C*, *H*, and *R* are associated with each memento. When replaying a memento at different times, we expect each attribute defined above (*S*, *URI-M*, *C*, *H*, and *R*) to always have the same value. We defined different types of changes based on how these attributes change. In other words, we compared an attribute's value observed at a particular time to the value of the same attribute observed at a different time and identified differences.

Set

ΔS = (*S'*, *URI-M*, *C*, *H*, *R*): One or more resources in the set comprising a composite memento has changed. This may include observing new resources added, resources replaced with others, or missing resources that were previously part of the composite memento.

Figure 90 shows an example of the *Set*, change where the memento

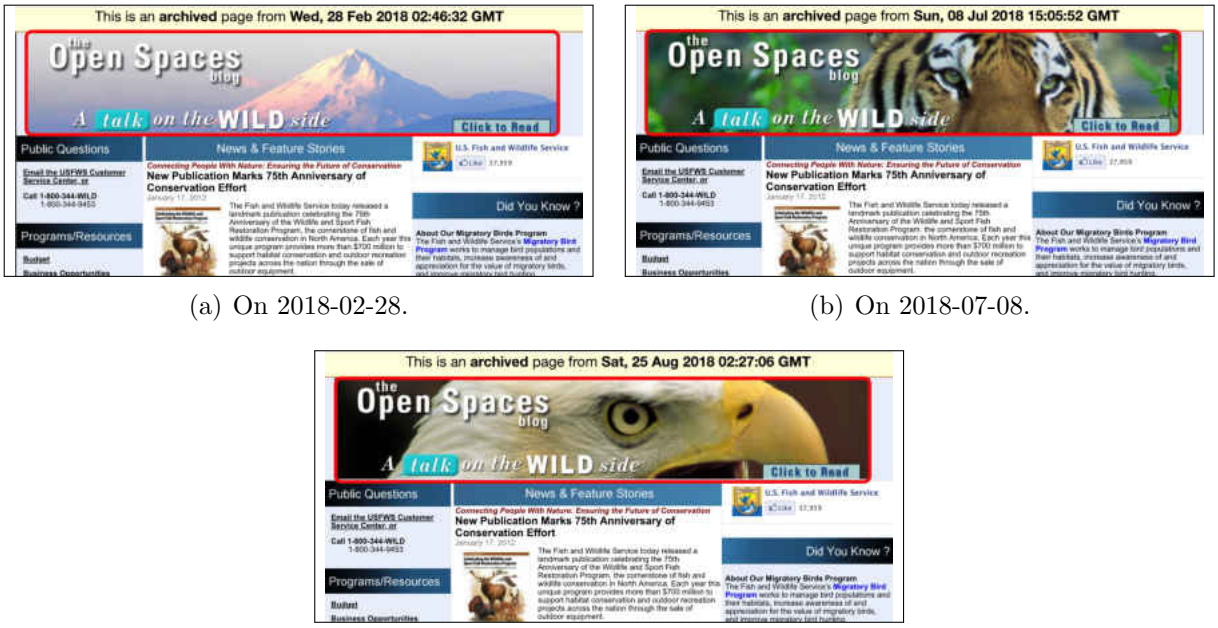
`https://webharvest.gov/congress112th/20130119060624/http://www.fws.gov/`

is downloaded at three different times. At each time, a different background image is displayed. The reason for observing different resources at replay time for the memento is that values in URI-Ms are generated randomly on the client-side by JavaScript as shown in Figure 91. This code results in selecting the image *bannerbluemnt.jpg* on February 28, 2018, the image *bannertiger.jpg* on July 08, 2018, and the image *bannereagle.jpg* on August 25, 2018. The background image (i.e., the URI-M to the background image) is actually selected randomly every time the page is reloaded.

Status code

ΔC = (*S*, *URI-M*, *C'*, *H*, *R*): The HTTP status code of one or more resources comprising a composite memento has changed.

One of the reasons for changes in the HTTP status code of a resource is the technique web archives use to crawl or capture web pages. It is not uncommon to encounter a situation where the embedded resources within a composite memento have different values for



(a) On 2018-02-28.

(b) On 2018-07-08.

(c) On 2018-08-25.

Fig. 90: Downloading the memento `https://www.webharvest.gov/congress112th/20130119060624/http://www.fws.gov/` at three different times produced three different background images.

`Memento-Datetime`. This is because archives, as part of the process of crawling web pages, place all discovered but not yet crawled URIs in a queue so that they are processed later. Thus, it is likely that we see some archived resources with `404 Not Found` at a particular time that then become `200 OK` when revisited at a later time.

Figure 92 shows an example of an HTTP status code change of the image

`https://web.archive.org/web/20141209193553im_/http://wac.450F.edgecastcdn.net/80450F/noisecreep.com/files/2009/06/aaron_a042209eb_200.jpg`

which is embedded in the memento

`https://web.archive.org/web/20141209193553/http://noisecreep.com/aaron-harris-of-isis-talks-twitter/`.

The HTTP status code of the image was `404` the first time it was requested on November 17, 2017 (as illustrated in the red square on Figure 92(a)). When requesting the same memento for the second time on November 18, 2017, the HTTP status code of the same embedded image became `200` as shown in Figure 92(b)). The Internet Archive has performed the following steps in order to successfully serve the image the second time it was requested:

```
function random_imglink(){
myimages[1]="/congress112th/20130119060624/http://www.fws.gov/home/
    feature/home-banner/open-spaces/bannerbluemnt.jpg";
myimages[2]="/congress112th/20130119060624/http://www.fws.gov/home/
    feature/home-banner/open-spaces/bannereagle.jpg";
myimages[3]="/congress112th/20130119060624/http://www.fws.gov/home/
    feature/home-banner/open-spaces/bannertiger.jpg";

var ry=Math.floor(Math.random(1)*myimages.length)

if (ry==0)
   ry=1

document.write('<a href='+'"'+imagelinks[ry]+'"'+'><img src="'+myimages
    [ry]+'" border="0" alt="The Open Spaces Blog. A Talk on the Wild
    Side. Click to Read"></a>')
}
```

Fig. 91: Because of the function `Math.random()`, each time the JavaScript code is executed, an image will be selected randomly.

1. No memento was found for the image in the archive when it was requested for the first time on November 17, 2017.

2. In response, the archive responded with a `302 Redirect` to a specific resource

   `https://web.archive.org/save/_embed/http://wac.450F.edgecastcdn.net/80450F/noisecreep.com/files/2009/06/aaron_a042209eb_200.jpg`

3. Following redirects, the browser requested this resource from the archive.

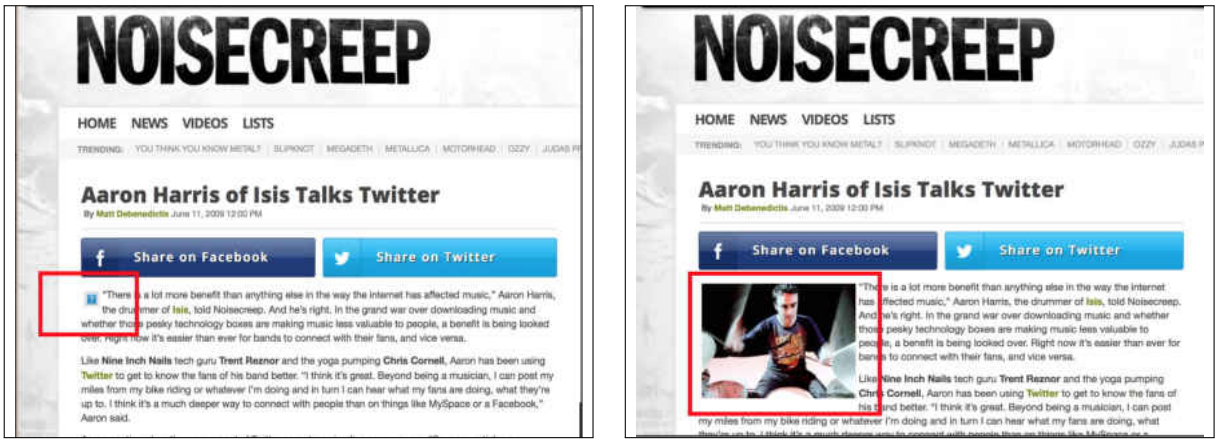
4. This resource (i.e., `web.archive.org/save/_embed/<URI-R>`) triggered a service in the archive for capturing the image from the live web. This service is similar to the Internet Archive's "Save Page Now" feature (`archive.org/web`) through which users can submit any URI-R to the archive.

5. The archive successfully captured the image from the live web and responded with `302 Redirect` to the URI-M of the image

   `https://web.archive.org/web/20171118103250/http://wac.450F.edgecastcdn.net/80450F/noisecreep.com/files/2009/06/aaron_a042209eb_200.jpg`

The archived image's `Memento-Datetime` (November 18, 2017 10:32:50 GMT) is very close to the request's datetime (November 18, 2017 10:33:14 GMT).

6. The browser requested the image and received `200 OK` as a response from the archive.



(a) The image was `404 Not Found` On 2017-11-17. (b) The image became `200 OK` On 2017-11-18.

Fig. 92: The memento `web.archive.org/web/20141209193553/http://noisecreep.com/aaron-harris-of-isis-talks-twitter/` was downloaded at two different times. We noticed two different HTTP status codes of the same embedded image `https://web.archive.org/web/20141209193553im_/http://wac.450F.edgecastcdn.net/80450F/noisecreep.com/files/2009/06/aaron_a042209eb_200.jpg`.

This example indicates that just by requesting mementos, we may actually change the archive, since these requests trigger a service in the archive to capture resources that have not yet been archived. There is a trade-off between capturing resources at the request time (as in the example of Figure 92) and simply returning `404 Not Found`. From the regular viewer's perspective, the archive takes the right action by capturing any missing resources in a composite memento. On the other hand, for a user interested in computing fixity information, this action affects generating repeatable hashes.

In addition, the HTTP status code may change because of transient errors. Archives frequently respond with `500 Error` if unable to serve resources at certain times. Also, the HTTP status code change can occur when archives apply updates to their replay services. For example, after deploying an upgraded version of PyWb, the archive `webarchive.org.uk` starts responding with `307 Temporary Redirect` (Figure 93) to requests that previously returned `302 Found` (Figure 94).

It is possible that different archival redirects for the same HTTP request eventually

return different HTTP status codes as illustrated in Figures 95 and 96. On December 07, 2017, the image

`http://webarchive.loc.gov/all/20001225075832/http://www.senate.gov/resources/sidebar_top.gif`

redirected to a URI-M with an HTTP status code of `200` and a value of the `Memento-Datetime` header of December 10, 2001 07:18:11 GMT. When the same image was requested on December 14, 2017, it redirected to a different URI-M with an HTTP status code of `415` and a value of the `Memento-Datetime` header of December 25, 2000 19:58:25 GMT.

```
1  ...
2  WARC-Type: request
3  WARC-Target-URI: https://www.webarchive.org.uk/wayback/archive
       /20130423035544im_/http://www.local.gov.uk/image/image_gallery?uuid=
       e419bddc-31bc-45a8-8174-5fa393c5758e&groupId=10171&t=1353597690550
4  ...
5  WARC-Type: response
6  ...
7  HTTP/1.1 307 Temporary Redirect
8  Location: https://www.webarchive.org.uk/wayback/archive/20130423030302
       im_/http://www.local.gov.uk/image/image_gallery?uuid=e419bddc-31bc
       -45a8-8174-5fa393c5758e&groupId=10171&t=1353597690550
9  Date: Fri, 30 Nov 2018 21:49:45 GMT
10 Server: nginx/1.14.0
11 ...
12 WARC-Type: request
13 WARC-Target-URI: https://www.webarchive.org.uk/wayback/archive
       /20130423030302im_/http://www.local.gov.uk/image/image_gallery?uuid=
       e419bddc-31bc-45a8-8174-5fa393c5758e&groupId=10171&t=1353597690550
14 ...
15 WARC-Type: response
16 ...
17 HTTP/1.1 200 OK
18 ...
```

Fig. 93: The HTTP response code was `307` on 2018-11-30.

Headers

ΔH = (*S*, *URI-M*, *C*, *H'*, *R*): One or more HTTP response headers, that we do not expect to change, has changed.

Some HTTP response headers are not expected to change, which include `Memento-Datetime`, `Content-Type`, and the original headers that begin with the string `X-Archive-orig-`.

```
1  ...
2  WARC-Type: request
3  WARC-Target-URI: https://www.webarchive.org.uk/wayback/archive
       /20130423035544im_/http://www.local.gov.uk/image/image_gallery?uuid=
       e419bddc-31bc-45a8-8174-5fa393c5758e&groupId=10171&t=1353597690550
4  ...
5  WARC-Type: response
6  ...
7  HTTP/1.1 302 Found
8  Date: Fri, 19 Oct 2018 21:27:35 GMT
9  Server: Apache-Coyote/1.1
10 Location: https://www.webarchive.org.uk/wayback/archive/20130423030302
       im_/http://www.local.gov.uk/image/image_gallery?uuid=e419bddc-31bc
       -45a8-8174-5fa393c5758e&groupId=10171&t=1353597690550
11 ...
12 WARC-Type: request
13 WARC-Target-URI: https://www.webarchive.org.uk/wayback/archive
       /20130423030302im_/http://www.local.gov.uk/image/image_gallery?uuid=
       e419bddc-31bc-45a8-8174-5fa393c5758e&groupId=10171&t=1353597690550
14 ...
15 WARC-Type: response
16 ...
17 HTTP/1.1 200 OK
18 ...
```

Fig. 94: The HTTP response code was 302 on -2018-10-19.

However, we show an example in Figures 97 and 98 of changes in the response header `Content-Type` when requesting the memento

`https://web.archive.org/web/20071111211818/http://images.sohu.com:80/chat_online/market/sohu/140140-1.html`

multiple times. On December 30, 2017, the value of the response header `Content-Type` was `text/html; charset=utf-8`. The value changed to `text/html; charset=gb2312` on January 31, 2018.

Representation

ΔR = (*S*, *URI-M*, *C*, *H*, *R'*): The returned HTTP entity body of one or more resources comprising a composite memento has changed.

Figure 99 shows an example of the *Representation* change where we requested the raw content of the same memento multiple times from `perma.cc`. We were expecting to always be presented with same raw content, but we noticed a different HTTP entity returned each

```
1  WARC-Type: request
2  WARC-Target-URI: http://webarchive.loc.gov/all/20001225075832/http://
       www.senate.gov/resources/sidebar_top.gif
3  ...
4  WARC-Type: response
5  ...
6  HTTP/1.1 302 Found
7  Location: http://webarchive.loc.gov/all/20011210071811/http://www.
       senate.gov/resources/sidebar_top.gif
8  ...
9  WARC-Type: request
10 WARC-Target-URI: http://webarchive.loc.gov/all/20011210071811/http://
       www.senate.gov/resources/sidebar_top.gif
11 ...
12 WARC-Type: response
13 ...
14 HTTP/1.1 200 OK
15 ...
```

Fig. 95: Part of a WARC file shown the request of the image on 2017-12-07. It ends up at a resource with the HTTP status code `200 OK`.

```
1  WARC-Type: request
2  WARC-Target-URI: http://webarchive.loc.gov/all/20001225075832/http://
       www.senate.gov/resources/sidebar_top.gif
3  ...
4  WARC-Type: response
5  ...
6  HTTP/1.1 302 Found
7  Location: http://webarchive.loc.gov/all/20001225195825/http://www.
       senate.gov/resources/sidebar_top.gif
8  ...
9  WARC-Type: request
10 WARC-Target-URI: http://webarchive.loc.gov/all/20001225195825/http://
       www.senate.gov/resources/sidebar_top.gif
11 ...
12 WARC-Type: response
13 ...
14 HTTP/1.1 415 Unsupported Media Type
15 Date: Thu, 14 Dec 2017 05:09:41 GMT
```

Fig. 96: The HTTP request of the same image (Figure 95) on 2017-12-14 ends up at a resource with different HTTP status code `415 Unsupported Media Type`.

```
1  WARC/1.0
2  WARC-Type: response
3  WARC-Target-URI: https://web.archive.org/web/20071111211818/http://
       images.sohu.com:80/chat_online/market/sohu/140140-1.html
4  WARC-Date: 2017-12-30T07:16:53Z
5  ...
6
7  HTTP/1.1 200 OK
8  Date: Sat, 30 Dec 2017 07:16:35 GMT
9  X-App-Server: wwwb-app45
10 Memento-Datetime: Sun, 11 Nov 2007 21:18:18 GMT
11 Server: Tengine/2.1.0
12 X-Archive-Guessed-Charset: utf-8
13 Content-Type: text/html; charset=utf-8
14 X-Archive-Orig-date: Fri, 09 Nov 2007 12:24:22 GMT
15 ...
```

Fig. 97: A change in the response header `Content-Type`. The value of the header was `text/html; charset=utf-8` on 2017-12-30, as our WARC file shows.

```
1  WARC/1.0
2  WARC-Type: response
3  WARC-Target-URI: https://web.archive.org/web/20071111211818/http://
       images.sohu.com:80/chat_online/market/sohu/140140-1.html
4  WARC-Date: 2018-01-31T01:08:13Z
5  ...
6
7  HTTP/1.1 200 OK
8  Date: Wed, 31 Jan 2018 01:07:54 GMT
9  X-App-Server: wwwb-app42
10 Memento-Datetime: Sun, 11 Nov 2007 21:18:18 GMT
11 Server: Tengine/2.1.0
12 X-Archive-Guessed-Charset: gb2312
13 Content-Type: text/html; charset=gb2312
14 X-Archive-Orig-date: Fri, 09 Nov 2007 12:24:22 GMT
15 ...
```

Fig. 98: The value of the response header `Content-Type` changed to `text/html; charset=gb2312` on 2018-01-31.

time. The actual change in the returned content has not been caused by the archive, but by Cloudflare, a third-party service used by the archive. This service modifies content being returned to the user by applying Email Address Obfuscation [181] to hide any email address included in the content and help to prevent spam.

```
1
2 curl -s http://perma-archives.org/warc/20171026200017id_/https://www.
      usa.gov/federal-agencies/a
3 | egrep -i "(cdn-cgi|^Date:)"
4 Date: Tue, 15 May 2018 21:00:45 GMT
5             <a href="/cdn-cgi/l/email-protection#28175b5d4a424d4b5c15690
6             85408690572086146 4c4d5008474e087d067b06086f475e4d5a46454d465
7             c086c4d58495a5c454d465c5b0849464c08694f4d464b414d5b0e4945581
8             34a474c5115405c5c585b1207075f5f5f065d5b49064f475e074e4d4c4d5
9             a494405494f4d464b414d5b0749"
10
11 curl -s http://perma-archives.org/warc/20171026200017id_/https://www.
      usa.gov/federal-agencies/a
12 | egrep -i "(cdn-cgi|^Date:)"
13 Date: Tue, 15 May 2018 21:00:50 GMT
14             <a href="/cdn-cgi/l/email-protection#68571b1d0a020d0b1c55294
15             81448294532482106 0c0d1048070e483d463b46482f071e0d1a06050d061
16             c482c0d18091a1c050d061c1b4809060c48290f0d060b010d1b4e0905185
17             30a070c1155001c1c181b5247471f1f1f461d1b09460f071e470e0d0c0d1
18             a090445090f0d060b010d1b4709"
19
20 curl -silent http://perma-archives.org/warc/20171026200017id_/https://
      www.usa.gov/federal-agencies/a
21 | egrep -i "(cdn-cgi|^Date:)"
22 Date: Tue, 15 May 2018 21:00:51 GMT
23             <a href="/cdn-cgi/l/email-protection#b986caccdbd3dcdacd84f8
24             99c599f894e399f0d7dddcc199d6df99ec97ea9799fed6cfdccbd7d4dcd
25             7cd99fddcc9d8cbcdd4dcd7cdca99d8d7dd99f8dedcd7dad0dcca9fd8d4
26             c982dbd6ddc084d1cdcdc9ca839696cecece97cccad897ded6cf96dfdcd
27             ddccbd8d594d8dedcd7dad0dcca96d8"
```

Fig. 99: An example of the type of change *Representation* (Changes in the HTTP entity).

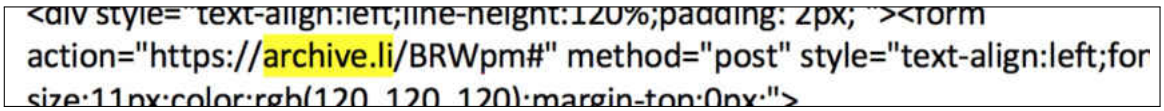
Figure 100 shows another example of HTTP entity change. At replay time, the archive `archive.is` refers to itself inconsistently using different TLDs, such as `.li`, `.is`, and `.today`. In particular, this change occurs in the content of the `index.html` in the returned ZIP file.

Furthermore, we may observe HTTP entity changes because of transient errors as shown in Figure 101. The image

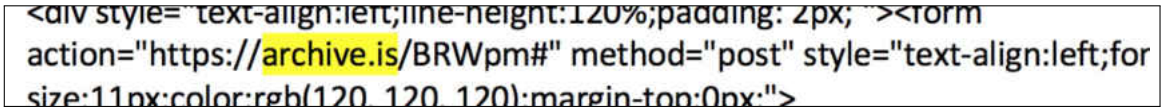
`https://webarchive.nationalarchives.gov.uk/20170303010736id_/https://cereals.ahdb.org.uk/media/1157842/corporate-strategy-1.jpg`

which is embedded in the memento

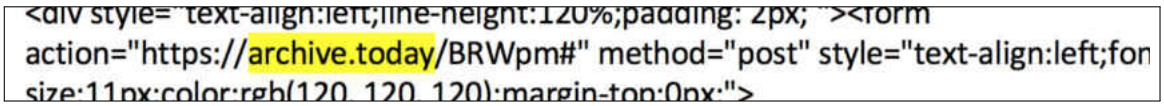
`https://webarchive.nationalarchives.gov.uk/20170303010736id_/https://cereals.ahdb.org.uk/`



(a) The archive uses `.li` on 2017-12-30.



(b) The archive uses `.is` on 2018-01-30.



(c) The archive uses `.today` on 2018-02-05.

Fig. 100: Downloading the ZIP file `http://archive.is/download/BRWpm.zip` of the memento `http://archive.is/BRWpm` at three different times. Each time the archive refers to itself differently in the `index.html` in the ZIP file.

was requested at two different times. The HTTP entity of the image was transferred incompletely the first time it was requested (Figure 101(a)), while the entity was complete when requested for the second time (Figure 101(b)).

(a) Incomplete HTTP entity on 2017-12-07.

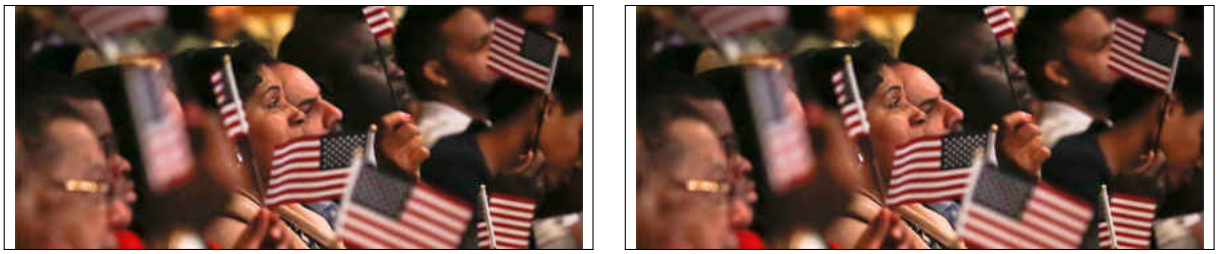
(b) Incomplete HTTP entity on 2017-12-16.

Fig. 101: We noticed a change in the HTTP entity body of the image `http://webarchive.nationalarchives.gov.uk/20170303010736id_/https://cereals.ahdb.org.uk/media/1157842/corporate-strategy-1.jpg` because of a transient error. The image is embedded in the memento `http://webarchive.nationalarchives.gov.uk/20170303010736id_/https://cereals.ahdb.org.uk/`.

URI-M

ΔURI-M = (*S*, *URI-M'*, *C*, *H*, *R*): One or more resources in the set comprising a composite memento redirects (i.e., through an HTTP `30x Redirect`) to a different memento with a different `Memento-DateTime`.

This type of change, in particular, refers to archival redirects that do not result in different HTTP entity. For example, Figure 102 illustrates the scenario where the same HTTP request for an image was sent at two different times. Each time the archive returned a `302 Redirect` to a different URI-M, but the HTTP entities of these two responses were identical.



(a) An image with `Memento-Datetime: Fri, 02 Dec 2016 20:54:58` on 2017-12-01.

(b) Another image with `Memento-Datetime: Thu, 01 Dec 2016 21:31:53` on 2017-12-07.

Fig. 102: The image `https://wayback.archive-it.org/all/20161201183709im_/https://www.ap.org/assets/images/ap-16166678969150-promo-rt.jpg`, which is embedded in the memento `wayback.archive-it.org/all/20161201183709/https://www.ap.org/en-us/`, has been retrieved from the archive through two different URI-Ms with different `Memento-Datetime` values. Both entity bodies of the image are identical.

In general, we identify two URI-Ms as "different" if either their URI-Rs or the values of `Memento-Datetime` are different. However, there are other cases where two lexigraphically different URI-Ms canonicalize to the same value. For example, we requested the same URI-M

`http://perma-archives.org/warc/20170303200708/http://id.loc.gov/`

at two different times on March 27, 2018 (Figure 103) and July 08, 2018 (Figure 104). The only difference between the two responses is that the archive `perma.cc` started serving over HTTPS (instead of HTTP) on or around July 08, 2018. In such cases, without canonicalizing URI-Ms (e.g., HTTP = HTTPS), they will produce different hashes. Considering these two different URI schemes as the same is commonly done in the web archiving community. Figure 105 shows another example where the archive `webharvest.gov` was using the subdomain

www on December 30, 2017. The archive started dropping www around January 30, 2018 (Figure 106).

```
WARC-Type: response
WARC-Target-URI: http://perma-archives.org/warc/20170303200708/http://id.loc.gov/
WARC-Date: 2018-03-27T17:43:38Z
...
HTTP/1.1 200 OK
```

Fig. 103: The archive perma.cc was using HTTP on 2018-03-27.

```
WARC-Type: response
WARC-Target-URI: http://perma-archives.org/warc/20170303200708/http://id.loc.gov/
WARC-Date: 2018-07-08T23:34:13Z
...
HTTP/1.1 301 Moved Permanently
Date: Sun, 08 Jul 2018 23:33:58 GMT
Location: https://perma-archives.org/warc/20170303200708/http://id.loc.gov/
...
WARC-Type: response
WARC-Target-URI: https://perma-archives.org/warc/20170303200708/http://id.loc.gov/
WARC-Date: 2018-07-08T23:34:13Z
...
HTTP/1.1 200 OK
...
```

Fig. 104: The archive perma.cc started using HTTPS on 2018-07-08.

```
WARC-Type: response
WARC-Target-URI: http://webharvest.gov/congress109th/20061114015422/http://www.dc.gov/
WARC-Date: 2017-12-30T04:56:45Z
...
HTTP/1.1 301 Moved Permanently
Location: https://www.webharvest.gov/congress109th/20061114015422/http://www.dc.gov/
...
```

Fig. 105: The archive webharvest.gov was using www on 2017-12-30.

Archives may serve requested mementos in iframes. For example, webarchive.org.uk began supporting the option mp_ for loading the archived content into an iframe. Therefore,

```
1 WARC-Type: response
2 WARC-Target-URI: http://webharvest.gov/congress109th/20061114015422/
      http://www.dc.gov/
3 WARC-Date: 2018-01-30T21:59:52Z
4 ...
5 HTTP/1.1 301 Moved Permanently
6 Location: https://webharvest.gov/congress109th/20061114015422/http://
      www.dc.gov/
7 ...
```

Fig. 106: The archive was omitting "www." on 2018-01-30.

any new requests for mementos from this archive will result in `302 redirect` to a URI-M that has `mp_` after the timestamp.

The *URI-M* change may also be referred to as TimeMap change, where archives respond, at different times, with various TimeMaps of the same *URI-R*. The TimeMap inconsistency occurs for reasons including deduplication and redaction techniques of mementos, archival restructuring, and transient errors [102]. The TimeMap change causes requests of the same memento to redirect to different URI-Ms.

URI-M and Representation

ΔH, ΔR = (*S*, *URI-M'*, *C*, *H*, *R'*): The *URI-M* and *Representation* change occurs when one or more resources in the set comprising a composite memento redirects to a different memento, with different values for both `Memento-DateTime` and HTTP entity body.

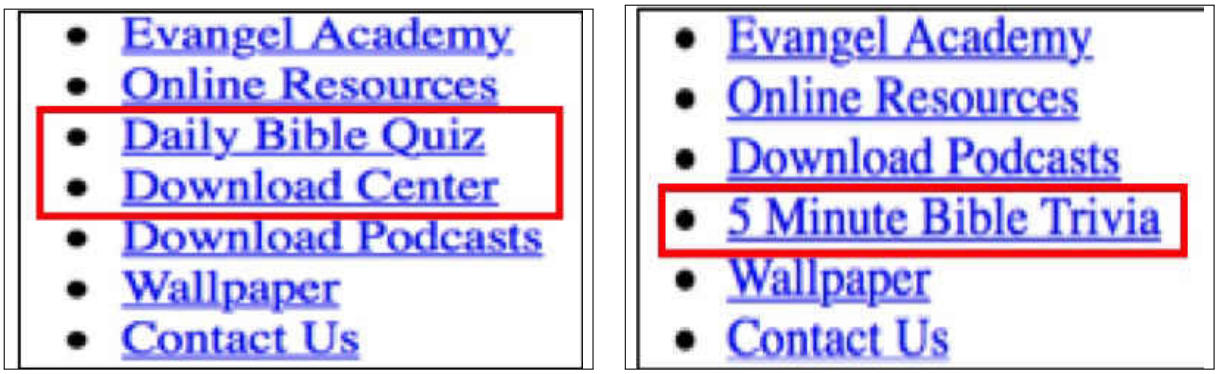
Figure 107 shows that requesting the same base HTML file

`https://web.archive.org/web/20080828005922id_/http://www.evangelcogdayton.org/`

at two different times results in two different HTTP entities. The first HTTP request is made on November 17, 2017, and the archive responded with `200 OK` as shown in Figure 107. We requested the same memento (URI-M) on December 28, 2017. The memento with the `Memento-Datetime` August 28, 2008 00:59:22 GMT redirects to the URI-M with the `Memento-Datetime` November 02, 2009 15:16:09 GMT

`https://web.archive.org/web/20090211151609id_/http://www.evangelcogdayton.org:80`.

which has a different HTTP entity.



(a) The image was HTTP `200 OK` on 2017-11-17. (b) The request redirects (i.e., `302`) on 2017-12-28.

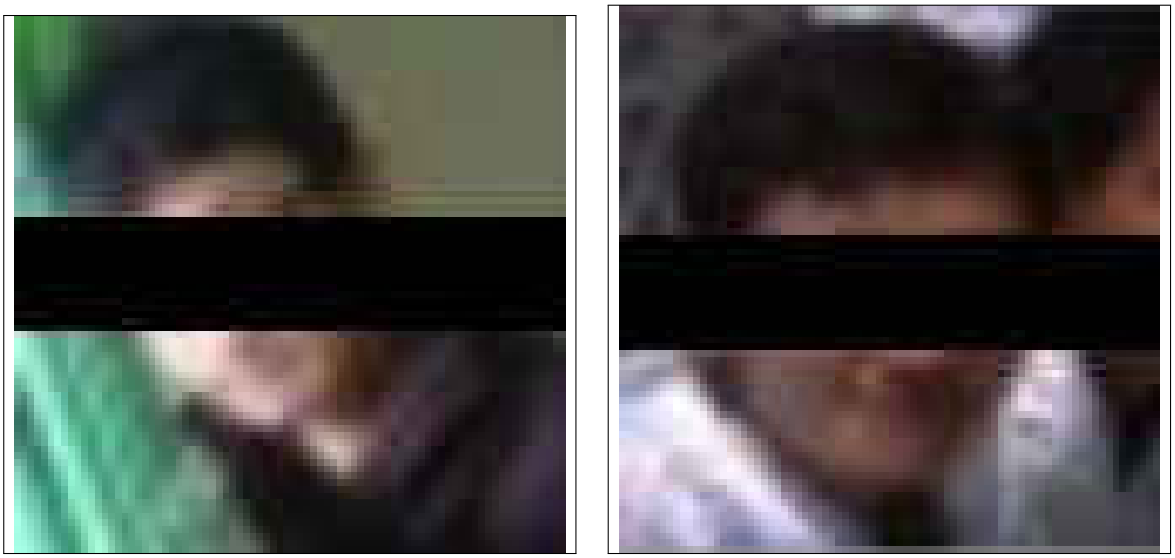
Fig. 107: Requesting the base HTML file `web.archive.org/web/20080828005922id_/http://www.evangelcogdayton.org/` at two different times. The second request on December 28, 2017 redirects to a memento that has a **different HTTP entity**.

The `302 Redirects` are issued based on what resources are available or can be served by the archive at the time of the HTTP requests (i.e., this is also called a TimeMap change as explained in the previous section). Figures 108 and 109 show other examples where changes in URI-Ms (through `302 Redirects`) resulted in different HTTP entities. The HTTP entity change in Figure 107 occurs in the base HTML file, while changes in the HTTP entities in Figures 108 and 109 affect embedded images within composite mementos. Furthermore, the different images in Figure 109 look the same, but we were able to identify differences between the two images using the image comparison tool Resemble [8].

Timeout/Not resolved

The *Timeout/Not resolved* change occurs when one or more HTTP requests of resources in the set comprising a composite memento has a connection `timeout error`. In general, this type of change refers to a situation where there is no HTTP response returned from the server, not even an HTTP status code. As shown in Figure 110, because of the connection `timeout error`, there is no WARC response record in the WARC file for the WARC request record (i.e., `WARC-Record-ID: <urn:uuid:fc...>`). Similarity, Figure 111 shows an example of an HTTP request that does not return an HTTP response because of the `timeout error`. The later example uses `cURL` to download the memento, while the first example (Figure 110) uses Squidwarc to download the memento and create the WARC file.

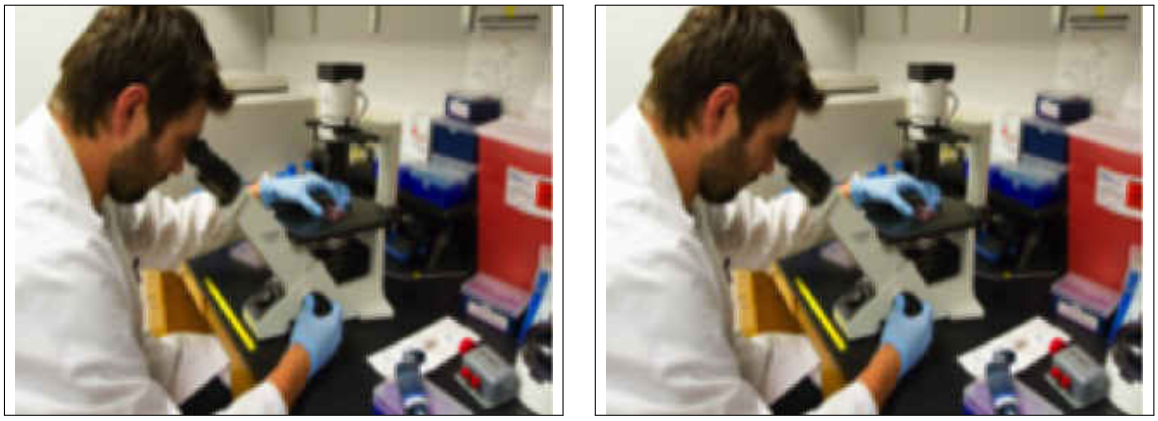
6.3 ONE IN SIX MEMENTOS PRODUCE DIFFERENT HASHES



(a) The image was HTTP `200 OK` on 2018-03-24. (b) The request redirects (i.e., `302`) on 2018-03-27.

Fig. 108: Requesting the image `web.archive.org/web/20110116134258id_/http://1.gravatar.com/avatar/117a6cc4203b951f11fc43f946106657?s=33&d=http%3A%2F%2F1.gravatar.com%2Favatar%2Fad516503a11cd5ca435acc9bb6523536%3Fs%3D33&r=G` which is **embedded** in the memento `https://web.archive.org/web/20110114074814/http://www.copyblogger.com:80/popular-blogger/` at two different times. The first HTTP request returns `200 Ok`, but the second request redirects to a URI-M (with the `Memento-Datetime` January 21, 2012 09:05:32 GMT) that has **different HTTP entity**.

We have calculated 39 hash values for each memento for a total 16,627 mementos. Table 14 shows the number of mementos per archive that have at least two different hashes. The table indicates that most mementos, 14,707 out of 16,627 (88.45%), have at least two different hashes, including all mementos from seven archives. We also show that about 11.55% of mementos (1,920) have only one distinct hash value. This means whenever the hash is calculated, we always obtain the same hash value, which is the initial hypothesis of all conventional fixity-based approaches. In other words, the conventional hashing algorithms work properly only for 11.55% of our set of mementos. On the other hand, about 16.06% (2,670) always produce a different hash. The latter case emphasizes the effect of JavaScript in dynamically generating content causing different hashes. Figure 112 illustrates how the pool of mementos that have at least two distinct hashes has increased over time. About 40.54% of mementos produced different hashes after download 2 on November 18, 2017. Then, after 37 more downloads (within 420 days), this cumulative percentage had increased to reach 88.45% by January 11, 2019.



(a) The image was HTTP `200 OK` on 2017-12-12. (b) The request redirects (i.e., `302`) on 2017-12-25.

(c) Comparing the two images (mismatched pixels in pink).

Fig. 109: Requesting the image `https://perma-archives.org/warc/20170101182814id_/http://umich.edu/includes/image/type/gallery/id/113/name/ResearchDIL-19Aug14_DM%28136%29.jpg/width/152/height/152/mode/minfit` which is embedded in the memento `https://perma-archives.org/warc/20170101182813/http://umich.edu/` at two different times. The first HTTP request returns `200 OK`, while the second HTTP request of the image redirects to a URI-M (with the `Memento-Datetime` June 19, 2017 14:54:58 GMT) which has **different HTTP entity that looks exactly the same**. The two images were compared using Resemble [8] (mismatched pixels are marked in pink).

Figure 113 shows the distribution of the distinct number of hash values per memento. Figure 114 shows the distribution of distinct number of hash values per memento by archive. The blue bar indicates the number of mementos that always produce the same hash value, while the red bar indicates the number of mementos that always produce different hashes. For example, there are 857 (out of 1,569) mementos (marked in red) from `arquivo.pt`

```
1 ...
2 WARC/1.0
3 WARC-Type: request
4 WARC-Target-URI: http://archive.is/20041112085120/http://www.reuters.
      com/
5 WARC-Date: 2018-07-22T09:38:27Z
6 WARC-Concurrent-To: <urn:uuid:fc822790-8d92-11e8-881e-739ebb24b29a>
7 WARC-Record-ID: <urn:uuid:fc8534d0-8d92-11e8-881e-739ebb24b29a>
8 Content-Type: application/http; msgtype=request
9 Content-Length: 323
10
11 GET /20041112085120/http://www.reuters.com/ HTTP/1.1
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Web Science and Digital Libraries Group ...
14 X-DevTools-Emulate-Network-Conditions-Client-Id:
      E511D4DA01A1C06660C5329BEA3CB09F
15 Host: archive.is
```

Fig. 110: The memento with the URI-M `http://archive.is/20041112085120/http://www.reuters.com/` is downloaded on 2018-07-22. Because of the connection `timeout error`, no HTTP response, associated with the HTTP request of the memento, was rewritten in the WARC file. The WARC file consists of 44 lines (only the last 14 lines are shown).

```
1 $ curl -i http://webarchive.bac-lac.gc.ca:8080/wayback/20060128020605/
      http://www.biostrategy.gc.ca/
2
3 $ curl: (56) Recv failure: Operation timed out
```

Fig. 111: An HTTP request returns the `timeout error` on 2018-09-05 (No HTTP response returned for the request). Timeout errors are not considered as HTTP events, so they would not show up in WARC files.

that always produce a different hash value. On the other hand, most mementos from `archive.is` have a maximum of five distinct hashes because this archive does not preserve the original JavaScript code, so no JavaScript code is executed at replay time [84]. The reason for mementos from `archive.is` having five distinct hashes is the transformation of the original content applied by the archive which includes using multiple TLDs and transient errors. Similar to `archive.is`, most mementos from `collectionscanada.gc.ca` have a maximum of four distinct hashes because the `Memento-Datetime` of these mementos (Table 3) is between 2005 and 2007 when JavaScript was less common. In general, early web

TABLE 14: Mementos per archive that produced at least two different hashes, the same hash, or always different hashes.

Archive	URI-Ms	with at least two different hashes (%)	always produced the same hash (%)	always produced a different hash (%)
webarchive.loc.gov	1,594	1,241 (77.85)	353 (22.14)	139 (8.72)
wayback.vefsafn.is	1,589	1,138 (71.62)	451 (28.38)	99 (6.23)
webcitation.org	1,585	988 (62.97)	587 (37.03)	179 (11.29)
arquivo.pt	1,569	1,563 (99.62)	6 (0.38)	857 (54.62)
web.archive.org	1,566	1,447 (92.40)	119 (7.60)	288 (18.39)
archive.is	1,396	1,379 (98.78)	17 (1.22)	0 (0)
wayback.archive-it.org	1,383	1,383 (100)	0 (0)	216 (15.62)
swap.stanford.edu	1,222	1,021 (83.55)	201 (16.45)	308 (25.20)
nationalarchives.gov.uk	994	986 (99.20)	8 (0.8)	243 (24.45)
europarchive.org	979	979 (100)	0 (0)	0 (0)
webharvest.gov	712	712 (100)	0 (0)	123 (17.27)
veebiarhiiv.digar.ee	488	310 (63.52)	178 (36.48)	94 (19.26)
webarchive.proni.gov.uk	469	469 (100)	0 (0)	119 (25.37)
collectionscanada.gc.ca	351	351 (100)	0 (0)	0 (0)
webarchive.org.uk	349	349 (100)	0 (0)	5 (1.43)
archive.bibalex.org	199	199 (100)	0 (0)	0 (0)
perma-archives.org	182	182 (100)	0 (0)	0 (0)
Total	**16,627**	**14,707 (88.45)**	**1,920 (11.55)**	**2,670 (16.06)**

pages were relatively static [85], and recent pages are more interactive and more difficult to archive [86, 177]. In general, the less often JavaScript is used in mementos to dynamically add content, the more chances we get repeatable hashes.

6.4 QUANTIFYING THE TYPES OF CHANGES

We compared consecutive hashes for each memento. Each time any two consecutive hash values are different, we identify one and only one type of change causing the different hashes. In other words, we assign at most one type of change even though other categories might apply. For example, if we detected that the set of resources comprising a memento at time t_0 varies from the set at t_1, then this change is marked as *Set*, and other categories of changes are not considered, because if sets are different, it implies that hashes will be also different. Similarly, if sets are identical, but there are some differences in the resources' HTTP status codes, then we assign the type of change *Status*, and no other types are considered. The order in which we look for changes is as following:

1. *Set*
2. *Status*

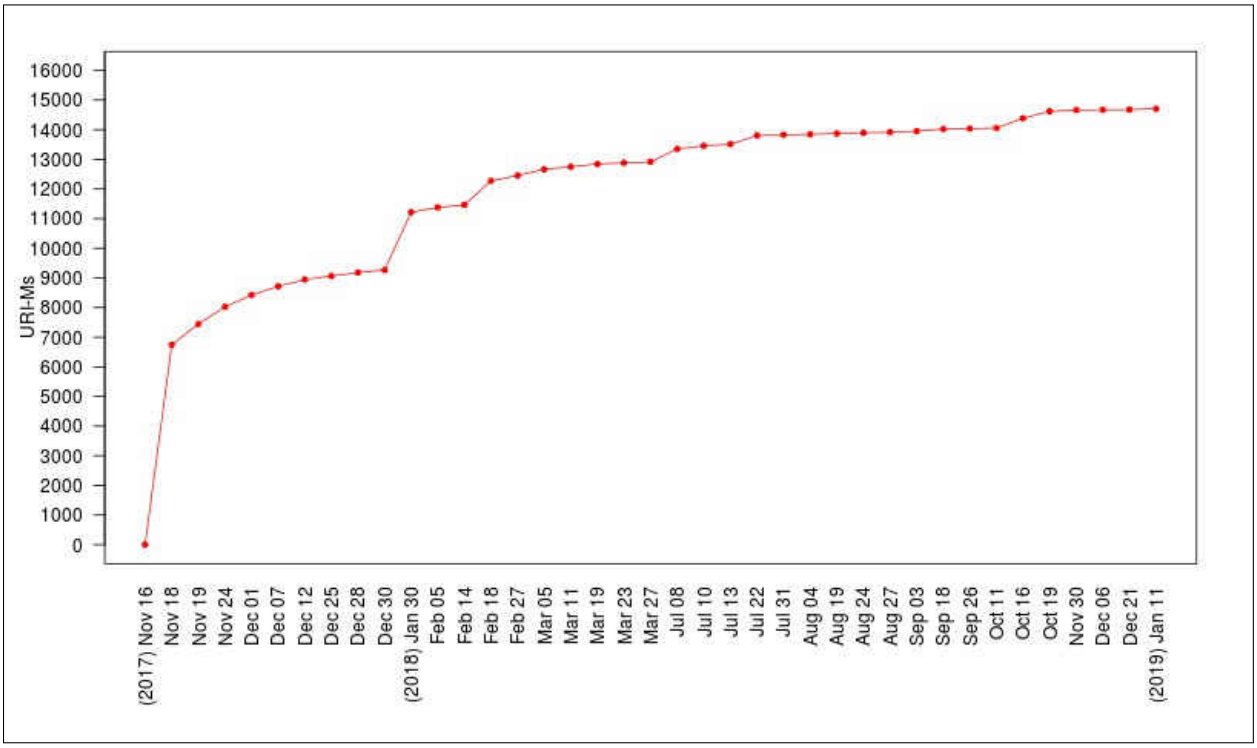


Fig. 112: The number of mementos which have at least two hashes increases over time.

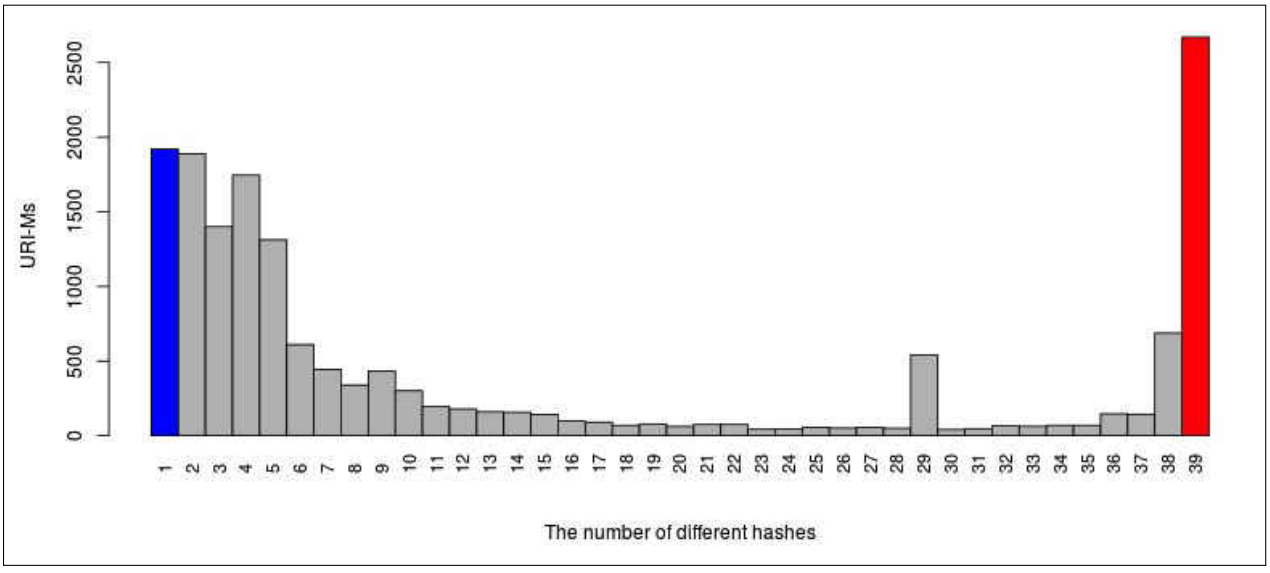


Fig. 113: The distribution of all 16,627 mementos for distinct number of hash values (blue=11.55% (1,920 mementos), red=16.06% (2,670 mementos)).

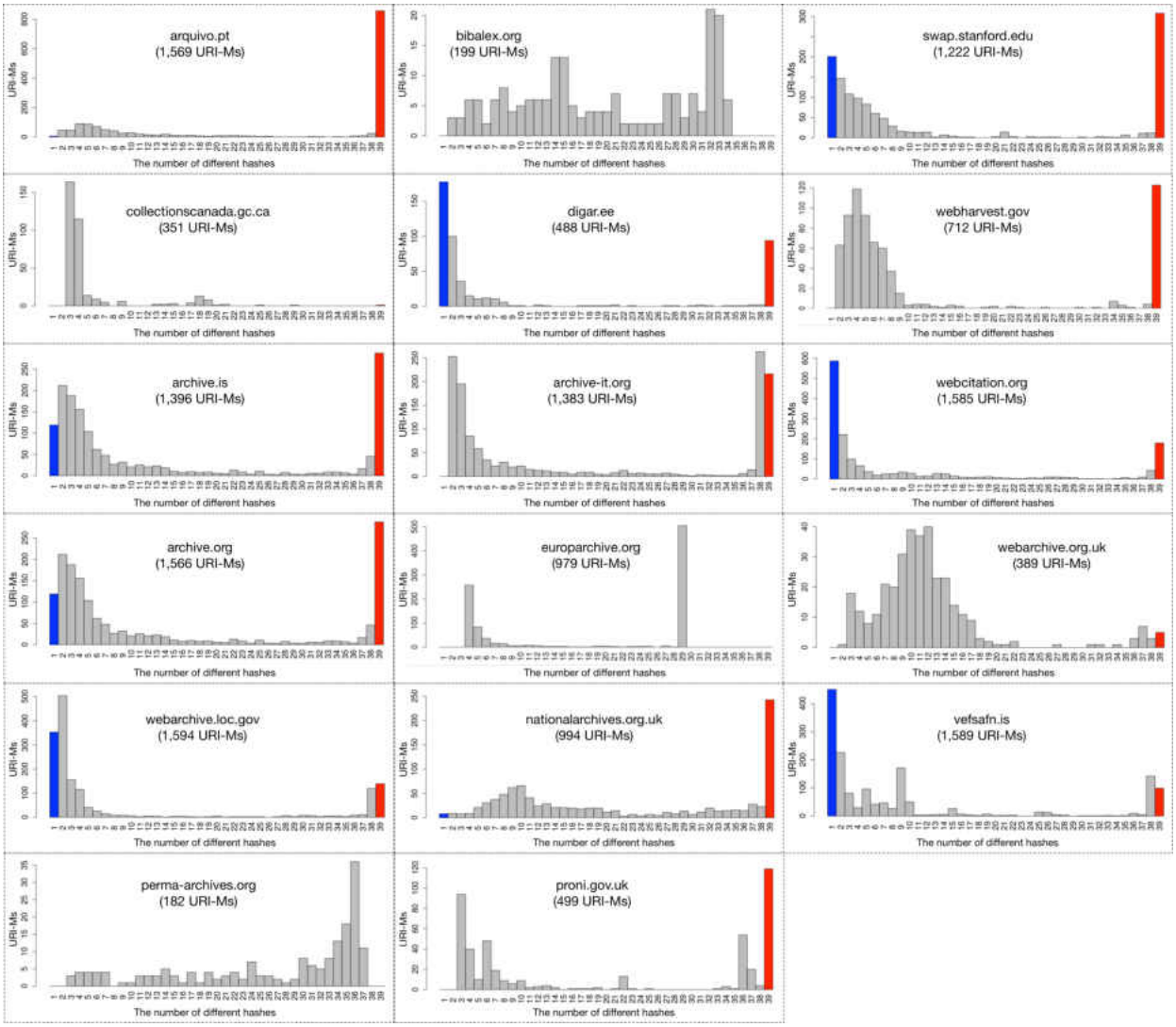


Fig. 114: The distribution of mementos per archive for distinct number of hash values.

3. *URI-M* or *Headers*

4. *Representation*

5. *URI-M* and *Representation*

Figure 115 illustrates the types of changes affecting mementos for each download by archive. A large number of mementos from each archive are producing different hashes because of the *Set* change (marked in red), which is mainly caused by dynamic/random resources generated after executing JavaScript. The *Representation* change (marked in yellow) is shown in multiple archives. For example, in `archive.is`, when comparing downloads 10

vs. 11, 11 vs. 12, and 13 vs. 14, we found that the archive refers to itself differently in the index.html file using `archive.li`, `archive.is`, and `archive.today` at downloads 11, 12, and 14, respectively, causing different hashes. The archives `vefsafn.is`, `webcitation.org`, and `perma-archives.org` are also tagged with the *Representation* change because of various reasons, such as (1) returning a rewritten page with HTTP `200` by the archive for raw mementos' requests, (2) computing hashes on rewritten content when raw mementos are not provided by the archive, and (3) altering content by a third-party service (Cloudflare) to prevent spam.

Figure 116 shows the types of changes affecting all mementos in each download. The figure indicates that on average only about one-third of the mementos have changes when comparing consecutive downloads. Download 11 (the first download in 2018) has the maximum number of mementos with changes, 7,557 (about 45% of the mementos).

In some cases, we can use the pattern of the hash values to infer the type of change that might have occurred to cause the hash value to change. For instance, consider the pattern of hash values `X` and `Y` for a URI-M:

```
X X X X X X X X X X X X X X X X X X X X
X X X X X X X X Y X X X X X X X X X X
```

In this example, there are 38 instances of `X` interrupted by one instance of a different hash value `Y`. In this case, it is likely that the `Y` value was the result of a transient error, since the hash values returned to `X` afterwards.

Another case, shown below with hash values of `W` and `Z`, may be the result of JavaScript execution:

```
Z Z W Z W Z W W Z Z Z W Z W Z Z Z Z W W
W W Z W Z Z Z W W W Z W Z Z Z Z W Z W
```

The pattern alternates between `W` and `Z` seemingly without a regular pattern. This is similar to the example from Figure 90 where the executed JavaScript randomly selects one of two resources to include in the composite memento. Both hashes `W` and `Z` are considered to be valid hashes for the memento.

A third example with hash values of `V` and `T` demonstrates a suspected update of the replay system in an archive, similar to Figures 93 and 94:

```
V V V V V V V V T T T T T T T T T T T T
T T T T T T T T T T T T T T T T T T T T
```

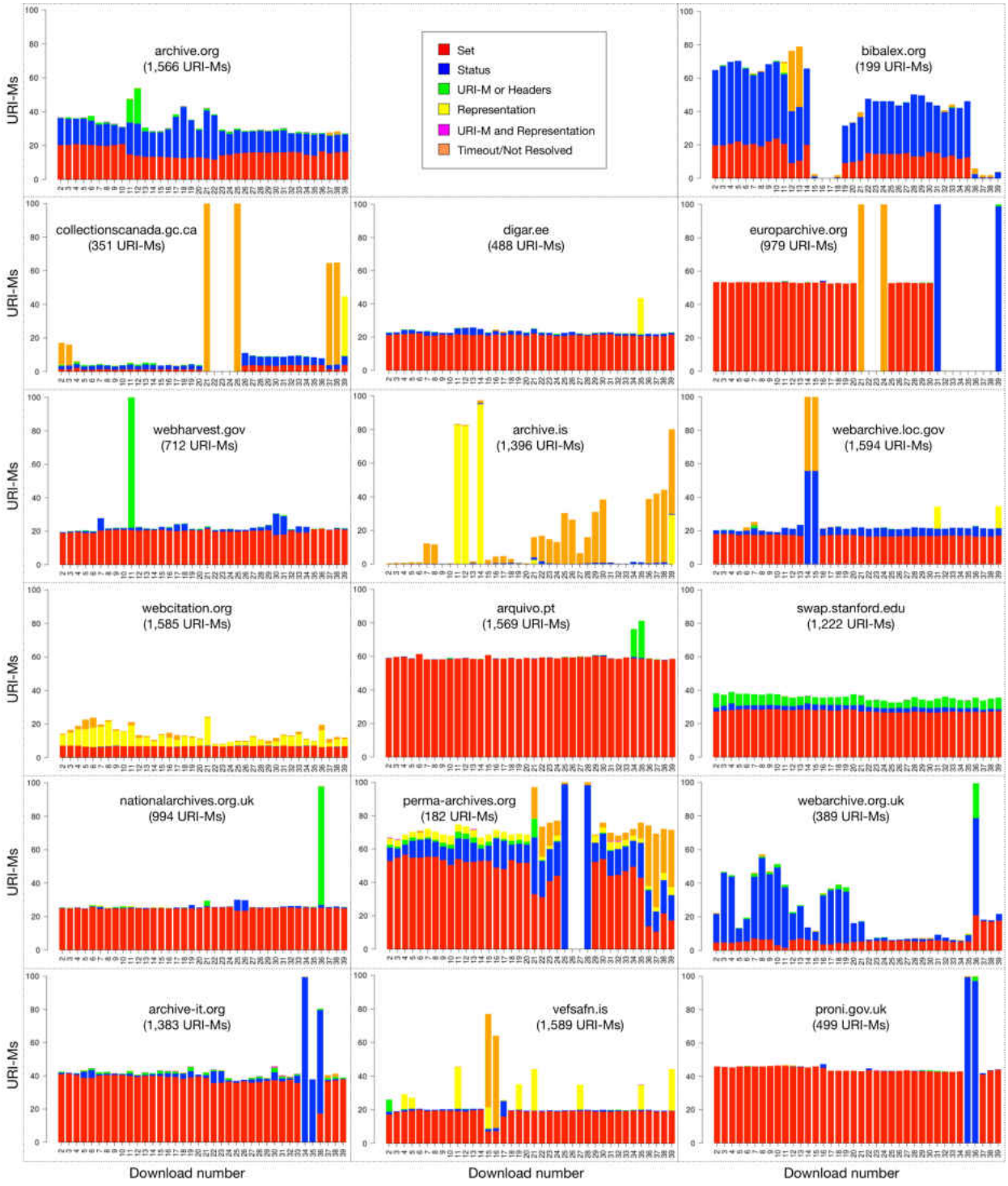


Fig. 115: The types of changes affecting mementos for each download by archive.

After the hash value `T` begins consistently appearing, the hash value `V` would no longer be considered valid. Further exploration of the pattern of hash values could help develop a

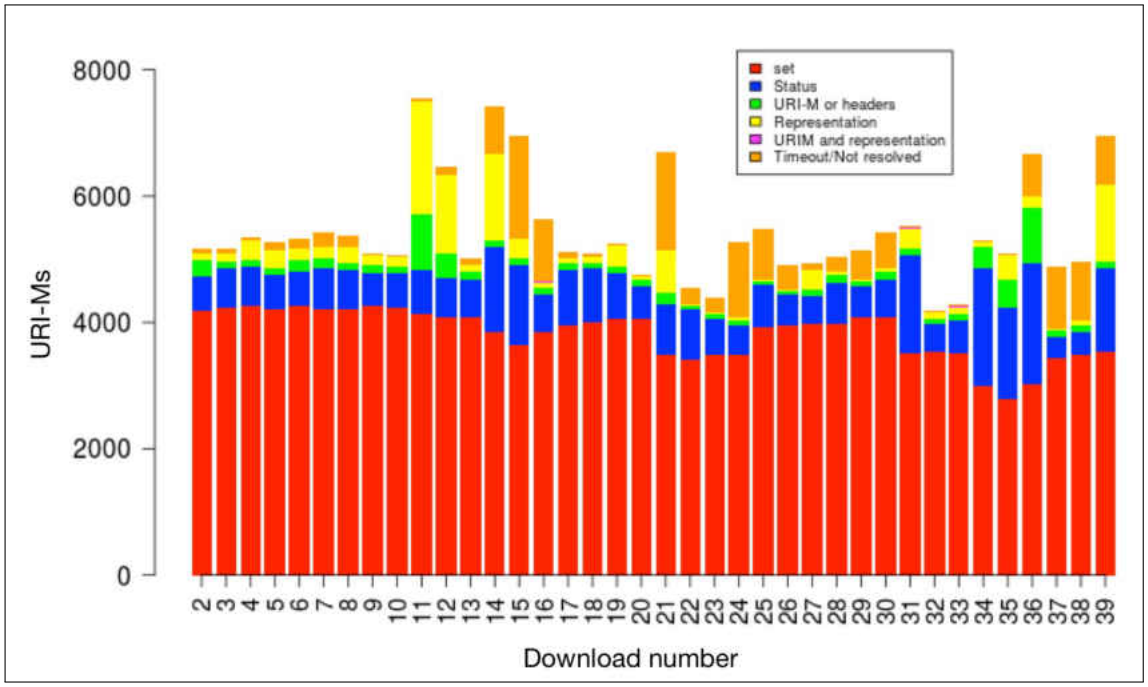


Fig. 116: The types of changes affecting all mementos for each download.

model for the probability of rendering the same composite memento over time (i.e., probability of computing the same hash value).

6.5 MIGRATED AND MISSING MEMENTOS

Web archives are established with the objective of providing permanent access to mementos. However, we shows five cases illustrated in Figure 117 where mementos became permanently unavailable in their original archives. Unfortunately, in these cases, the original archives from which the mementos (URI-Ms) have moved did not leave a machine readable method of locating the corresponding URI-Ms in the new archives (e.g., using `HTTP 301 Moved Permanently`). However, we were able to manually discover the five new archives to which the mementos have moved. For instance, the memento

`http://www.collectionscanada.gc.ca/webarchives/20051228174058/http://nationalatlas.gov/`

is now available at

http://webarchive.bac-lac.gc.ca:8080/wayback/20051228174058/http://nationalatlas.gov/

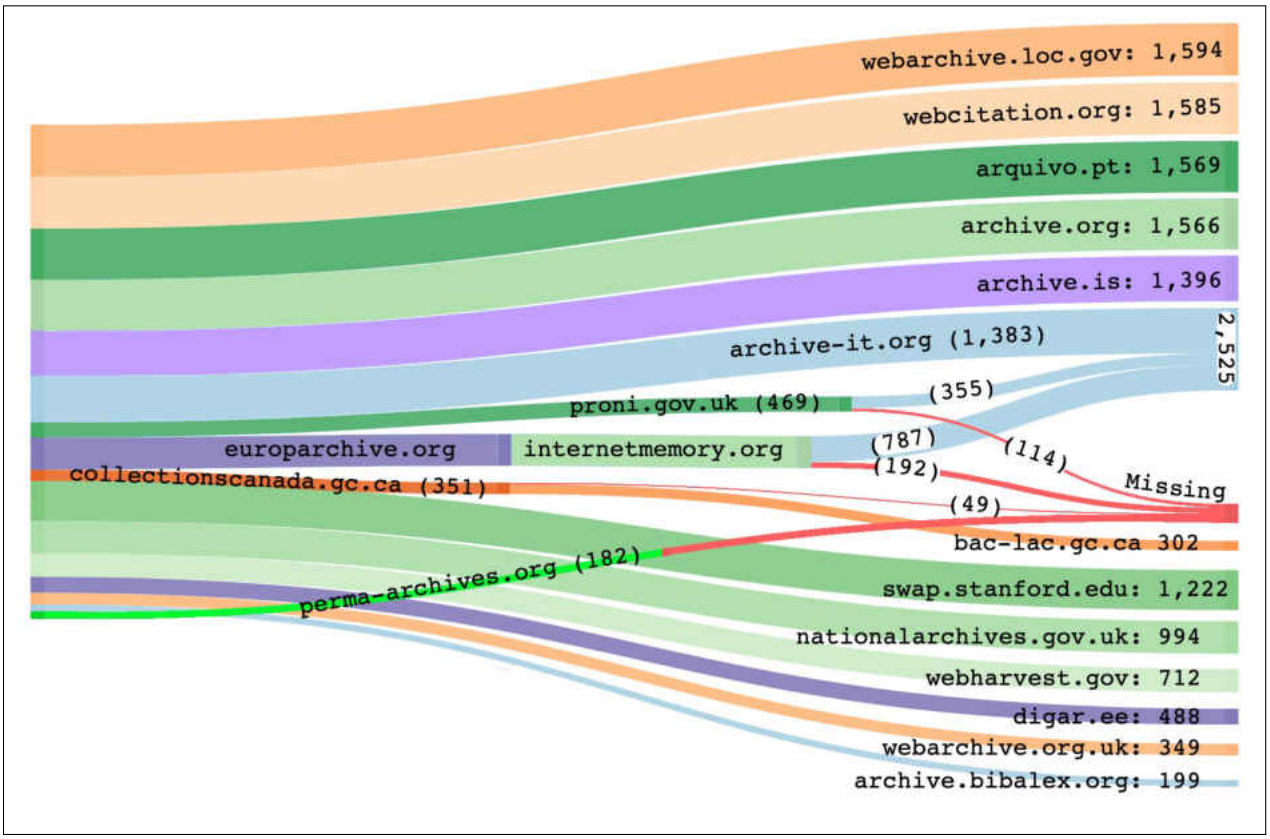


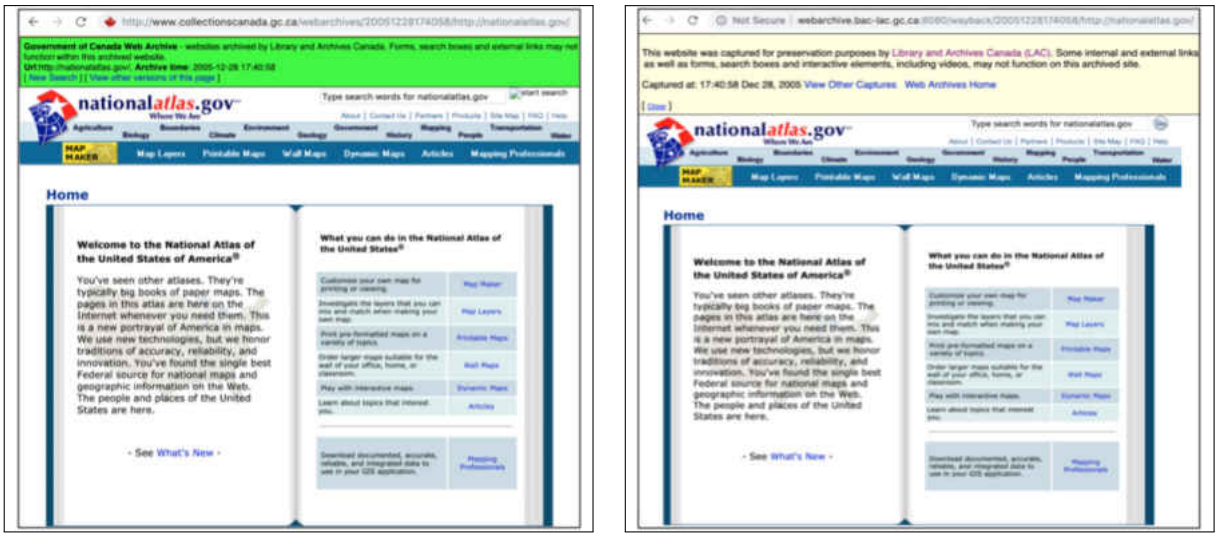
Fig. 117: The number of mementos per archive. It shows the number of migrated and missing mementos from four archive.

As shown in Figure 118, the representations of both mementos are almost the same except the color of the banner used by each archive.

Table 15 shows the estimated migration date and the status of each original archive. The HTTP requests to the homepage URIs of four original archives return `200 OK` either directly or after following redirects. The HTTP request to the fifth original archive (`internetmemory.org`) does not return any HTTP response as shown in Figure 119 (i.e., not resolving or timeout error). Table 15 shows also the number of missing mementos after each memento migration. A memento is considered "missing" if the values of its `Memento-Datetime`, URI-R, or HTTP status code in the original archive do not match the corresponding values in the new archive.

All mementos have been moved from the archives `collectionscanada.gc.ca`, `webarchive.proni.gov.uk`, and `http://perma-archives.org` [179, 182–184]. Generally, mementos

in Perma.cc were accessible through two different domains, `perma.cc` and `perma-archives.org`. Around February 2020, this archive no longer serves mementos under `perma-archives.org` [175]. Mementos from the National Library of Ireland (NLI) collection (`https://www.nli.ie`) have been moved from `europarchive.org` to `internetmemory.org` before moving to `archive-it.org`. We are not certain if other archived collections used to be available in both archives (`europarchive.org` and `internetmemory.org` also moved to `archive-it.org` or to other web archives). Table 16 shows the number of mementos grouped based on whether the values of `Memento-Datetime`, URI-R, and HTTP status code have been changed or not. We present three examples of missing mementos.



(a) In `www.collectionscanada.gc.ca`. (b) In `webarchive.bac-lac.gc.ca`.

Fig. 118: The representation (a) of a memento from the original archive and the representation (b) of its corresponding memento from the new archive.

```
1 $ date
2 Tue May 21 08:03:51 EDT 2019
3
4 $ curl http://www.internetmemory.org
5 curl: (7) Failed to connect to www.internetmemory.org port 80:
      Operation timed out
```

Fig. 119: After mementos migrated from `http://internetmemory.org`, the HTTP requests to the archive return the `timeout error`.

The first missing memento example shows a change in the HTTP status code. The memento

TABLE 15: About 27% (537 out of 1981) of mementos are missing from the four archives that transitioned in our 14-month study.

Migration Estimated Date	URI-Ms	Moved From (Original Archive)	Moved To (New Archive)	The Status of the Original Archive	Missing Mementos
May 2018	351	www.collectionscanada .gc.ca/webarchives/	webarchive.bac-lac .gc.ca:8080/wayback/	Redirects to the new archive's homepage	49
May 2018	979	collection.europarchive. org/nli/	collections.internet memory.org/nli/	Online but not providing any archiving services	0
Sep 2018	979	collections.internet memory.org/nli/	wayback.archive-it .org/10702/	Unreachable (timeout error)	192
Oct 2018	469	webarchive.proni .gov.uk/	wayback.archive-it .org/11112/	The homepage provides a list of URI-Rs. Clicking on any will redirect to archive-it.org	114
Feb 2020	182	perma-archives.org	perma.cc	Redirects to the new archive's homepage	182
Total	**1,981**				**537**

`http://collections.internetmemory.org/nli/20121223031837/http://www2008.org/`

is from 2012 and was available with `200 OK` in the original archive (`internetmemory.org`). We replayed the memento from our WARC files in 2018. The representation of the memento is shown in Figure 120. The corresponding memento

`http://wayback.archive-it.org/10702/20121223031837/http://www2008.org/`

is not available in `archive-it.org` (i.e., returning `404 Not Found`). Thus, we consider this memento as missing because it was `200` in the original archive and returns `404` in the new archive.

The second missing memento example shows a change in both the `Memento-Datetime` and the URI-R. The request to the URI-M

`http://www.collectionscanada.gc.ca/webarchives/20060304001905/http://www.dhs.gov/`

redirected (i.e., `302`) to the URI-M

`http://www.collectionscanada.gc.ca/webarchives/20060304001905/http://www.dhs.gov/dhspublic/`

TABLE 16: The number of mementos based on what has changed. The number of missing mementos are shown in bold.

Original archive → New archive	Same Memento-Datetimes?	Same URI-Rs?	Same status code?	URI-Ms
collectionscanada.gc.ca → bac-lac.gc.ca	Yes	Yes	Yes	302
	No	Yes	Yes	**28**
	No	No	Yes	**18**
	No	No	No	**2**
	No	Yes	No	**1**
europarchive.org/NLI → internetmemory.org	Yes	Yes	Yes	979
internetmemory.org/NLI → archive-it.org	Yes	Yes	Yes	787
	No	Yes	Yes	**184**
	No	No	Yes	**5**
	Yes	No	No	**2**
	Yes	Yes	No	**1**
proni.gov.uk → archive-it.org	Yes	Yes	Yes	355
	No	Yes	Yes	**106**
	No	No	Yes	**6**
	Yes	Yes	No	**2**
perma-archives.org → perma.cc	No	Yes	Yes	**164**
	No	No	No	**18**

which once again redirected to the following URI-M that ended up with `200 OK`:

`http://www.collectionscanada.gc.ca/webarchives/20060304001907/http://www.dhs.gov/dhspublic/`

The request to the corresponding URI-M

`http://webarchive.bac-lac.gc.ca:8080/wayback/20060304001905/http://www.dhs.gov/`

resulted in `302 Redirect` to the URI-M

`http://webarchive.bac-lac.gc.ca:8080/wayback/20140304192946/http://www.dhs.gov/`

which is `200 OK`. Even though the HTTP status codes of the corresponding mementos are identical, we consider the memento

`http://www.collectionscanada.gc.ca/webarchives/20060304001905/http://www.dhs.gov/`

as missing because the corresponding memento can not be retrieved from the new archive with the same `Memento-Datetime` and URI-R. The `Memento-Datetime` of the final

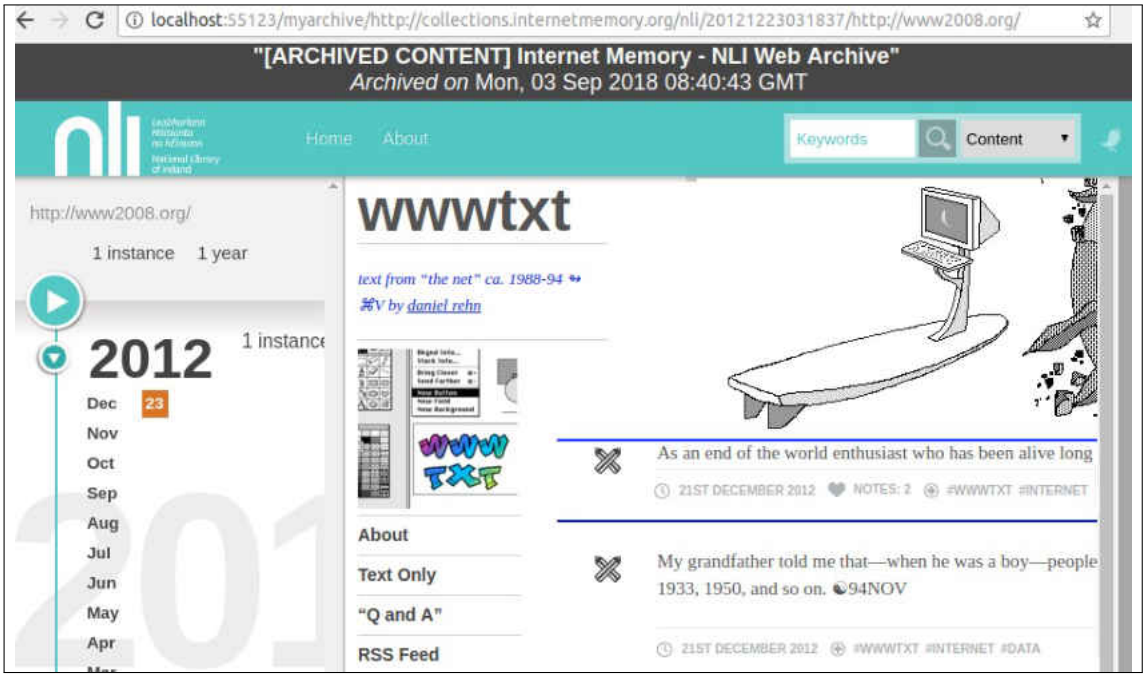


Fig. 120: The representation of the memento `http://collections.internetmemory.org/nli/20121223031837/http://www2008.org/`. It is from 2012, and it was available (`200 OK`) in the archive `internetmemory.org` between May 2018 and August 2018. We replayed this memento from our WARC files.

URI-M (after following redirects) from the original archive is March 04, 2006 00:19:07 GMT while it is March 04, 2014 19:29:46 GMT for the final URI-M from the new archive (for a delta of 8 years). The URI-R of the final URI-M from the original archive is `http://www.dhs.gov/dhspublic/` while it is `http://www.dhs.gov/` in the new archive.

The third example shows a set of 18 URI-Ms (below) from the original domain (`perma-archives.org`). These 18 mementos can not be found in the new domain (`perma.cc`). For example, `perma.cc` has no mementos for the URI-R `www.consumer.ftc.gov` as shown in Figure 121, while the same URI-R has at least one memento available under the original domain as illustrated in Figure 122.

- `http://perma-archives.org/warc/20170731024959/https://www.tmall.com/`
- `http://perma-archives.org/warc/20170203233757/https://www.cms.gov/cciio/index.html`
- `http://perma-archives.org/warc/20141021173834/http://databank.worldbank.org/data/home.aspx`
- `http://perma-archives.org/warc/20151101192051/http://www.homedepot.com/`
- `http://perma-archives.org/warc/20161118165253/https://studentloans.gov/myDirectL`

oan/index.action

- http://perma-archives.org/warc/20151009235521/http://readwrite.com/
- http://perma-archives.org/warc/20160914003340/https://www.texas.gov/
- http://perma-archives.org/warc/20170220021805/https://www.sam.gov/portal/SAM/
- http://perma-archives.org/warc/20151101193342/http://www.spiegel.de/
- http://perma-archives.org/warc/20150404172754/https://www.att.com/
- http://perma-archives.org/warc/20150820154813/http://ici.radio-canada.ca/
- http://perma-archives.org/warc/20170925035325/https://www.dropbox.com/?landing=dbv2
- http://perma-archives.org/warc/20161105175946/http://www.hostgator.com/
- http://perma-archives.org/warc/20150123144122/http://www.usa.gov/page-not-found
- http://perma-archives.org/warc/20151002194849/http://eric.ed.gov/
- http://perma-archives.org/warc/20160204202605/https://www.icloud.com/
- http://perma-archives.org/warc/20171003185440/https://travel.state.gov/content/travel/en/404.html
- http://perma-archives.org/warc/20171022230759/https://www.consumer.ftc.gov/

```
$ curl -i https://perma.cc/timemap/link/https://www.consumer.ftc.gov/

HTTP/2 404
date: Wed, 13 May 2020 02:36:33 GMT
content-type: text/html; charset=utf-8
x-memento-count: 0
server: cloudflare

404 page not found
```

Fig. 121: The new Perma archive does not have any mementos for the original page **www.consumer.ftc.gov**.

Figures 123, 124, 125, and 126 depict the time difference between the `Memento-Datetime` of final URI-Ms from the original archives (represented by the centered line at 0) and the `Memento-Datetime` of the corresponding URI-Ms from the new archives. If the `Memento-Datetime` of a memento in the new archive is older than the memento's `Memento-Datetime` in the original archive, then the memento is placed on left side, otherwise it is placed on the right. The memento migration from `europarchive.org` to `internetmemory.org` is not included in the figures because it did not result in missing mementos.

As illustrated in Figures 124, 125 and 126, most of the differences in time between the `Memento-Datetime` from the original archives and new archives are less than one minute. Unexpectedly, we do not find any memento in `perma.cc` that has the same `Memento-Datetime` compared to their corresponding mementos in the original domain (`perma-archives.org`).

```
WARC/1.0
WARC-Type: response
WARC-Target-URI: https://perma-archives.org/warc/20171022230759/https
    ://www.consumer.ftc.gov/
WARC-Date: 2018-10-20T16:20:39Z
WARC-Record-ID: <urn:uuid:15b9f480-d484-11e8-ab64-d71ba9b59307>
Content-Type: application/http; msgtype=response
Content-Length: 31067

HTTP/1.1 200 OK
x-archive-orig-x-generator: Drupal 7 (http://drupal.org)
status: 200
memento-datetime: Sun, 22 Oct 2017 23:07:59 GMT
content-type: text/html; charset=utf-8
link: <https://perma-archives.org/warc/20171022230759/https://www.
    consumer.ftc.gov/>; rel="memento"; datetime="Sun, 22 Oct 2017
    23:07:59 GMT", <https://www.consumer.ftc.gov/>; rel="original", <
    https://perma-archives.org/warc/timemap/*/https://www.consumer.ftc.
    gov/>; rel="timemap"; type="application/link-format", <https://perma
    -archives.org/warc/https://www.consumer.ftc.gov/>; rel="timegate"
expires: Sat, 20 Oct 2018 20:19:26 GMT
date: Sat, 20 Oct 2018 16:19:26 GMT
...
```

Fig. 122: The original Perma archive has at least one memento for the original page **www.consumer.ftc.gov**.

This might be related to how indexing of WARC files is performed by the new archives. Figures 127 and 128 show an example of a memento where the difference between the value of its `Memento-Datetime` and the value of the `Memento-Datetime` from the new archive is only three seconds. The HTTP request to the URI-M

`http://webarchive.proni.gov.uk/20170701153654/http://www.fda.gov/`

returned `302 Redirect` to the URI-M

`http://webarchive.proni.gov.uk/20170701153654/https://fda.gov/`

The corresponding URI-M

`http://wayback.archive-it.org/11112/20170701153654/http://www.fda.gov/`

from the new archive returned `302 Redirect` to the URI-M

`http://wayback.archive-it.org/11112/20170701153657/https://www.fda.gov/`

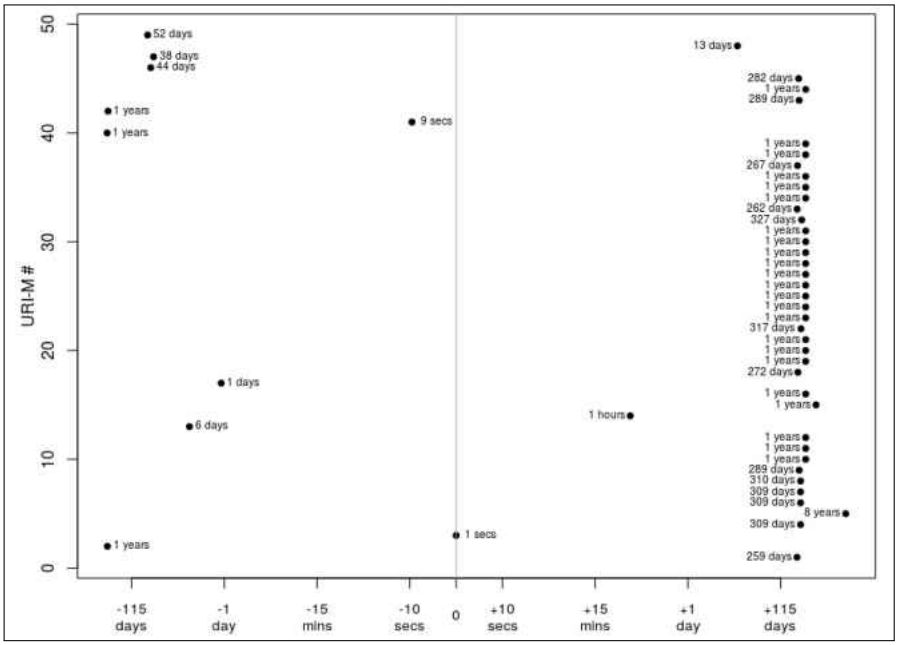


Fig. 123: Mementos migrated from `collectionscanada.gc.ca` to `webarchive.bac-lac.gc.ca`.

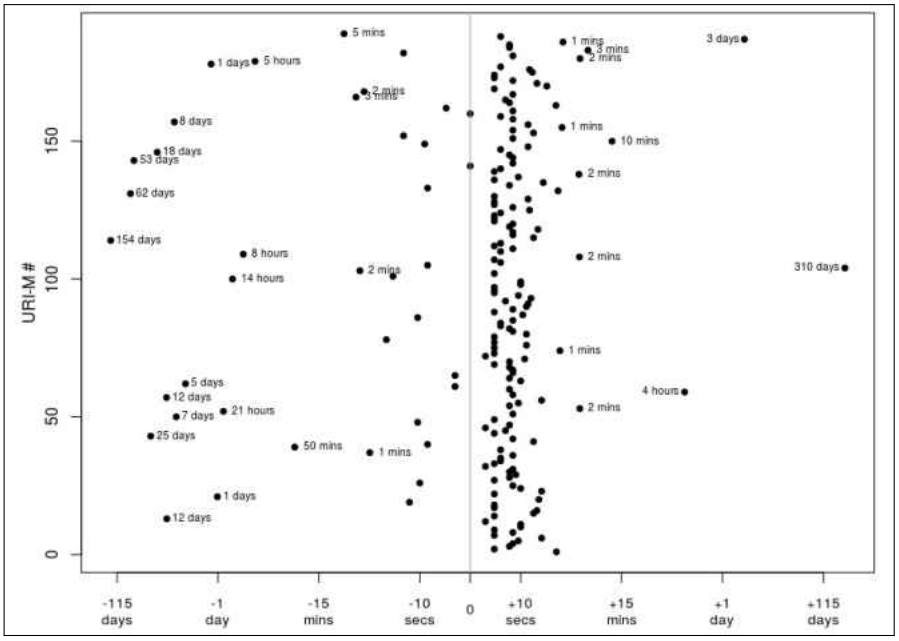


Fig. 124: Mementos migrated from `internetmemory.org` to `archive-it.org`.

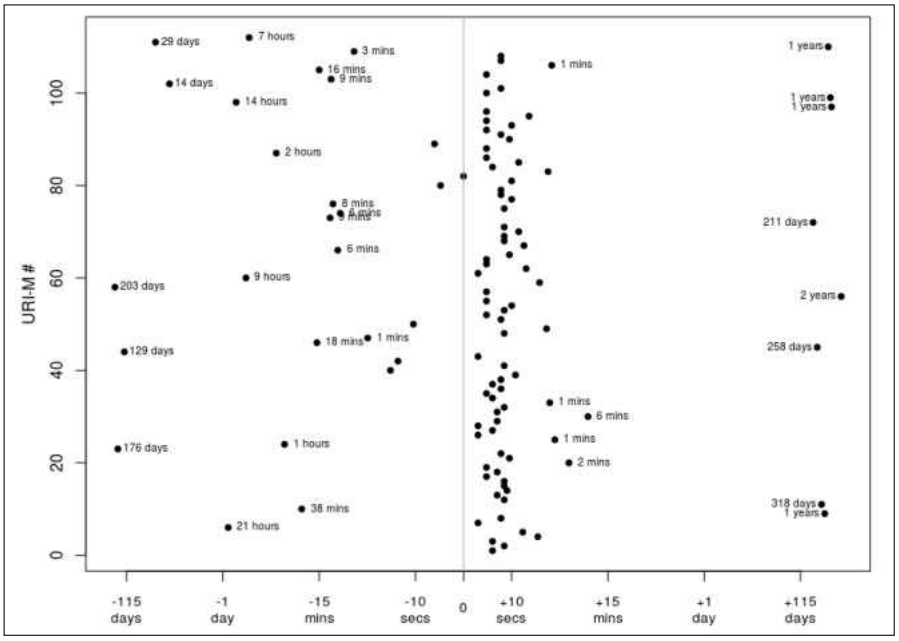


Fig. 125: Mementos migrated from `proni.gov.uk` to `archive-it.org`.

There are three seconds difference in the `Memento-Datetime`s of the final URI-Ms.

New archives might use an updated version of a replay tool (e.g., OpenWayback) that enables new features that were not supported by the original archives. For example, the new archive (`webarchive.bac-lac.gc.ca`) uses an updated version of OpenWayback that allows accessing raw mementos and supports the Memento Framework. These features were not available in the original archive `collectionscanada.gc.ca`.

It is possible that web archives permanently remove or prevent access to specific archived content for security or copyright issues [185, 186]. However, our research shows cases where mementos become unreachable (i.e., migrate) from their original archives for unknown reasons to us. In these cases, archives do not point to the corresponding URI-Ms in the new archives or respond with the appropriate HTTP status codes that explain why users no longer have access to certain collections. To mitigate this issue, the original archives, when possible, can take one of the following actions:

- **Respond with `301 Moved Permanently`**: Instead of responding with `404 Not Found` or `302 moved temporarily` to the homepage of a new archive, archives can respond with `301` to the corresponding URI-M in the new archive. For example, the End of Term Archive (`eot.us.archive.org`) was established with the goal of preserving the United States government web (with the Top Level Domain `.gov`). The domain name

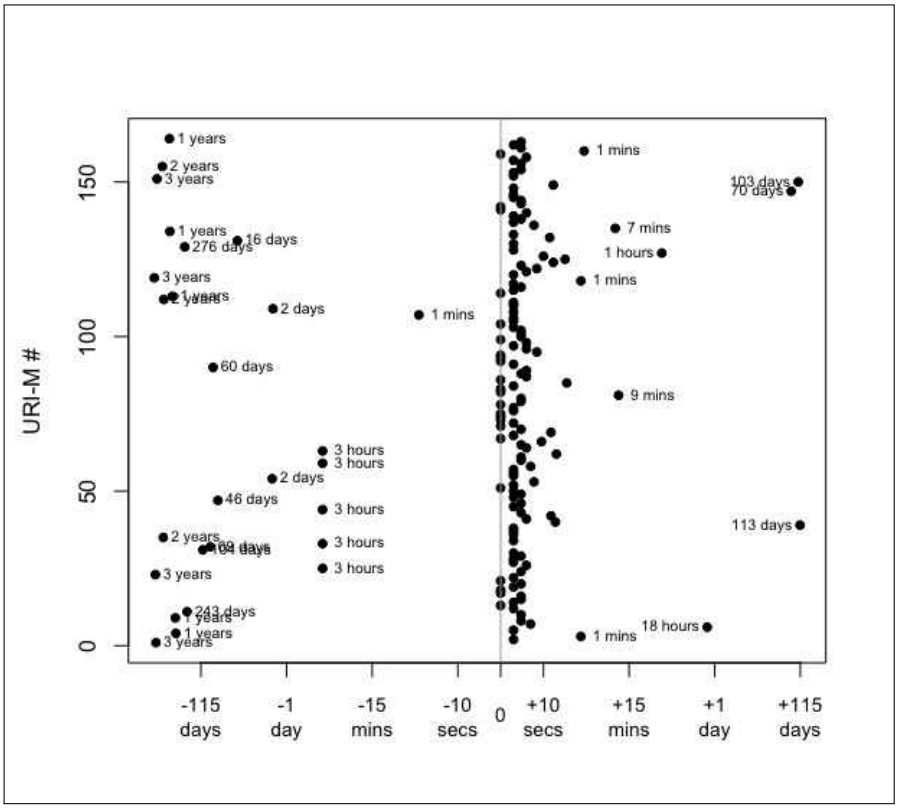


Fig. 126: Mementos migrated from `perma-archives.org` to `perma.cc`.

`eot.us.archive.org` is still under the control of the Internet Archive. Figure 129 shows an example of how the HTTP request to a URI-M in the End of Term Archive redirects to the corresponding URI-M in the Internet Archive. This practice maintains link integrity via "follow-your-nose" [187] from the old URI-M to the new URI-M.

Technically, enabling `301 Moved Permanently` in a server can be achieved through simple configuration steps. For example, for Apache web server [188], the *mod_rewrite* rule [189] can be used to perform automatic redirects and rewrite URI-Ms at the request time. The rewrite rule example shown in Figure 130 can be used by the original archive `collectionscanada.gc.ca` to redirect HTTP requests to the new archive (`bac-lac.gc.ca`).

- **Respond with the appropriate HTTP status code**: If original archives determine not to point to the new archives, they can respond with the HTTP status code `410 Gone` to indicate that an archived resource used to exist in the archive but has been permanently (and intentionally) deleted instead of responding with `404 Not Found`

```
...
WARC-Type: response
WARC-Target-URI: http://webarchive.proni.gov.uk/20170701153654/http://
    www.fda.gov/
WARC-Date: 2018-10-11T10:14:43Z
...
HTTP/1.1 302 FOUND
Location: http://webarchive.proni.gov.uk/20170701153654/https://fda.gov
    /
Memento-Datetime: Sat, 01 Jul 2017 15:36:54 GMT
...
WARC-Type: response
WARC-Target-URI: http://webarchive.proni.gov.uk/20170701153654/https://
    fda.gov/
...
HTTP/1.1 200 OK
...
```

Fig. 127: Requesting a memento from the original archive, the archive responded with `302 Redirect` to a URI-M that has the same `Memento-Datetime` but with lexigraphically different URI-R.

```
...
WARC-Type: response
WARC-Target-URI: @http://wayback.archive-it.org/11112/20170701153654/
    http://www.fda.gov/@
WARC-Date: 2019-01-21T05:44:57Z
...
HTTP/1.1 302 Found
Location: ~/11112/20170701153657/https://www.fda.gov/~
...
WARC-Type: response
WARC-Target-URI: ~http://wayback.archive-it.org/11112/20170701153657/
    https://www.fda.gov/~
WARC-Date: 2019-01-21T05:44:57Z
...
HTTP/1.1 200 OK
...
```

Fig. 128: Requesting the memento (from Figure 127) from the new archive, the archive responded with `302 Redirect` to a URI-M that has a different `Memento-Datetime` (three seconds difference) but with lexigraphically different URI-R.

or `503 Service Unavailable`.

```
$ curl --head --location --silent http://eot.us.archive.org/eot
    /20120520120841/http://www2.ed.gov/espanol/parents/academic/
    matematicas/brochure.pdf | egrep -i "(HTTP/|^location:)"

HTTP/1.1 302 Found
Location: https://web.archive.org/web/20120520120841/http://www2.ed.gov
    /espanol/parents/academic/matematicas/brochure.pdf
HTTP/2 200
```

Fig. 129: The HTTP requests to URI-Ms in the End of Term Archive ideally redirect to the corresponding URI-Ms in the Internet Archive.

```
# With mod_rewrite
RewriteEngine on
RewriteRule   "^/webarchives/(\d{14})/(.+)" http://webarchive.bac-lac.
    gc.ca:8080/wayback/$1/$2  [L,R=301]
```

Fig. 130: HTTP requests to URI-Ms in `collectionscanada.gc.ca` are redirected by an Apache rewrite rule to the corresponding URI-Ms in the new archive `bac-lac.gc.ca`.

6.6 ARCHIVE-LEVEL CHANGES

Figure 131 shows how each web archive behaved based on comparing consecutive downloads of mementos. Each cell represents the percentage of mementos with changes at a particular download compared with the previous download (white indicates no change, while dark blue indicates a large percentage of mementos have changed). For example, the mementos of the NLI collection in `europarchive.org` became unreachable in download 21. We discovered the new location starting from download 24 (i.e., `internetmemory.org` presented in the next row). Then, these NLI mementos in `internetmemory.org` became inaccessible, returning `403 Error`, at download 31. We manually discovered the new location, `archive-it.org`, of these disappeared mementos starting from download 39.

The heatmap in Figure 131 can be used to visualize the overall performance of each archive and to identify points in time where major changes occur. For instance, looking at `webarchive.org.uk`, we noticed that the performance of the archive before download 20 is totally different from the performance after download 20. From the WARC files, we knew that the archive had upgraded the replay service to a new version of PyWb—the archive began supporting the option `mp_` for loading the archived content into an iframe. In

general, we should be cautious in interpreting the heatmap because multiple consecutive cells with similar colors do not always imply better performance. For example, there are some periods where all mementos from particular archives were not available, like mementos from `perma-archives.org` in downloads 25, 26, and 27 (i.e., returning `500 Internal Server Error` between July 31, 2018 and August 19, 2018), so the white cells indicate no changes in mementos, but they were actually unavailable during those three downloads. We added annotations to the heatmap to explain some major archive-level changes.

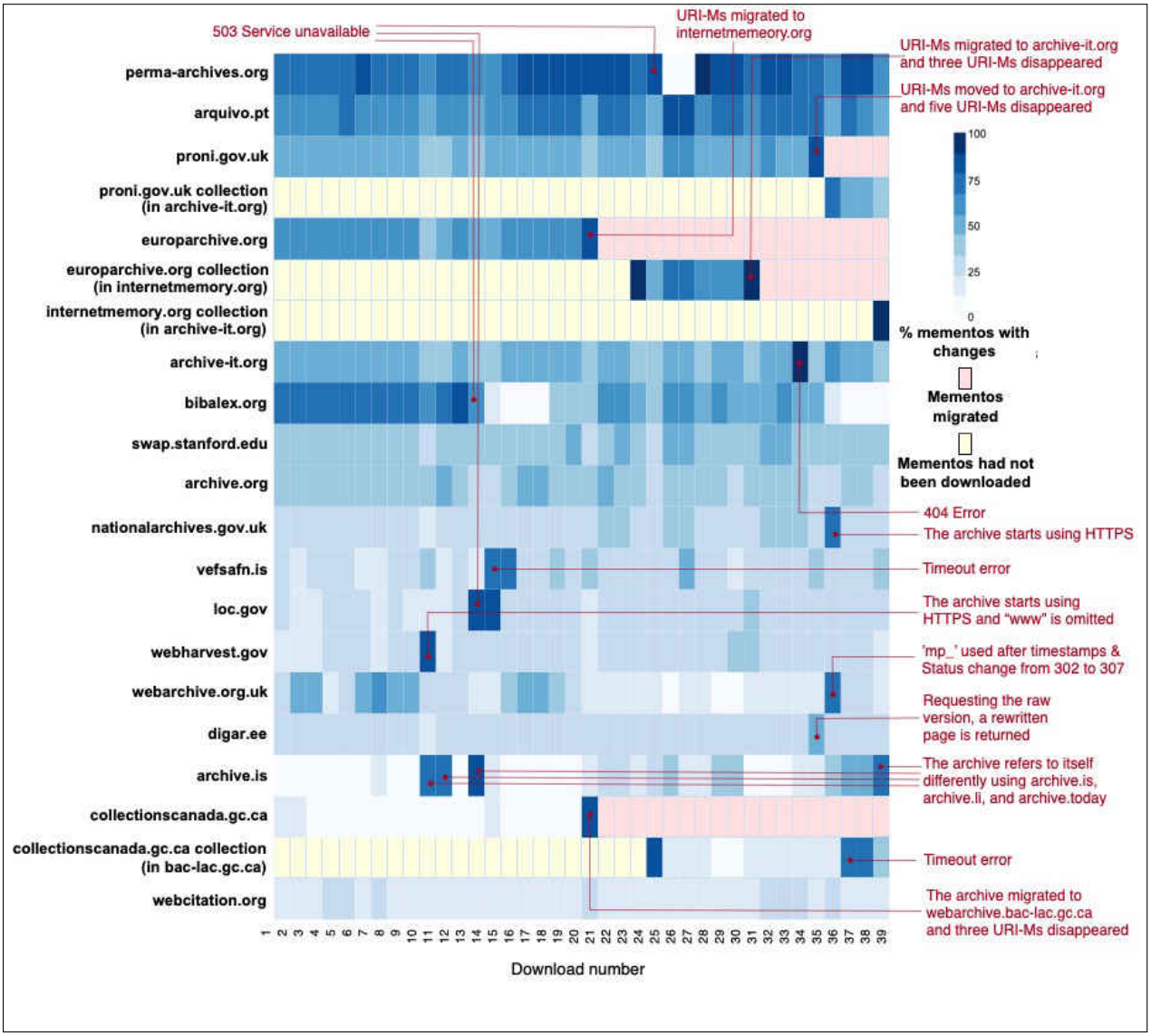


Fig. 131: An annotated heatmap to show archive-level changes when comparing consecutive downloads. Light blue=no (few) mementos with changes compared to a previous download, and dark blue=most of the mementos have one or more changes.

6.7 URI-M-BASED AND ENTITY-BASED HASHING

Our study (Chapter 6.3) shows that about 88.45% of mementos produce at least two different aggregated hash values. One of the reasons why we get different hash values is that we use multiple components in hash calculation including URI-Ms, HTTP response headers and entity bodies, and HTTP status codes. Any small change (e.g., JavaScript creates URI-Ms with random values) affecting one or more of these components comprising a composite memento will result in different hash values.

In this section, we introduce two additional techniques, URI-M-based and entity-based hashing techniques for generating hash values on composite mementos. We want to assess how these two hashing techniques can help produce repeatable hash values on composite mementos.

6.7.1 URI-M-BASED HASHING TECHNIQUE

In the URI-M-based hashing technique we only consider the URI-Ms of resources, comprising a composite memento, in hash calculation. Therefore, any changes in HTTP response headers/entities and status codes will not affect final hash values. The final hash values are generated using a Merkle tree (introduced in Chapter 6.1), so the only difference here is that we consider only the URI-M of each resource to generate root hashes on a composite memento. This technique is to address the question whether or not the set of URI-Ms of a composite memento has changed over time.

For example we downloaded the composite memento `https://perma.cc/R84B-LJWJ` at two different times on April 10, 2020 and April 24, 2020. In both downloads, there were 37 resources embedded in the composite memento. We generated two aggregated hashes values on the composite memento after each download. The first aggregated hash value was calculated based on URI-Ms, and we used URI-Ms, HTTP headers/entities, and status codes to generate the second aggregated hash value. We did not notice any change in the resulting URI-M-based hash value. It was `87e2c8b9f717b826ac8c7d716d05d07f21c8ce23c47409b79812637b3c0ac23d` in both downloads. The two other hash values were different (i.e., `086907181ba3cfe0c4478971f7bd9924363b9fcfe4923f5fc771a02955a624b8` on April 10th and `bde3cd7c25b22527e2d8a77c20181906c82fe5a058ea623417f0501d04856ef0`) because the HTTP entity of the based HTML file of the composite had a string generated randomly as a result of Cloudflare's Email Address Obfuscation (Chapter 6.2). This example indicates that there is no change in the set comprising the composite

memento over time even though the HTTP entity body of the base file has changed.

6.7.2 ENTITY-BASED HASHING TECHNIQUE

The entity-based hashing technique considers only the HTTP entity bodies of resources comprising a composite memento in the hash calculation. It answers the question whether or not a composite memento contains the same HTTP entity bodies over time of embedded resources with the `200 OK` status code. Thus, this technique concerns about observing the entity bodies of resources regardless of the locations (or URI-Ms) from which these resources are retrieved. We expect this technique to produce more repeatable hash values than other techniques because, in many cases, archival redirects (i.e., TimeMap changes) or changes in HTTP response headers will not affect the resulting hash values.

For example, we downloaded the memento

`https://web.archive.org/web/20160110010525/http://sharehouse-bonbon.com/`

two times on April 10, 2020. The resulting aggregated hash values calculated on only the HTTP entities were identical, while the aggregated hash values computed using only the URI-Ms were different. This is because the same HTTP entity of the embedded CSS resources *www-embed-player-vflsgofpf.css* was retrieved using different URI-Ms:

`https://web.archive.org/web/20160126155112cs_/https://s.ytimg.com/yts/cssbin/www-embed-player-vflsgofpf.css`

and

`https://web.archive.org/web/20160127012758cs_/https://s.ytimg.com/yts/cssbin/www-embed-player-vflsgofpf.css`

The example shows that we can still generate repeatable aggregated hash values on HTTP entity bodies of composite mementos even if their URI-Ms change over time.

6.7.3 WE ALWAYS DETECT NEW HASH VALUES AFTER EACH DOWNLOAD

We introduced three techniques for generating final hash values on composite mementos:

1. URI-M-based hashing (Chapter 6.7.1): Each resource in a composite memento is represented by only the hash of the resource's URI-M, `hash(URI-M)`

2. Entity-based hashing (Chapter 6.7.2): Each resource in a composite memento is represented by the hash of the resource's HTTP entity body, `hash(HTTP entity body)`

3. Hashing based on different components (Chapter 6.1): Each resource in a composite memento is represented by the hash value calculated on the HTTP response headers, entity body, URI-M, and status code of the resource, `hash(HTTP headers, HTTP entity body, URI-M, status code)`

We expect the entity-based hashing technique to produce more repeatable hash values than the other techniques. As shown in Chapter 6.7.2, changes in only URI-Ms, HTTP response headers, or HTTP status codes of any resource embedded in a composite memento will not affect the final hash value calculated by the entity-based hashing technique.

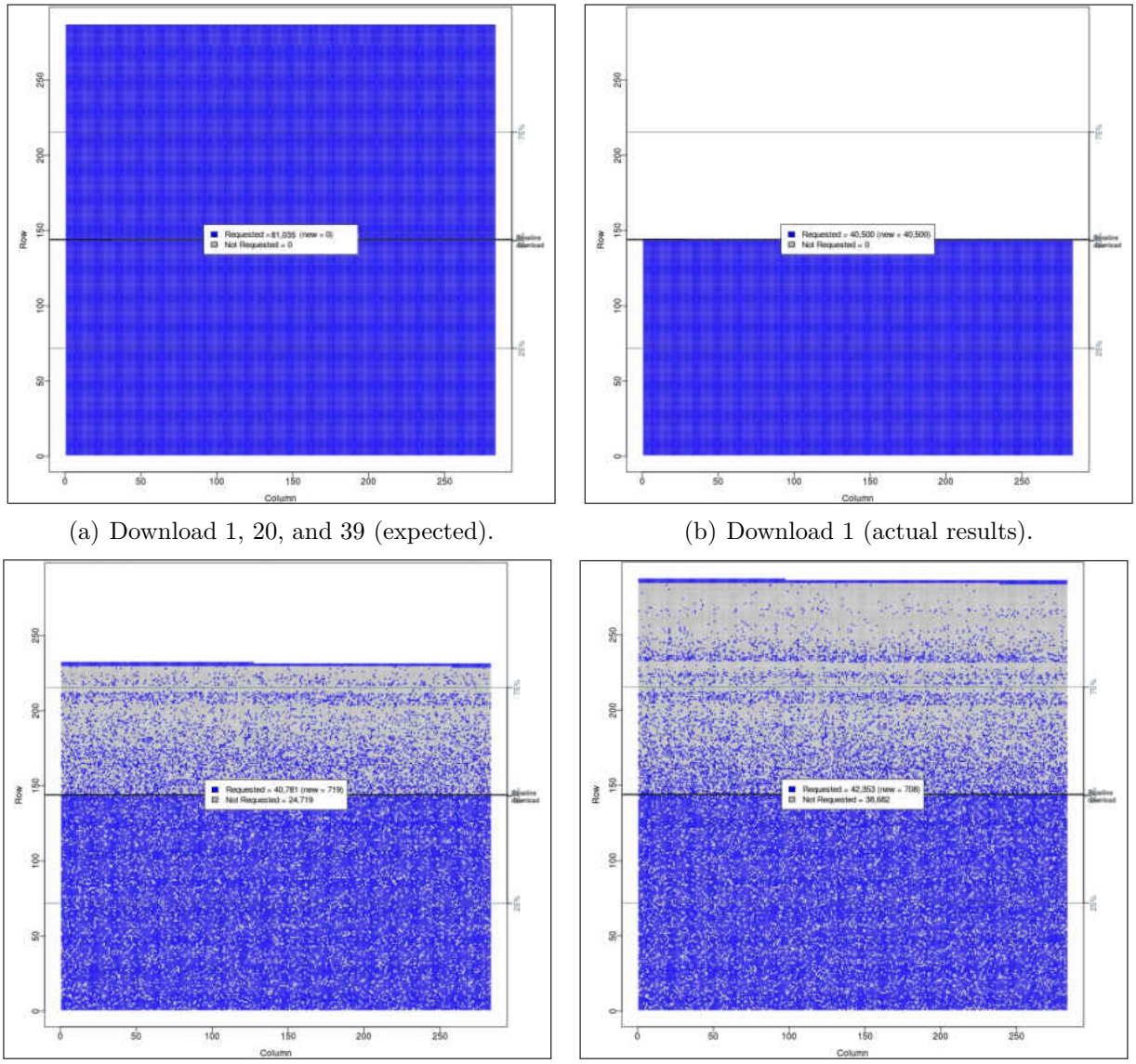
We compare the three techniques by generating several animated GIF files[1] for each archive to show the number of resources requested each time we download the set of mementos from the archive. The blue, red, green colors in Figures 133, 134, and 135 indicate that the URI-M-based, all components based, and entity-based hashing techniques are used, respectively. The unique resources (or their hashes) are represented by points/dots in the figures and stacked in rows, so each point represents one resource. The location of resources do not change regardless of the download number, and the number of unique resources increases over time (i.e., new resources are requested in each download). The only special meaning to the position `(x, y)` of a resource is that resources appear on the bottom of the figures (e.g., download 1) are requested before resources appear at the top (e.g., download 39).

For example, Figure 133 shows the number of URI-Ms requested in each download. Any red point indicates that the URI-M is requested, otherwise it is not requested (marked in grey). We have several observations from Figures 133, 134, and 135 (additional Figures for other archives are available in [190]):

- Ideally, the same number of resources should be requested in each download (e.g., illustrated in Figure 132(a)). However, our study indicates that new resources are requested on every download (e.g., illustrated in Figures 132(b), 132(c), and 132(d)).

- The number of requested URI-Ms in Download 1 (baseline download) in Figure 133 is only 40,500 compared to the total number of resources requested in downloads 1-39, which is 81,035 URI-Ms. Download 1 requested only 50% of the total number of

[1]The animated GIFs are available at `https://github.com/oduwsdl/mementos-fixity/tree/master/hashing_techniques`



(a) Download 1, 20, and 39 (expected).

(b) Download 1 (actual results).

(c) Download 20 (actual results).

(d) Download 39 (actual results).

Fig. 132: Resources (URI-Ms) requested in downloads 1, 20, and 39 from the Internet Archive. Blue = URI-M is requested, Gray = URI-M is not requested. Total URI-Ms requested in downloads 1-39 is 81,035. (a) represents the ideal case where the same resources are requested in every download, (b), (c), and (d) show the actual results.

resources seen by Download 39.

- Looking at the three figures, we always observe new resources after each new download. New resources or hash values are marked in purple rectangles. As we explained in Chapter 6.2, these new resources are generated or caused by JavaScript or archival redirects. Figures 136 shows the number of new resources or hash values detected

after each download from the Internet Archive.

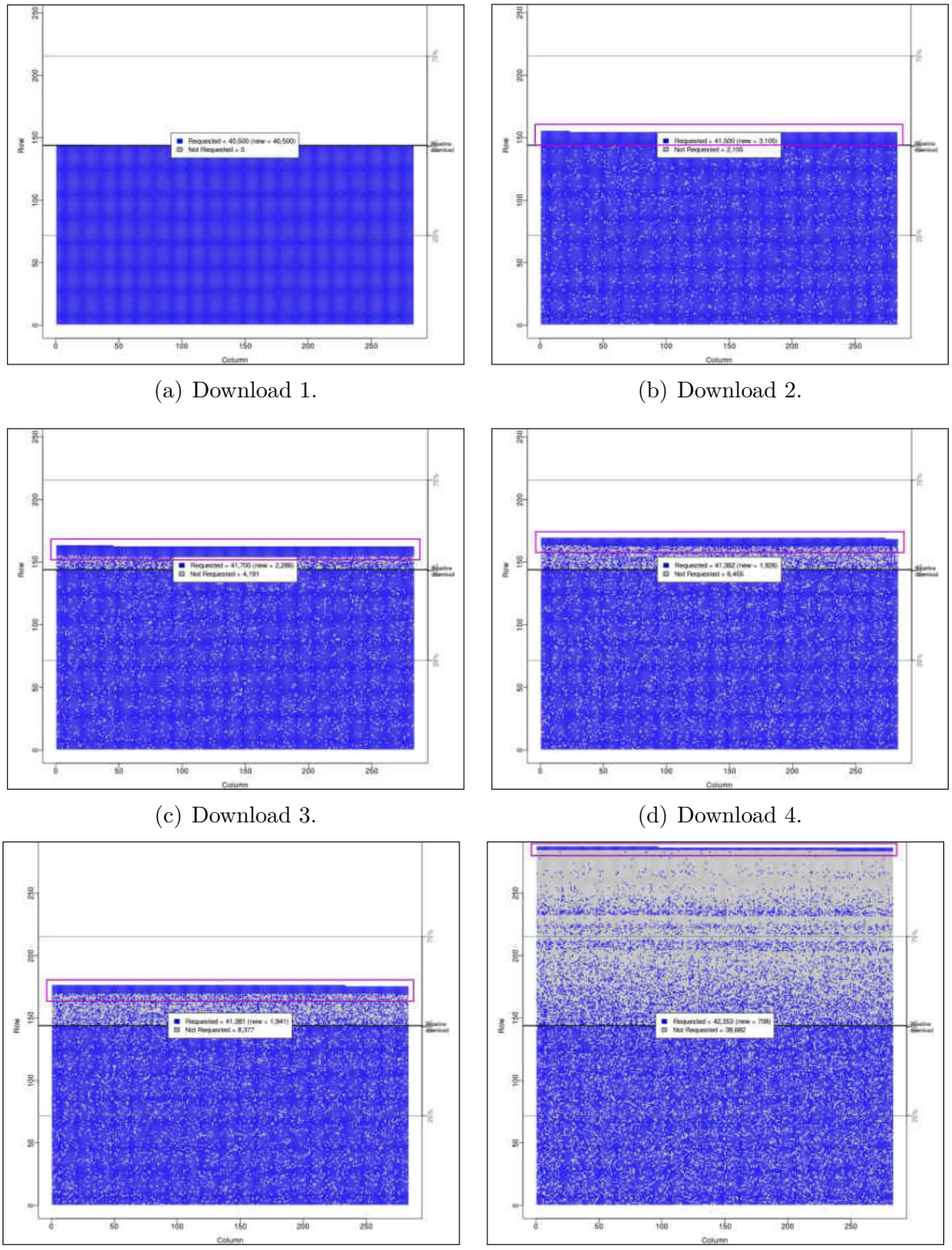
- The colored area (points with blue, red, or green color) under the baseline download line in Figures 133, 134, and 135 are denser than areas above the baseline download line. This indicates that resources requested in Download 1 have a higher chance of getting requested in the future than new resources requested in downloads 2-39. Table 17 shows that the probability of requesting resources again in future downloads depends on the download they were first requested at.

TABLE 17: The probability of requesting resources again in future downloads depends on the download they were first requested at. Only downloads 1-20 are shown.

Download number at which resources were first requested	**URIM-based hashing**	**Entity-based hashing**	**Hashing based on all components**
1	0.89	0.94	0.89
2	0.38	0.77	0.38
3	0.17	0.76	0.17
4	0.18	0.81	0.18
5	0.06	0.46	0.06
6	0.07	0.57	0.07
7	0.15	0.76	0.15
8	0	0.50	0
9	0	0.36	0
10	0	0.40	0
11	0	0.22	0
12	0	0.25	0.12
13	0	0	0
14	0	0.83	0
15	0	0.6	0
16	0	0.75	0
17	0	0	0
18	0	0.50	0
19	0	0	0
20	0	0	0

- Figure 135 demonstrates that the entity hashing technique produces fewer new hash values than the other techniques. For example, most of the HTTP entity bodies (about 88%, 30,203 entity bodies out of 34,202) appear in Download 1.

As shown in Chapter 6.5, mementos have been moved from the European Web Archive `europarchive.org` to `internetmemory.org` before moving to `archive-it.org`. Figure 137



(a) Download 1.

(b) Download 2.

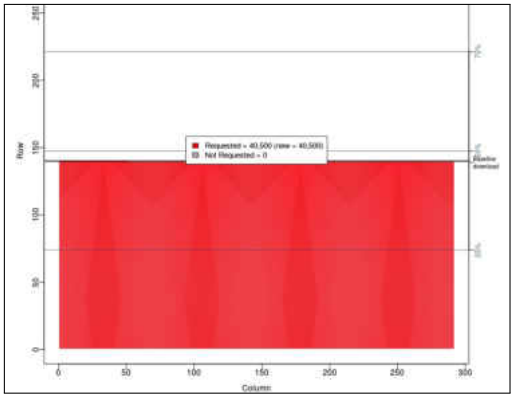
(c) Download 3.

(d) Download 4.

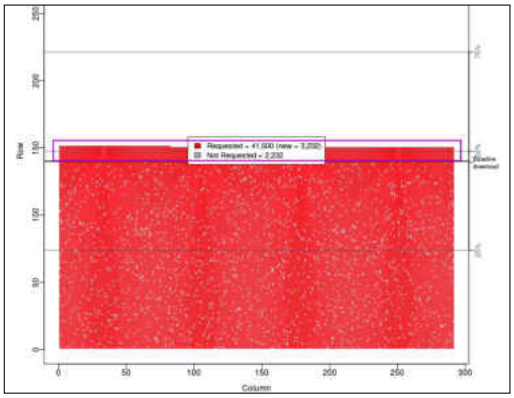
(e) Download 5.

(f) Download 39.

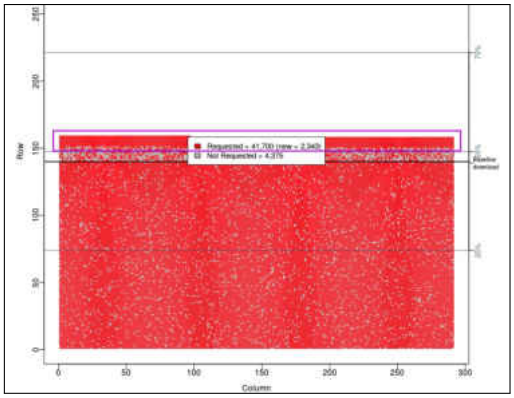
Fig. 133: Resources (URI-Ms) requested in the first five downloads and download 39 from the Internet Archive. Blue = URI-M is requested, Gray = URI-M is not requested. Total URI-Ms requested in downloads 1-39 is 81,035.



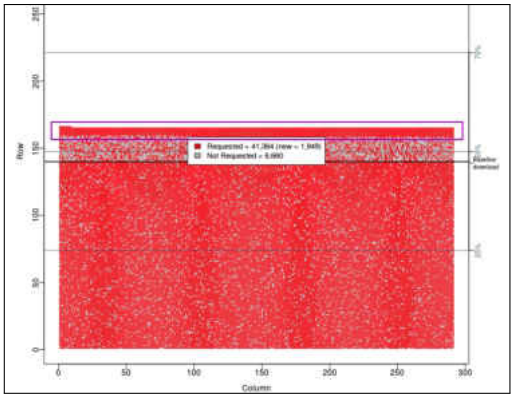
(a) Download 1.



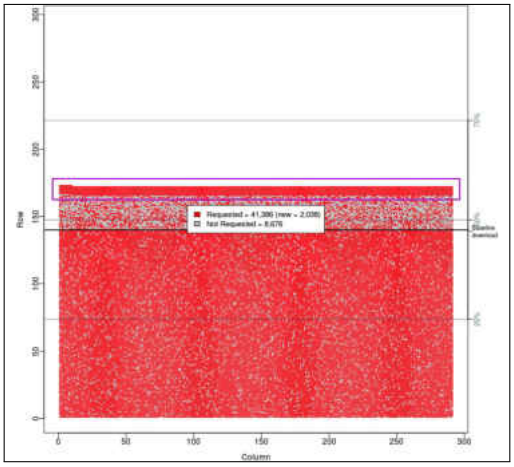
(b) Download 2.



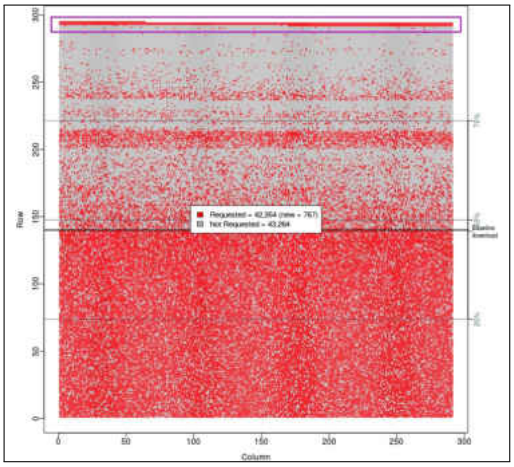
(c) Download 3.



(d) Download 4.

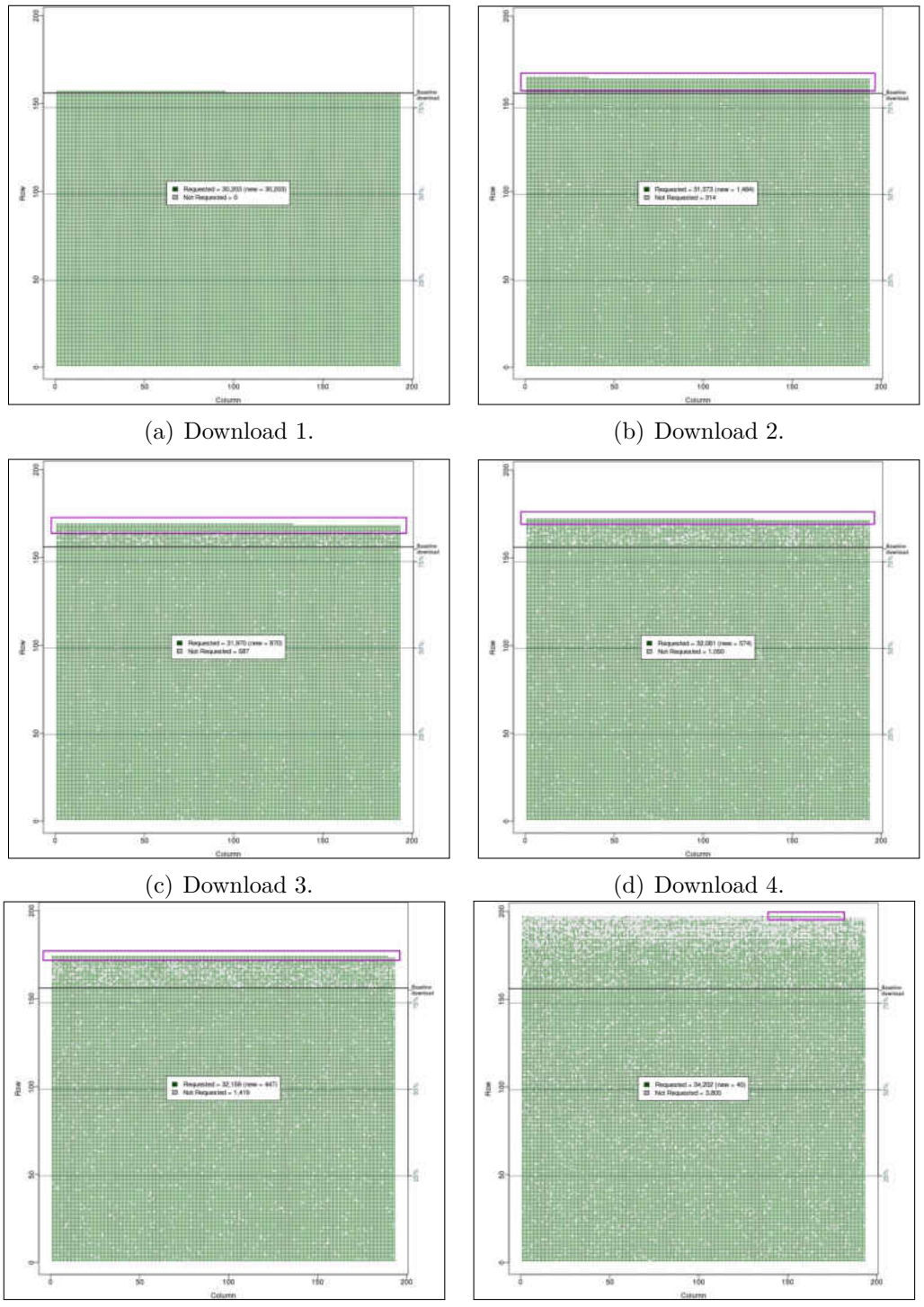


(e) Download 5.



(f) Download 39.

Fig. 134: Each point = `hash(HTTP response headers, HTTP entity body, HTTP status code, URI-M)`. Only downloads 1-5 and 39 are shown. We download 1,566 composite mementos from the Internet Archive in each download. Red = the hash value is observed, Gray = the hash value is not observed.



(a) Download 1.

(b) Download 2.

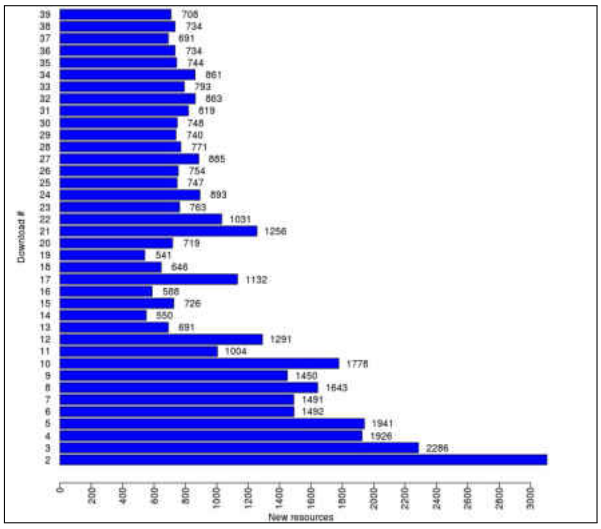
(c) Download 3.

(d) Download 4.

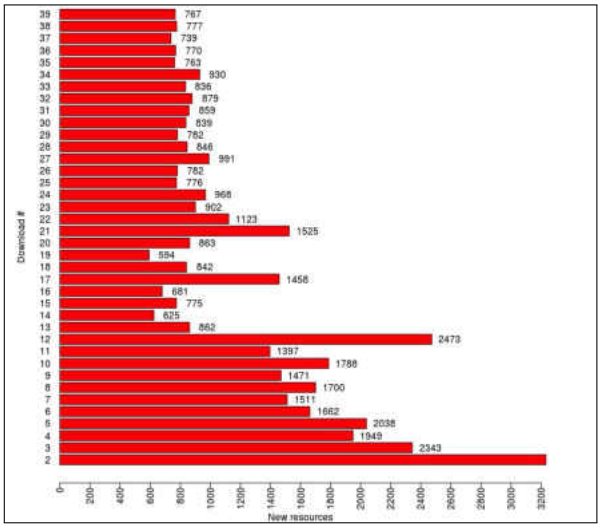
(e) Download 5.

(f) Download 39.

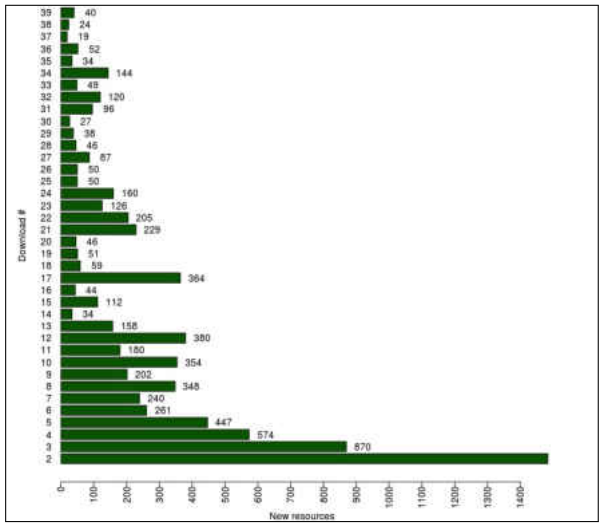
Fig. 135: Entities returned in the first five downloads and download 39 from the Internet Archive. Green = Entity is returned, Gray = Entity is not returned.



(a) URI-M-based hashing technique.



(b) Including the HTTP response headers and entity, status code, and URI-M in hash calculation.



(c) Entity-based hashing technique.

Fig. 136: New resources observed per download from the Internet Archive using the three techniques of hashing.

shows that entity-based hashing technique produces fewer new hash values compared to results from the URI-M-based hashing technique for mementos originally from `europarchive.org`. That is because most of the returned HTTP entity bodies from `europarchive.org` and `internetmemory.org` are identical, while their URI-Ms are different. Figure 137 (a, c, e) also shows that all URI-Ms requested between Download 1 and 23 have not been requested in Downloads 24 to 38 since the mementos moved to a new archive. Similarly, URI-Ms requested between Download 1 and 38 will not be requested after Download 39.

6.8 CHAPTER SUMMARY

In this chapter, we used our collected dataset of 16,627 mementos to conduct a study to identify and quantify the types of changes that cause different fixity information on composite mementos. For each memento, we have 39 hash values generated after downloading each memento 39 times over a time span of 14 months. We defined several types of changes that cause different hash values including, *Set*, *Status code*, *Headers*, *Representation*, *URI-M*, *URI-M and Representation*, and *Timeout/Not resolved*.

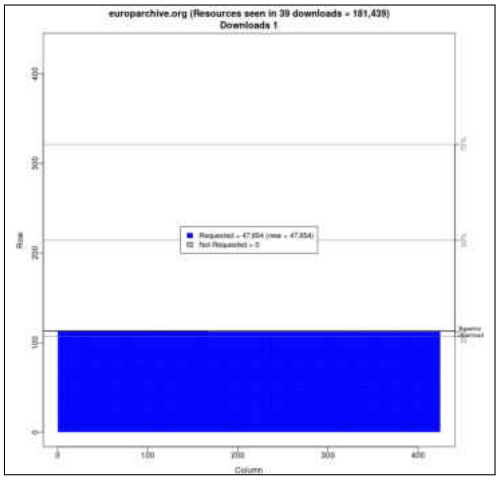
We found that 88.45% of mementos have at least two different hashes, including all mementos from seven archives. Only 11.55% of mementos always produce the same hash, while 16.06% of mementos always produce a different aggregated hash value after each replay. Factors like the availability of raw mementos, JavaScript, changes in TimeMaps, transient errors have great impact on the number of distinct hash values generated on each memento.

We found that three web archives had either changed their domain or moved to different web archives. Mementos from `collectionscanada.gc.ca` are now available at `webarchive.bac-lac.gc.ca` (raw mementos become available after this change), and mementos from `webarchive.proni.gov.uk` are now available at `archive-it.org`. Mementos that were served under `perma-archives.org` are now available at `perma.cc`. We also found that mementos of the National Library of Ireland (NLI) archived collection (`https://www.nli.ie`) have been moved from `europarchive.org` to `internetmemory.org` before moving to `archive-it.org`. In all of these memento migration cases, there are always some mementos that cannot be accessed in their new domain/archive. These mementos were available in the original domain.

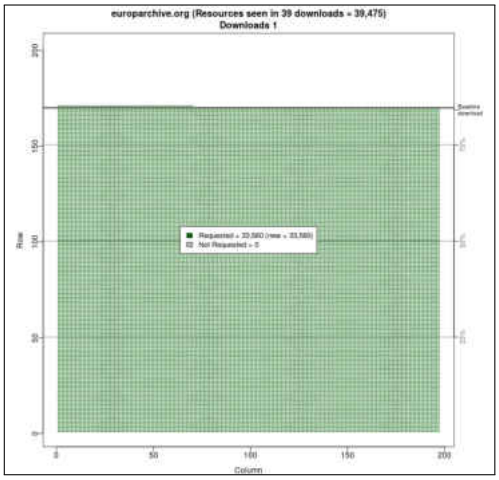
Some web archives have applied major changes over time. For example, `webcitation.org` no longer accepts any archival requests after July 9, 2019, and `perma.cc` no longer serves any memento whose creation is not initiated explicitly by users. Other archives have started using HTTPS rather than HTTP and/or the subdomain `www`.

We created a heatmap that allows us to detect archive-level changes. The heatmap can be used to visualize the overall performance of each archive and to identify points in time where major changes occur, For example, through the heatmap we were able to identify archives that recently start to use a new version of their replay system. We also introduced two additional hashing techniques, entity-based and URI-M-based, that allows us to generate multiple aggregated hash values on a composite memento. We found that there are some mementos that always produce new hash values regardless of the hashing techniques used

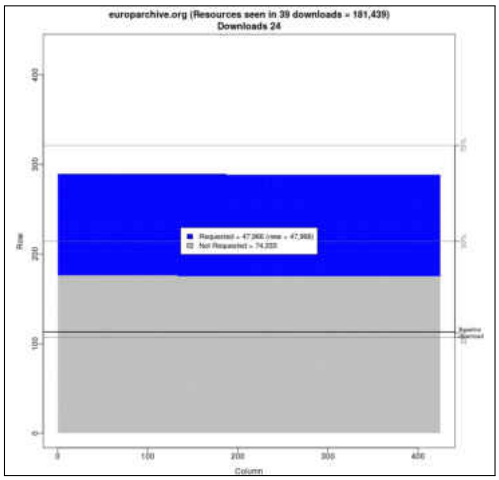
in the hash calculation.



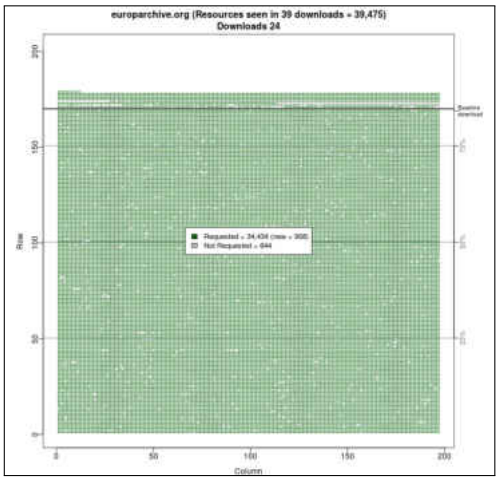
(a) Download 1 (URI-M-based hashing).



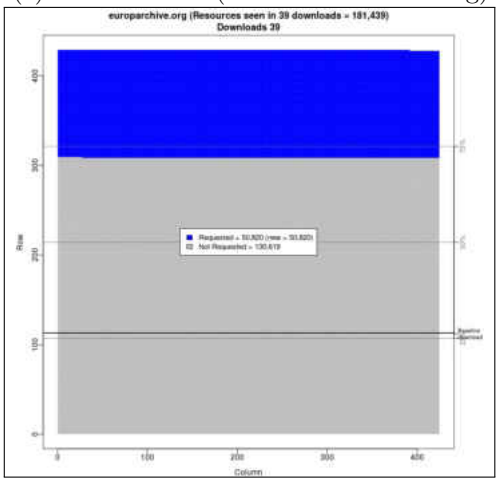
(b) Download 1 (entity-based hashing).



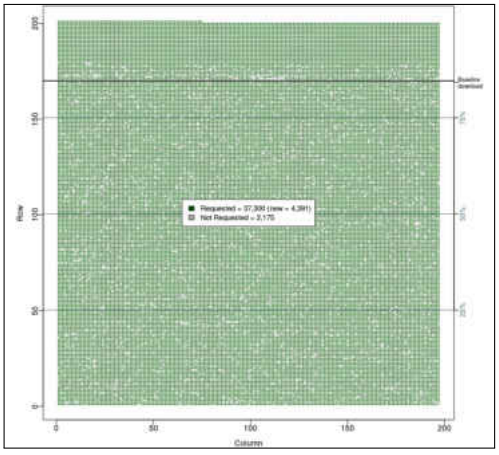
(c) Download 24 (URI-M-based hashing).



(d) Download 24 (entity-based hashing).



(e) Download 39 (URI-M-based hashing).



(f) Download 39 (entity-based hashing).

Fig. 137: Entity-based technique produces fewer new hash values compared to results from URI-M-based hashing technique for mementos from `europarchive.org`.

CHAPTER 7

ARCHIVE ASSISTED ARCHIVAL FIXITY VERIFICATION FRAMEWORK

This chapter explains our study toward addressing RQ3:

RQ3: How can we store and retrieve fixity information independently from the web archives from which the associated mementos are preserved?

In this chapter, we introduce two approaches, *Atomic* and *Block*, that can be used to disseminate fixity information to web archives and verify the fixity of mementos [191]. In the *Atomic* approach, the fixity information of each archived web page is stored in a separate 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 being disseminated to archives.

This work introduces a basic, yet extensible, format of fixity information in the form of a structured manifest file. However, the main contribution of this study focuses on the structure of the manifest file and the two suggested approaches of disseminating fixity information (or manifests) rather than strength, applicability, extension, scope, or security of the manifest. The framework describes how manifests are generated, published, discovered, and used to verify mementos. The proposed framework does not require any change in the infrastructure of web archives. It is built based on well-known standards, such as the Memento protocol, and works with current archives' APIs. The framework allows for the generation of manifests for selected resources instead of incurring the overhead of creating manifests for all archived resources.

We performed two evaluations for the fixity dissemination approaches. In the first study (Chapter 7.2), we selected 1,000 mementos from `archive.org` and only considered the HTML file in hash calculation. This first study focused more on disseminating fixity information and retrieving archived fixity information, rather than the structure of fixity information. In the second study, we used our full set of 16,627 mementos from 17 public web archives (Chapter 5) to ensure that our framework can work with any memento from

any web archive. The second study also used a new manifest file structure to include fixity information about not only the base HTML file but also all web resources embedded in a composite memento. The new manifest structure also contains (1) hash values generated by the URI-M-based (Chapter 6.7.1) and entity-based hashing techniques (Chapter 6.7.2), and (2) multiple aggregated hash values generated by three hashing techniques.

7.1 METHODOLOGY FOR DISSEMINATING AND VERIFYING FIXITY

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. We have two approaches for manifest dissemination, namely, *Atomic* and *Block*.

7.1.1 MANIFEST GENERATION

A manifest (identified by `URI-Manif`) consists of metadata summarizing 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 138 and has the following components:

- **@context**: 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**: The URI of an original resource, the URI of a memento, and the datetime when the memento was created, respectively.
- **@id**: The URI that identifies a published manifest file (URI-Manif).
- **http-headers**: Selected HTTP Response headers of the memento. As proposed by Jones et al. [92], we insert the `Preference-Applied` header to specify options used to retrieve the memento. For example, `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/
            c6ad485819abbe20e37c0632843081710c95f94829f59bbe3b6ad3251
            d93f7d2/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-Orig-link')
                         | tee >(sha256sum) >(md5sum) >/dev/null
                         | cut -d ' ' -f 1
                         | paste -d':' <(echo -e 'md5\nsha256') -
                         | paste -d' ' - -",

    "hash": "md5:969d7aba4c16444a6544bdc39eefe394 sha256:c68a215eb1
             c3edbf51f565b9a87f49646456369e51791a86106a6667630737a6"
}
```

Fig. 138: 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 [192].
- **hash**: The hash values calculated based on commands defined in `hash-constructor`.

7.1.2 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 of maintaining content separately from its fixity information is recommended by the TRAC report [103]. Manifests can be disseminated and 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`.

We briefly describe the steps involved in generating, publishing, and disseminating fixity information of mementos with examples. Figure 139 shows the web page `https://2019.jcdl.org` pushed into multiple archives, resulting in four mementos. The Python module `ArchiveNow` [193] can be invoked via the command-line interface or user interface for simultaneously disseminating a web page into on-demand web archives.

Next, as shown in Figure 140, for each memento, a manifest is generated and published on the web at the Archival Fixity server,

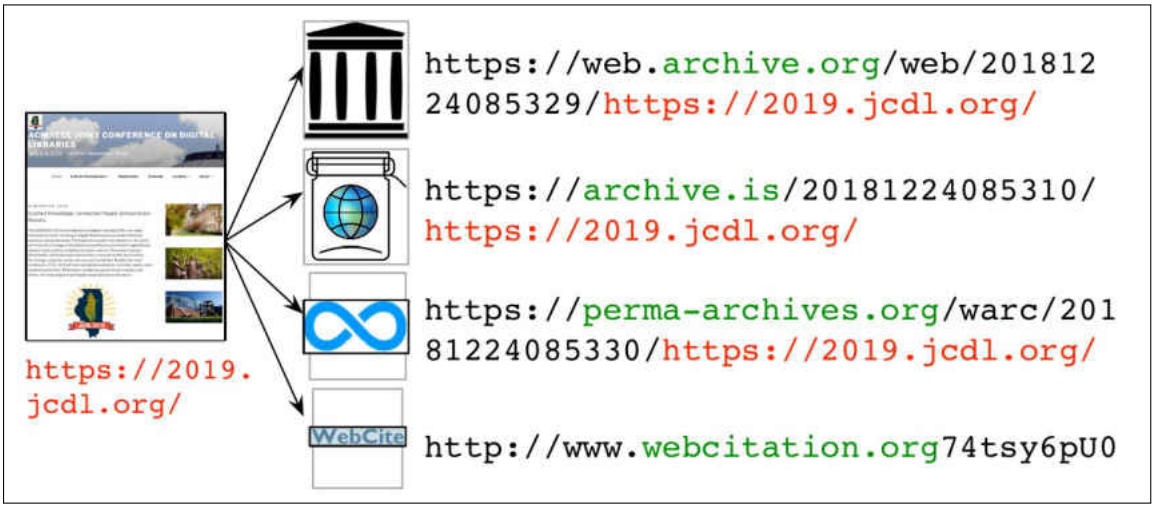


Fig. 139: A web page is pushed into multiple archives: `archive.org`, `archive.is`, `perma.cc`, and `webcitation.org`.

`http://manifest.ws-dl.cs.odu.edu`

so that archives are able to access and capture those manifests. For example, the manifest of the memento

`https://web.archive.org/web/20181224085329/https://2019.jcdl.org/`

is available at the `URI-Manif`

`manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org/web/20181224085329/https://2019.jcdl.org/`

This URI-Manif is a generic URI, which means if the Archival Fixity server creates another manifest for the same memento (marked in red), the server will publish it using the same generic URI. For this reason, the generic URI must always redirect to the most recent manifest of a memento, since each published manifest is made available on the web using a URI that contains the hash of the manifest. This URI is called a trusty URI (as explained in Chapter 3.2.1). This `302 Redirect` from the generic URI to the trusty URI has two advantages. First, having a trusty URI will help validate the manifest content, as the hash included in the URI is the hash of the content it identifies. Second and more importantly, we can use the generic URI to discover manifests in the Archival Fixity server and archived manifests in the archives. Therefore, even in cases where the Archival Fixity server is unavailable or compromised, we can still can discover manifests in the archives

Fig. 140: Compute fixity and publish it on the web

directly (e.g., using a TimeGate or TimeMap). For instance, the HTTP request to the manifest's generic URI

```
manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org/web/2018122408
5329/https://2019.jcdl.org/
```

rdirects (`302 Redirect`) to the trusty URI (Figure 141)

```
manifest.ws-dl.cs.odu.edu/manifest/20181224093024/8c31ccfbb3a664c9160f98b
e466b7c9fb9afa80580ab5052001174be59c6a73a/https://web.archive.org/web/201
81224085329/https://2019.jcdl.org/
```

Figure 142 shows an example of retrieving all mementos (the TimeMap) from the Internet Archive of the URI-Manif:

```
manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org/web/2018122408
5329/https://2019.jcdl.org/
```

As Figure 143 shows, requesting the memento of the manifest (with generic URI) found in the TimeMap results in `302 Redirect` to the archived manifest (with the trusty URI). Figure 145 shows how the live web, the archive, and the Archival Fixity server are related in the *Atomic* approach.

Generally, we build trust in the content of memento from the time when fixity information is computed and published. One of the best scenarios is when a manifest is generated at

```
$ 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/20181224093024/8
          c31ccfbb3a664c9160f98be466b7c9fb9afa80580ab5052001174be59c6a
          73a/https://web.archive.org/web/20181224085329/https://2019.
          jcdl.org/
HTTP/2 200
```

Fig. 141: The manifest identified with the generic URI redirects to the manifest with the trusty URI.

```
$ curl -i http://web.archive.org/web/timemap/link/https://
  manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org/web/20181
  224085329/https://2019.jcdl.org/

HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Wed, 01 May 2019 05:38:20 GMT
Content-Type: application/link-format
Transfer-Encoding: chunked
Connection: keep-alive
X-App-Server: wwwb-app38
X-ts: ----
X-location: cdx-p
X-Cache-Key: httpweb.archive.org/web/timemap/link/https://manifest.ws-
             dl.cs.odu.edu/manifest/https://web.archive.org/web/201812
             24085329/https://2019.jcdl.org/US
X-Page-Cache: MISS

<http://manifest.ws-dl.cs.odu.edu/manifest/https://web.archive.org/web
    /20181224085329/https://2019.jcdl.org/>; rel="original",

<http://web.archive.org/web/timemap/link/https://manifest.ws-dl.cs.odu.
    edu/manifest/https://web.archive.org/web/20181224085329/https
    ://2019.jcdl.org/>; rel="self"; type="application/link-format"; from
    ="Mon, 24 Dec 2018 09:33:54 GMT",

<http://web.archive.org>; rel="timegate",

<http://web.archive.org/web/20181224093354/http://manifest.ws-dl.cs.odu
    .edu/manifest/https://web.archive.org/web/20181224085329/https
    ://2019.jcdl.org/>; rel="first memento"; datetime="Mon, 24 Dec 2018
    09:33:54 GMT",
```

Fig. 142: Retreiving the TimeMap of a manifest from the Internet Archive. In this example, the TimeMap contains only one memento.

```
$ curl -sIL http://web.archive.org/web/20181224093354/http://manifest.
  ws-dl.cs.odu.edu/manifest/https://web.archive.org/web/2018122408532
  9/https://2019.jcdl.org/ | egrep -i "(HTTP/|^location:)"

HTTP/1.1 302 Found
Location: /web/20181224093354/http://manifest.ws-dl.cs.odu.edu/manifes
          t/20181224093024/8c31ccfbb3a664c9160f98be466b7c9fb9afa80580
          ab5052001174be59c6a73a/https://web.archive.org/web/20181224
          085329/https://2019.jcdl.org/
HTTP/1.1 200 OK
```

Fig. 143: The archived manifest with the generic URI redirects to the archived manifest with the trusty URI.

Fig. 144: Push the fixity information into multiple archives

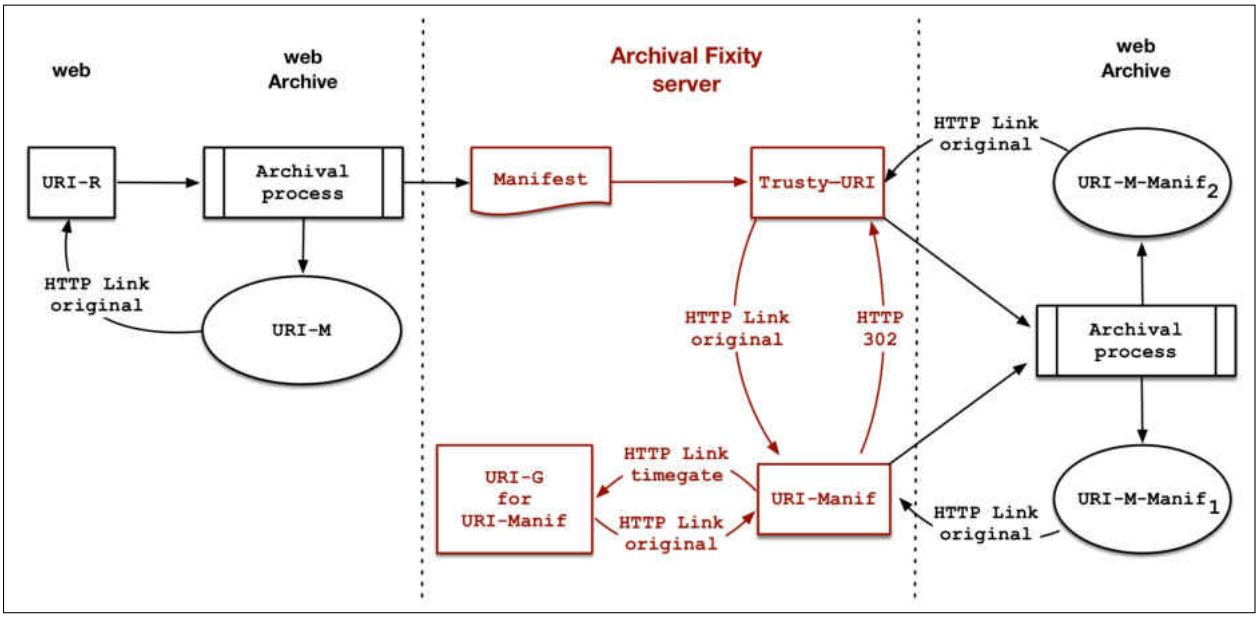


Fig. 145: The *Atomic* approach. The generic URI (URI-Manif) redirects to the most recent trusty URI, so when the archive captures the generic URI, the archive follows the `302 Redirect` and captures the trusty URI as well. This figure is a modified version of an original diagram contributed by Herbert Van de Sompel (from DANS).

ingest by the archive. In other words, the archive crawls a web page and immediately after that computes and publishes its fixity information.

The final step is to push the published manifest into multiple archives. In the example shown in Figure 144, the fixity information (or the manifest) of the memento from `archive.org` is disseminated to the same archive and other three archives, which are `archive.is`, `perma-archives.org`, and `webcitation.org`.

7.1.3 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 [194,195]), 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 [196]. Manifests' chain of blocks are 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 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 is performed in the following steps:

1. Identify a set of URI-Ms for their manifests to be included in the same block. (A strategically chosen set may improve block compression factor and enable a more efficient lookup for verification later.)

2. Generate their individual manifests in the form of a single-line JSON file (exclude `@id` field, needed in case of records being placed in a block, and eliminate many common fields that can go in the headers of the block).

3. Prefix each manifest JSON line with the Sort-friendly URI Reordering Transform (SURT) [197] of the corresponding URI-M.

4. Write these lines in a UKVS file along with the metadata headers as illustrated in Figure 148.

5. Add the content-addressable hash of the latest published block in the metadata as the previous block.
6. Sort the file using `LC_ALL=C` locale.
7. Calculate the content-addressable hash (e.g., SHA256) of this block.
8. Name the file using its content-addressable hash.
9. Compress the block file to reduce its size before archiving it.
10. Publish the compressed block file at a URI that contains its hash (using a trusty URI).
11. Make the entrypoint (the well-known URI) redirect to the latest block's URI (as illustrated in Figure 146).
12. Add a `Link` response header with appropriate links to navigate through the chain of blocks, which is visually illustrated on the landing page as shown in Figure 147. (A similar approach of creating bidirectional linked list of HTTP messages was used in the HTTPMailbox [198].)
13. Archive the entrypoint in multiple web archives, which will implicitly archive the latest block as, well due to the redirect.
14. Optionally, for further tamper-proofing, post the URI of the newly published block on immutable platforms not controlled by a single authority (e.g., Twitter and GitHub's Gist).

Although the number of public web archives is increasing [14, 15], only a few support an on-demand web archiving service. However, a small number (greater than one) of independent on-demand archives can suffice for the purpose of disseminating manifests.

The *Block* dissemination approach has a number of advantages over the *Atomic* approach. It requires far fewer network requests to push it to web archives and creates significantly fewer independently published manifest resources to keep track of mementos. By bundling multiple manifests in a single file, The *Block* yields a significant compression factor due to the repeated boilerplate content in each manifest file. As web archives die and new ones come to life, these blocks can be replicated and migrated externally to other places efficiently. In the case of the *Atomic*, approach we might lose historical manifests as old web archives die without donating their holdings to live archives. Moreover, these blocks are

```
$ curl -IL https://manifest.ws-dl.cs.odu.edu/blocks

HTTP/2 302
content-type: text/html; charset=utf-8
date: Mon, 21 Jan 2019 22:27:14 GMT
location: https://manifest.ws-dl.cs.odu.edu/blocks/59bc17511de502b7a7bd
          f39b2020c3bd4ad08aaefd7135604edb2a8e3e89540b
server: ArchivalFixity/0.1
content-length: 417

HTTP/2 200
accept-ranges: bytes
cache-control: immutable
content-disposition: attachment; filename="59bc17511de502b7a7bdf39b2020
                     c3bd4ad08aaefd7135604edb2a8e3e89540b.ukvs.gz"
content-encoding: gzip
content-type: application/ukvs
date: Mon, 21 Jan 2019 22:27:14 GMT
etag: "59bc17511de502b7a7bdf39b2020c3bd4ad08aaefd7135604edb2a8e3e89540b
      "
expires: Tue, 22 Jan 2019 10:27:14 GMT
last-modified: Fri, 11 Jan 2019 18:19:00 GMT
link: <https://manifest.ws-dl.cs.odu.edu/blocks/59bc17511de502b7a7bdf39
      b2020c3bd4ad08aaefd7135604edb2a8e3e89540b>; rel="self", <https://
      manifest.ws-dl.cs.odu.edu/blocks/3c4575450979f4283ffb5a1b385450a
      c4c82f1b746de34385dbc177e493a6096>; rel="prev", <https://manifest
      .ws-dl.cs.odu.edu/blocks/7bbf757046ac0a0a60015a1cb847c3189160d18c
      809b210073822df157609e01>; rel="first", <https://manifest.ws-dl.
      cs.odu.edu/blocks/59bc17511de502b7a7bdf39b2020c3bd4ad08aaefd71356
      04edb2a8e3e89540b>; rel="last"

server: ArchivalFixity/0.1
content-length: 15227
```

Fig. 146: Requesting the Blocks entrypoint `https://manifest.ws-dl.cs.odu.edu/blocks`

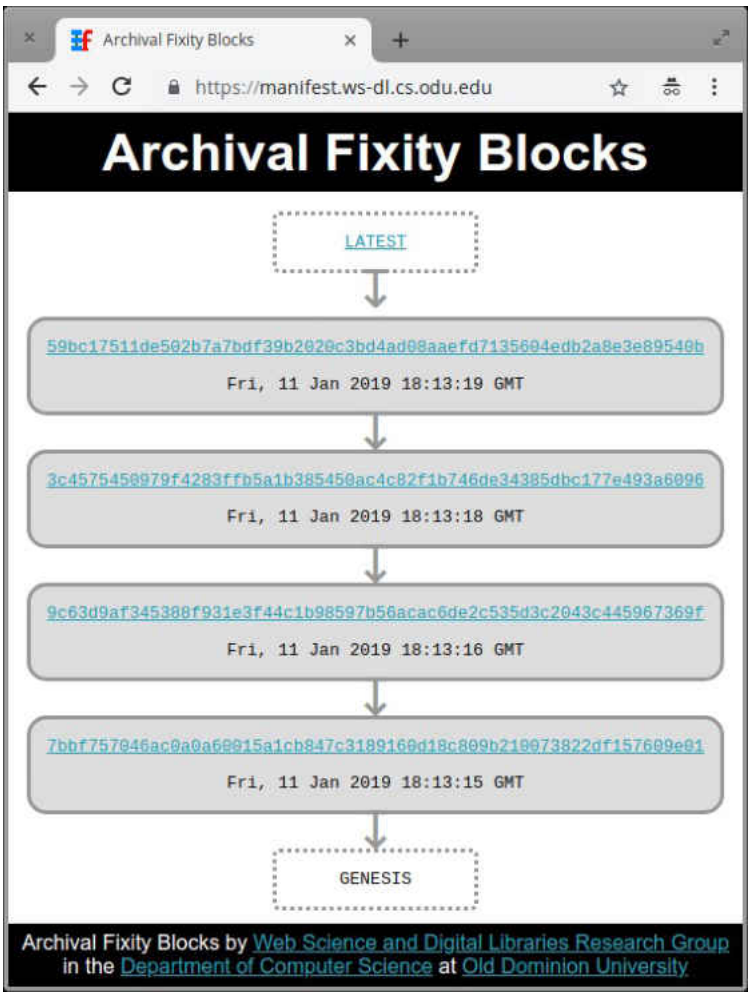


Fig. 147: The landing page showing a chain of blocks.

more tamper-proof than atomic manifests due to chaining. On the other hand, the *Block* approach has the disadvantage of shifting the burden of lookup of a specific record in the entire chain of blocks to the user or a service that provides verification. While individual blocks are binary searchable for fast lookup, as the number of blocks increases, one has to scan through all of them. However, this can easily be solved by scanning the entire chain once and creating a search index over the SURT field.

7.1.4 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*

```
1 !context ["http://oduwsdl.github.io/contexts/fixity"]
2 !fields {keys: ["surt"]}
3 !id {uri: "https://manifest.ws-dl.cs.odu.edu/"}
4 !meta {created_at: "20190111181327"}
5 !meta {prev_block: "sha256:
     d4eb1190f9aaae9542fd3ad8a3c4519450cfb00845b632eb2b3f4f098a34144d"}
6 !meta {type: "FixityBlock"}
7 org,archive,web)/web/19961022175434/http://search.com/ {<Single line
     JSON as illustrated in Figure 2>}
8 org,archive,web)/web/19961219082428/http://sho.com/ {<Single line JSON
     >}
9 org,archive,web)/web/19961223174001/http://reference.com/ {<Single line
      JSON>}
10 ...
```

Fig. 148: A sample Block with metadata headers and records

approach, this step requires also discovering archived copies `URI-M-Manif` of the manifest.

(b) Recompute current fixity information of the memento.

(c) Compare current fixity information with discovered manifests.

In the *Atomic* approach, we can discover a manifest of a given memento through the Archival Fixity server. Which manifest is returned depends on the server's API. For example, the server may respond with the closest manifest to the memento's creation date or return the manifest that is closest to a given datetime (i.e., via a TimeGate). Once a manifest is discovered, we may use TimeGates and/or TimeMaps to retrieve its archived copies available in web archives. Again, it is possible to discover archived manifests using the generic URI even without the Archival Fixity server being involved. Next, we compute current fixity information by generating a new manifest for the given memento. Then, we compare current hash values in the new manifest with the hashes in the discovered archived manifests. In this compression step, we should only consider independent copies of the manifest. For example, if an archived manifest is delivered from the same archive where the memento is from, then this copy of the manifest should not be considered independent. In other cases, two manifests might be discovered in two cooperating archives (e.g., we know `archive-it.org` is a service established by the Internet Archive).

In case of the *Block* approach, the fixity verification server (or any equivalent tool) needs

to have access to all the blocks, either over HTTP (e.g., from a web archive) or stored locally. These blocks are then scanned for one or more matching records for the given *URI-M*. Corresponding single-line JSON entries (as shown in Figure 148) are extracted as historical fixity records for comparison. Remaining steps for creating current fixity information, comparing with the historical records, and generating the response summary are the same as in the case of *Atomic* approach.

Due to the immutable nature of blocks we can only have back references, creating a single linked list pointing from the most recent blocks to the older ones. With the help of some external metadata, our archival fixity block server provides bidirectional navigational links for easy navigation along the chain back and forth (as illustrated in Figure 146 with `first`, `last`, `prev`, and `next` link relations in the `link` header). The content of these blocks is sorted to enable fast lookup in each block using binary search, but the chain of blocks has to be scanned linearly, which can decrease the throughput as the number of blocks increases. To deal with this issue, one can create an inverted index of existing blocks, treating *URI-Ms* as the keyword and blocks as the document. Additionally, the chain of blocks is in chronological order, which makes it easy to create a lightweight skip index to identify segments of the chain that were created around certain points of time in the past. Creating large blocks with a slowly growing chain will be more efficient than a rapidly growing chain of small blocks. However, an optimal block size can be decided based on how long one is willing to wait for enough records to be available for a new block creation and on the largest size of a single block that can easily be stored in web archives. Creating blocks with strategically grouped URI-Ms (e.g., mementos of nearby `datetime` values, URI-Rs from a set of domains, or URI-Ms from a set of archives) can also improve the efficiency of lookup (or indexing).

7.2 INITIAL EVALUATION OF ATOMIC AND BLOCK APPROACHES

We conducted an initial study on 1,000 mementos from the Internet Archive, which are a subset of the larger set of URI-Ms described in Chapter 5. We did not take the size of mementos into consideration (i.e., the number of embedded resources, such as images and JavaScript/CSS files) because, at this point, we compute fixity 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. For example, `webcitation.org`[1] responded with "WebCite has flagged your IP address for suspicious activity" after making 100 requests, but the issue was resolved after contacting the archive. `perma.cc` on the other hand allows users to freely submit a maximum of 10 URIs per month. Fortunately, the archive supported us for this study by increasing this limit, so we were able to disseminate more manifests to the archive. 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.

We wrote Python scripts for performing different functions:

- **generate_atomic()**: Accepts a URI-M and returns the fixity information of the memento in JSON format. We generated 3,000 manifests. The main purpose of generating three manifests for each memento is because we are interested in reporting the average time for generating a manifest for each memento.

- **publish_atomic()**: Submits a given JSON file 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.

- **disseminate_atomic()**: Pushes a published manifest into different archives using ArchiveNow. In our study, we used `archive.org`, `archive.is`, `perma.cc`, and `webcitation.org` resulting in creating 12,000 archived manifests (i.e., 3,000 URI-M-Manif in each archive). We used the Generic URI to push manifest into archives. Again, this URI always redirects to the trusty URI. If archives consider a "302 Redirect" as a separate resource, then the total number of archived resources created in the four archives was 24,000.

- **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 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.

[1]The archive `webcitation.org` no longer accepts archiving requests after July 9, 2019. Our study was conducted prior to this date.

- **generate_block()**: Accepts multiple JSON files. It generates one or more blocks depending on the selected block size. In this study, we set it to 100 manifests per block, so the total number of generated blocks was 10.

- **disseminate_block()**: Pushes a block into two archives (`archive.org` and `perma.cc`). Again, because we are interested in calculating the average time for disseminating a block, each block is pushed three times into both archives, resulting in creating 60 archived blocks (i.e., 30 per archive). We did not use `archive.is` and `webcitation.org` because .gz files were not handled correctly by those archives.

- **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.

In addition to the Python scripts, we implemented the Archival Fixity server that is responsible for publishing and discovering manifests and blocks. For example, Figure 149 shows a request for discovering the closest manifest's creation date to December 22, 2018 for the given memento. The server response indicates that the closest manifest was created on December 12, 2018.

```
$ curl -I https://manifest.ws-dl.cs.odu.edu/manifest/20181222/https://
  web.archive.org/web/20171115140705/http://rln.fm/

HTTP/2 302 Found
content-length: 501
content-type: text/html; charset=utf-8
date: Thu, 10 Jan 2019 09:16:40 GMT
location: https://manifest.ws-dl.cs.odu.edu/manifest/20181212074423/bd
          669de8835e38d54651fe9d04709515beec0c727db82a5366f4bc2506e103
          d8/https://web.archive.org/web/20171115140705/http://rln.fm/
server: ArchivalFixity/0.1
```

Fig. 149: Discovering the closest manifest to December 22, 2018 for the memento `web.archive.org/web/20171115140705/http://rln.fm/`.

The selected number of records per block affects the total size of all blocks and the time required to generate these blocks. Figure 150 illustrates that creating large blocks with a slowly growing chain is more efficient than a rapidly growing chain of small blocks. As mentioned in Chapter 7.1.4, one factor of choosing the optimal number of records in each block is the largest size of a single block that can easily be stored in web archives. For example, we tested the Internet Archive (IA) to identify the largest single file that the

archive can accept for preservation. After submitting multiple files with different sizes, we found that IA can accept up to 800 MB, and beyond that the archive returns "504 Gateway Timeout".

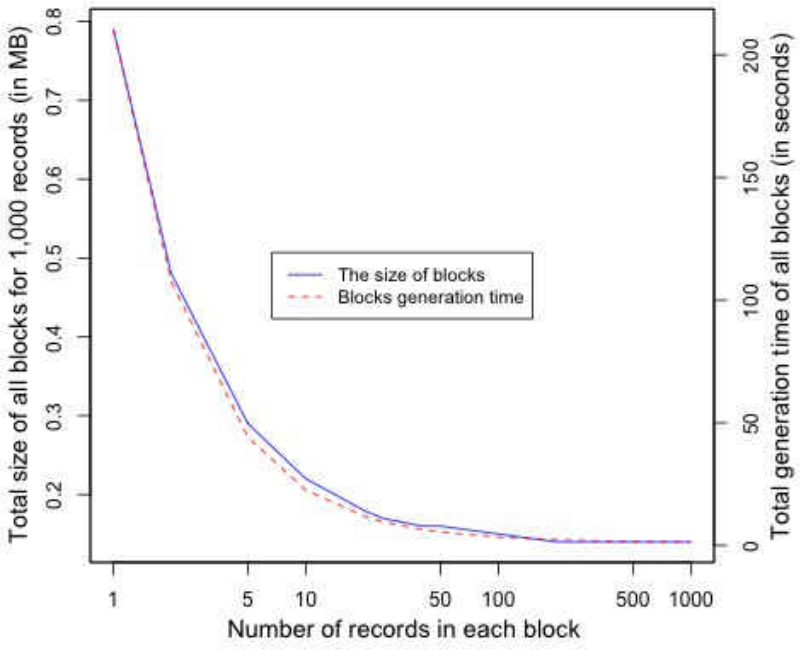


Fig. 150: The effect of the selected number of records per block.

Figure 151 illustrates the distribution of the average time taken to generate manifests. We generated three manifests for each memento, and calculated the average time, so the 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 fixity information of the downloaded content, and 3) storing the fixity information locally in JSON format. We noticed that 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 the fixity information of the same number of mementos.

As expected, the time for disseminating manifests and blocks was the longest compared with other operations, such as generating and verifying manifests. Figure 152 shows that pushing manifests into `webcitation.org` (WebCite) takes more 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 18 indicates. We observed that `archive.org` and

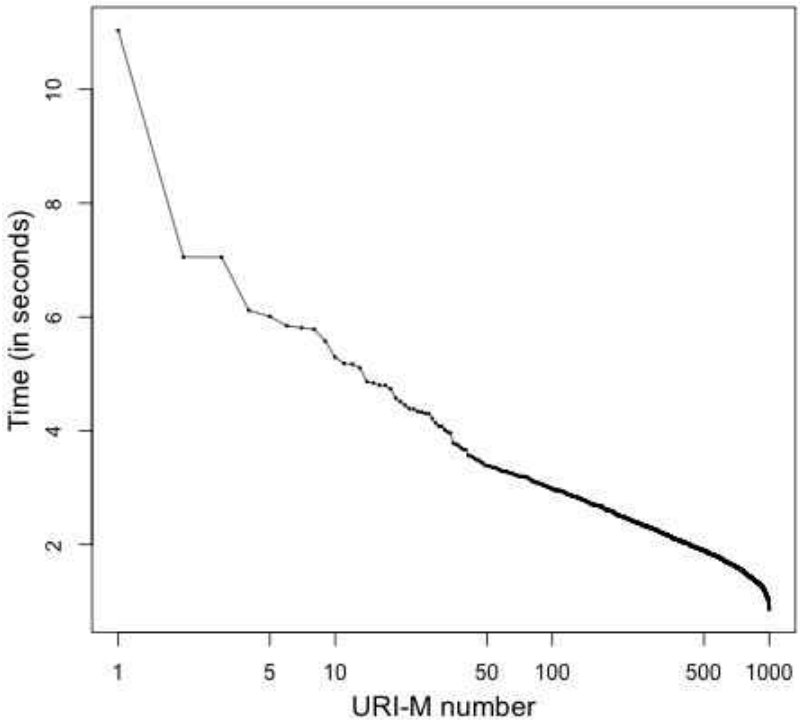


Fig. 151: Generating manifests of mementos.

`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 153.

Given a collection of N mementos and K web archives, the total number of resources that we are creating in the K archives by the *Atomic* and *Block* approaches are $(N * K)$ and $(k * (N/B))$ respectively, where B is the selected block size. In our study, $N = 1,000$, $k_{\text{atomic}} = 4$, $k_{\text{block}} = 2$, and $B = 100$. Then a total of $12,000$ resources were created by the *Atomic* approach and only 60 resources were created by the *Block* approach considering the fact that we repeated the dissemination process for three times.

TABLE 18: Average time (in seconds) for disseminating and downloading of manifests and blocks.

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	-

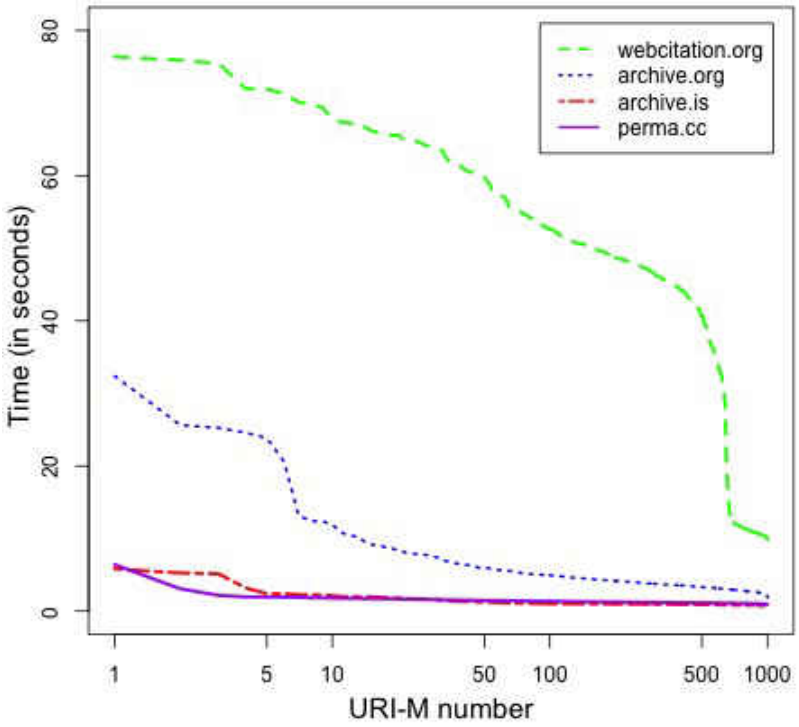


Fig. 152: Disseminating manifests to four archives.

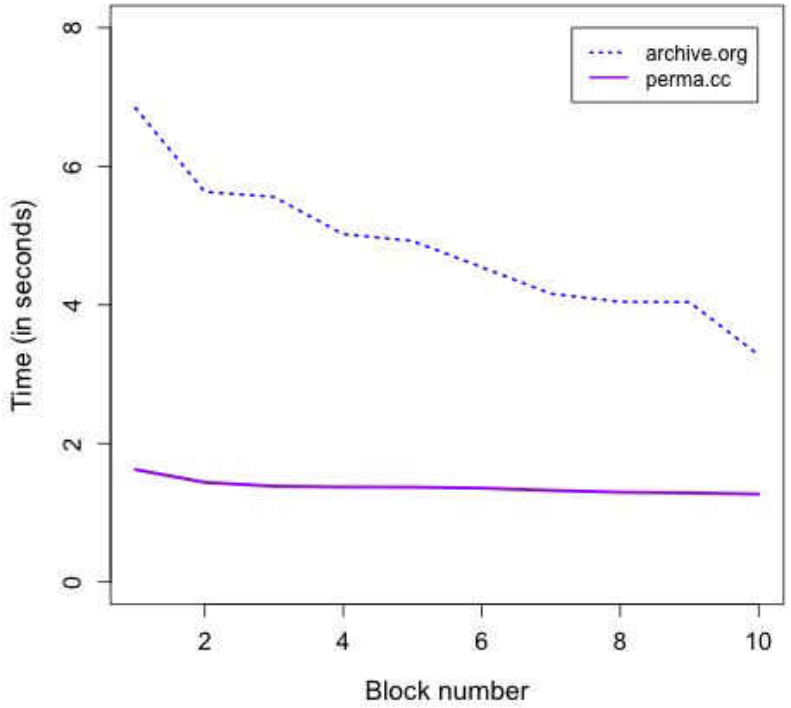


Fig. 153: Disseminating blocks to two archives.

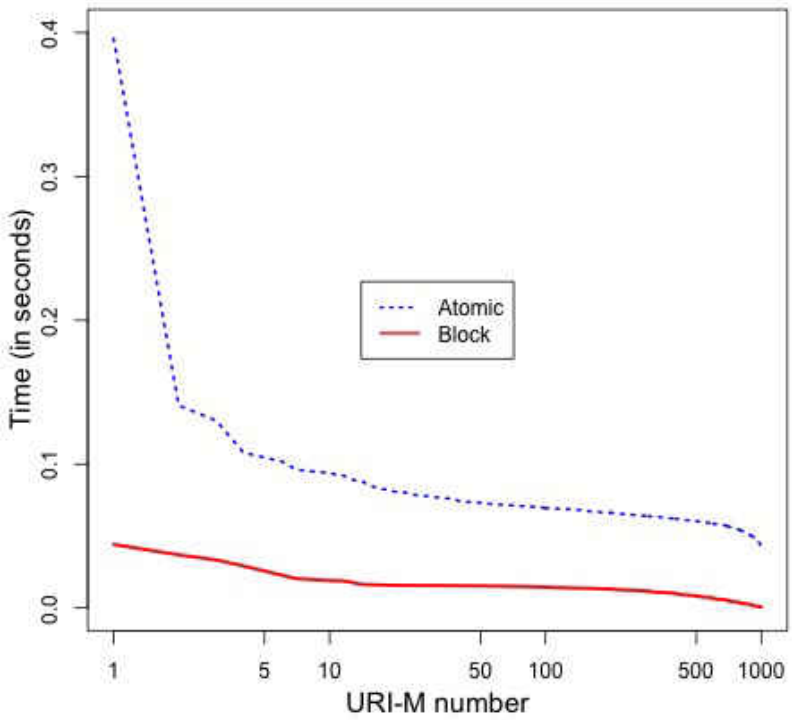


Fig. 154: Discovering manifests by both approaches.

Figure 154 shows the time required to discover manifests for each memento from the Archival Fixity server. Figure 162 illustates the total time for verifying the fixity of all mementos by both approaches. 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 performs 4.46X faster than the *Atomic* approach on verifying the fixity of memento. Although we have predicted that some mementos might not be verified for reasons such as an archive responds with `HTTP 500` Error, we have not yet encountered any failed cases (i.e., all mementos are verified successfully).

7.3 EVALUATION USING A NEW MANIFEST FILE STRUCTURE AND LARGER SET OF MEMENTOS

We performed a new evaluation on the *Atomic* and *Block* approaches for several reasons. In our previous study (Chapter 7.2), we used only mementos from `archive.org`. In this new study, we want to include mementos from various public web archives (17 archives) to ensure that our framework can work with any memento from any web archive.

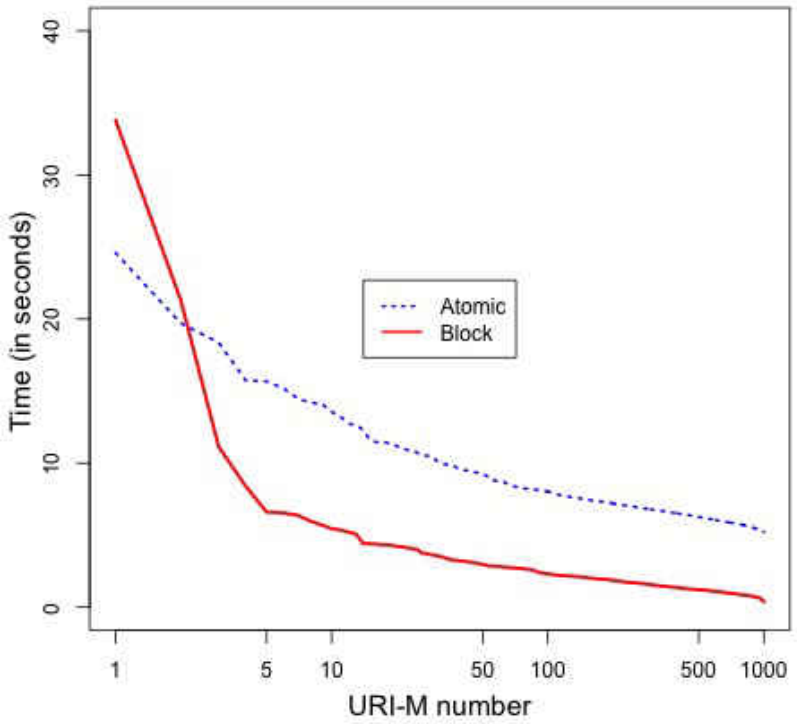


Fig. 155: Verifying mementos by both approaches.

We also describe a new manifest file structure that contains fixity information about not only the base HTML file but also about any web resource (e.g., images) embedded in a composite memento. This is important because, as we described (Chapter 1.4), a single hash value calculated on only the base HTML file of a composite memento will not help detecting changes on any embedded resources. The new manifest structure should also include hash values generated by the URI-M-based (Chapter 6.7.1) and the entity-based hashing techniques (Chapter 6.7.2). For each resource in a composite memento, we create other important hash values to be included in the manifest file. These hash values can help detecting whether a particular resource has small or big changes over time. Finally, we will compare results from the previous evaluation (Chapter 7.2) with results from the new study. For example, will the average manifest generation time change in the new study?

7.3.1 NEW MANIFEST FILE STRUCTURE

Our new proposed structure of the manifest file should include fixity information about the base HTML file and all resources embedded in a composite memento. It should also contain hash values generated by the three hashing techniques described in Chapter 6.7 and other hashes values for estimating the level of changes in a resource over time. The new

structure of the manifest file is illustrated in Figure 156 and has the following components:

- **@context**: 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-M**: The URI of a memento.
- **@id**: The URI that identifies a published manifest file (URI-Manif).
- **composite-memento-entity-hash**: An aggregated hash value computed using a Merkle tree, built on only the HTTP entity bodies of all resources comprising a composite memento.
- **composite-memento-URI-M-hash**: An aggregated hash value computed using a Merkle tree, built on only the URI-Ms of all resources comprising a composite memento.
- **composite-memento-overall-hash**: An aggregated hash value using a Merkle tree, calculated based on the HTTP response headers and entity bodies, the HTTP status codes, and URI-Ms of all resources comprising a composite memento (Chapter 6.1).

  The manifest file also has the following components for each resource comprising a composite memento:

- **http-headers**: Selected HTTP Response headers of the resource. As proposed by Jones et al. [92], we may insert the `Preference-Applied` header to specify options used to retrieve the memento. For example, `Original-Content` refers to the raw memento—accessing unaltered archived content because archives by default return the memento after transforming its content.
- **status-code**: The HTTP status code of the resource returned from the archive.
- **URI-M**: The URI of the resource.
- **entity-hash**: The hash value calculated on the HTTP entity body of the resource.
- **overall-hash**: The hash value calculated on the HTTP headers and entity, HTTP status code, and URI-M of the resource.

```
{
    "@context": "http://oduwsdl.github.io/contexts/fixity",
    "created_at": "Wed, 08 Apr 2020 02:22:56 GMT",
    "@id": "http://manifest.ws-dl.cs.odu.edu/manifest/20200408022256/
       d5dd6bf95219300d642cd03356d541eceb2a7fef9efd62c1b531453875bf1f04/
       https://web.archive.org/web/19970104075414/http://www.un.org
       :80/",
    "composite-memento-uri-m-hash": "8bb453d8aa...db5f5fbf6",
    "composite-memento-entity-hash": "cdf47062a...3ebe030b0",
    "composite-memento-overall-hash": "69ca5a85...b9206930",
    "uri-m": "https://web.archive.org/web/19970104075414/http://www.un.
       org:80/",
    "resources": [
        {
            "http-headers": ["X-Archive-Orig-last-modified", "Content-Type
               ", "X-Archive-Orig-date", "Memento-Datetime", "X-Page-Cache
               "],
            "entity-phash": null,
            "entity-hash": "ba140a5bede7f10bea0...7514725862eda82a003",
            "overall-hash": "3e4133b3766c2a58d6f...23f5f95a206a1ba9878",
            "entity-simhash": 9695187482751709335,
            "uri-m": "https://web.archive.org/web/19970104075414/http://
               www.un.org:80/",
            "status-code": "200"
        },
        {
            "http-headers": ["X-Archive-Orig-last-modified", "X-Archive-
               Orig-date", "Memento-Datetime", "Content-Type", "X-Page-
               Cache"],
            "entity-phash": "87d5798529a75a58",
            "entity-hash": "d9305a4da88570700d92...17c0c0f72b9f4e514b9",
            "overall-hash": "de002fbe292372199e6...6059044f4695f0dda2c",
            "entity-simhash": null,
            "uri-m": "https://web.archive.org/web/19970315165323im_/http
               ://www.un.org/homepage.gif",
            "status-code": "200"
        },
        {
            "http-headers": ["Content-Type", "Memento-Datetime", "X-Page-
               Cache"],
            "entity-phash": "219d1a8362a71040",
            "entity-hash": "d5fd59c929e1c62b17b...d5e321f64a919e4294e",
            "overall-hash": "7df9dde47fa5bab643...cef3c84694bb1db8b1c",
            "entity-simhash": null,
            "uri-m": "https://web.archive.org/web/20120129120857/http://
               web.archive.org/screenshot/http://www.un.org/",
            "status-code": "200"
        }
    ]
}
```


Fig. 156: A manifest showing fixity information of the memento `https://web.archive.org/web/19970104075414/http://www.un.org:80/`. This composite memento consists of three resources.

- **entity-phash**: The hash value generated on the entity body of a resource using *phash* hashing function [114, 115]. We can use *phash* to find how different/similar images are.

- **entity-simhash**: The hash value generated on the entity body of a resource using *simhash* hashing function [108]. We can use *simhash* to find how different/similar text files are.

7.3.2 RESULTS OF THE ATOMIC AND BLOCK APPROACHES EVALUATION ON 16K MEMENTOS

The main difference between this study and the previous study (Chapter 7.2) is that we used a larger set of mementos, 16,608 from 17 public web archives, and the new manifest file structure (Chapter 7.3.1). This new study requires pushing at least one manifest file for each memento into multiple web archives. We only encountered one issue in sending about 16k archival requests to web archives: `archive.is` rejected all archival requests after accepting and successfully handling about 11k requests, but the issue was solved after contacting the archive administrator. We also had to exclude `webcitation.org` from this study because the archive currently does not accept any archival requests [184], but we still consider creating fixity information for mementos from this archive.

We want to measure the time taken to generate, disseminate, and verify manifests in the *Atomic* and *Block* approaches. In addition, we want to compute the size of files created in both approaches.

We used modified versions of Python scripts from the previous study for performing different functions:

- **generate_atomic()**: Accepts a URI-M and returns fixity information of the memento in JSON format. We generated 16,608 manifests (one manifest per memento). We are interested in reporting the time for generating a manifest for each composite memento. We must download the composite memento in order to generate its fixity information (cf. downloading only the base HTML file of a memento in our previous study).

- **publish_atomic()**: Submits a given JSON file 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.

- **disseminate_atomic()**: Pushes a published manifest into different archives using ArchiveNow. In our study, we used `archive.org`, `archive.is`, and `perma.cc` resulting in creating 49,824 archived manifests (i.e., 16,608 URI-M-Manif in each archive). We used the Generic URI to push manifest into archives. Again, this URI always redirects to the trusty URI. If archives consider a "302 Redirect" as a separate resource, then the total number of archived resources created in the four archives was 99,648.

- **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 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.

- **generate_block()**: Accepts multiple JSON files. It generates one or more blocks depending on the selected block size. In this study, we set it to 1038 manifests per block, so the total number of generated blocks was 16.

- **disseminate_block()**: Pushes a block into two archives (`archive.org` and `perma.cc`). Each block is pushed three time into both archives resulting in creating 32 archived blocks (i.e., 16 per archive). We did not use `archive.is` because .gz files were not handled correctly by this archive.

- **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 and the number of verified resources.

In addition to these Python scripts, we used the same Archival Fixity server from our previous study.

As in our previous study, the result of this study also indicate that creating large blocks with a slowly growing chain is more efficient than a rapidly growing chain of small blocks, as shown in Figure 157. Figure 158 shows the time taken to generate manifests, and Table 19 shows the median and the average time to generate manifest files per archive. We generated

one manifest for each memento, so the the total number of generated manifests is 16,608. The manifest generation time includes:

- downloading a composite memento using `Squidwarc`
- downloading the raw version of each resource comprising the composite memento
- computing the fixity information for the downloaded content
- storing the fixity information locally in a JSON file

The manifest generation time in this study takes longer than the time taken to generate manifest files in our previous study. It is because in this study all resources comprising a composite memento must be download, while in the previous study, only the base HTML file is downloaded. The total size of the generated manifest files is 366 MB, and the total size of the downloaded composite mementos is 31.12 GB. The median size of a manifest file is 15.29 KB, and the median size of a composite memento is 1143.85 KB. Therefore, this size of a manifest represents 1.33% of the size of a composite memento. The total size the generated blocks is 274 MB. This indicates again that the *Block* approach requires less storage space than the *Atomic* approach to store fixity information of the same number of mementos.

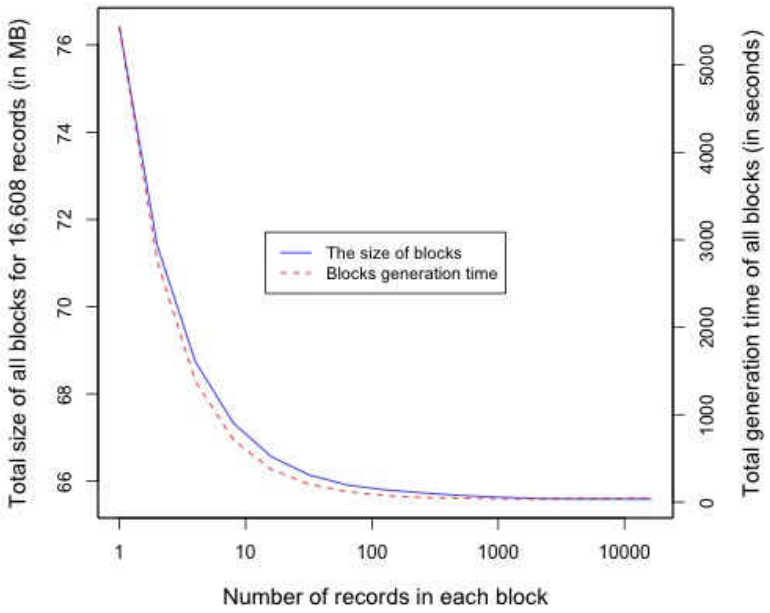


Fig. 157: The effect of the selected number of records per block.

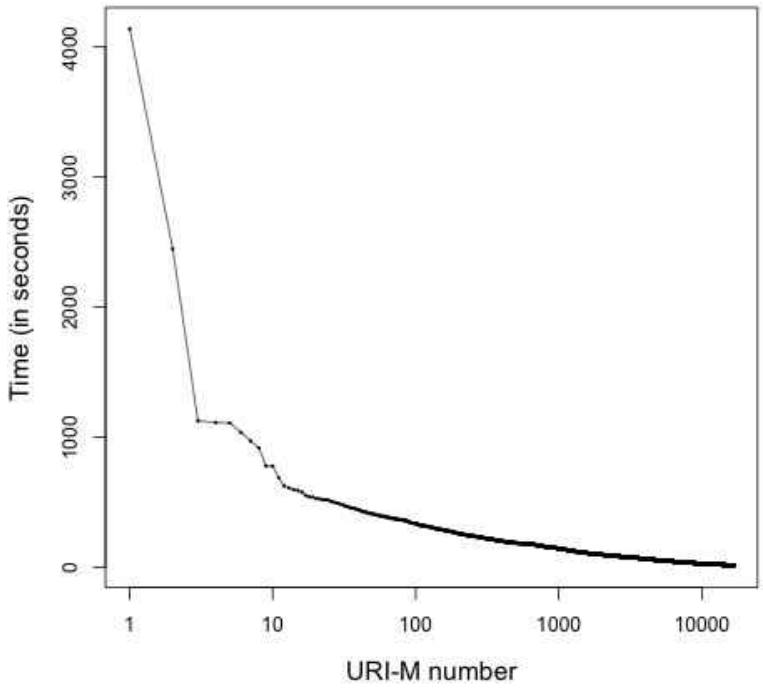


Fig. 158: Generating manifests of 16,608 mementos.

Unlike the previous study, this study shows that verifying mementos took the longest time compared with other operations, such as generating and disseminating manifests. Figure 159 shows that pushing manifests into `archive.org` takes a much longer time than other archives. On average, we wait for 5.36 seconds before `archive.org` finishes processing an archival request of a manifest, while the manifest disseminating average time drops down in the other two archives as Table 20 indicates. In sum, it takes about 1.09X and 4.54X longer to disseminate a manifest to `perma.cc`, `archive.org`, respectively, than `archive.is`, while it takes 9.20X 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 160.

The total number of resources created by the *Atomic* approach was $16,603 * 3 = 49,824$ and only $16 * 2 = 32$ resources were created by the *Block* approach.

Figure 161 shows the time required to discover and download manifests of each memento from the web archives and Archival Fixity server. Figure 162 illustrates the total time for verifying the fixity of all mementos by both approaches. The verification time in Figure 162 does not include the time for discovering manifests, computing current fixity information, or downloading copies of manifests (in the *Atomic* approach). If we include all of these

TABLE 19: The median and average time to generate manifest files per archive.

Archive	Median time (in seconds)	Average time (in seconds)
perma.cc	363.9	440.9
archive.is	154.1	163.4
webharvest.gov	99.94	99.60
swap.stanford.edu	62.50	62.59
vefsafn.is	48.97	49.53
archive.bibalex.org	38.31	38.30
nationalarchives.gov.uk	36.87	36.93
archive-it.org	34.24	40.29
arquivo.pt	32.89	33.00
webarchive.org.uk	30.15	30.15
webarchive.loc.gov	27.54	27.62
webcitation.org	23.52	23.55
digar.ee	21.52	21.52
web.archive.org	17.38	17.32
collectionscanada.gc.ca	15.49	15.19

TABLE 20: Average time (in seconds) for disseminating and downloading of manifests and blocks.

Operation	archive.is	perma.cc	IA
Manifest dissemination	1.18	1.29	5.36
Block dissemination	-	1.66	15.28
Manifest download	0.85	0.89	2.09
Block download	-	4.45	11.51

factors in the memento verification time, then on average, the verification time of the fixity of a memento would be 58.31 seconds by the *Atomic* approach and 55.32 seconds by the *Block* approach, so the *Block* approach performs 1.05X faster than the *Atomic* approach on verifying the fixity of memento.

In addition to the final hash values calculated on a composite mementos using the URIM-based, entity-based, and complete hashing techniques, we can use the hash values of embedded resources, which are also stored in the manifest file, to detect how many resources are verified in the composite memento and how many resources are not. For example, Figure 163 shows the composite memento

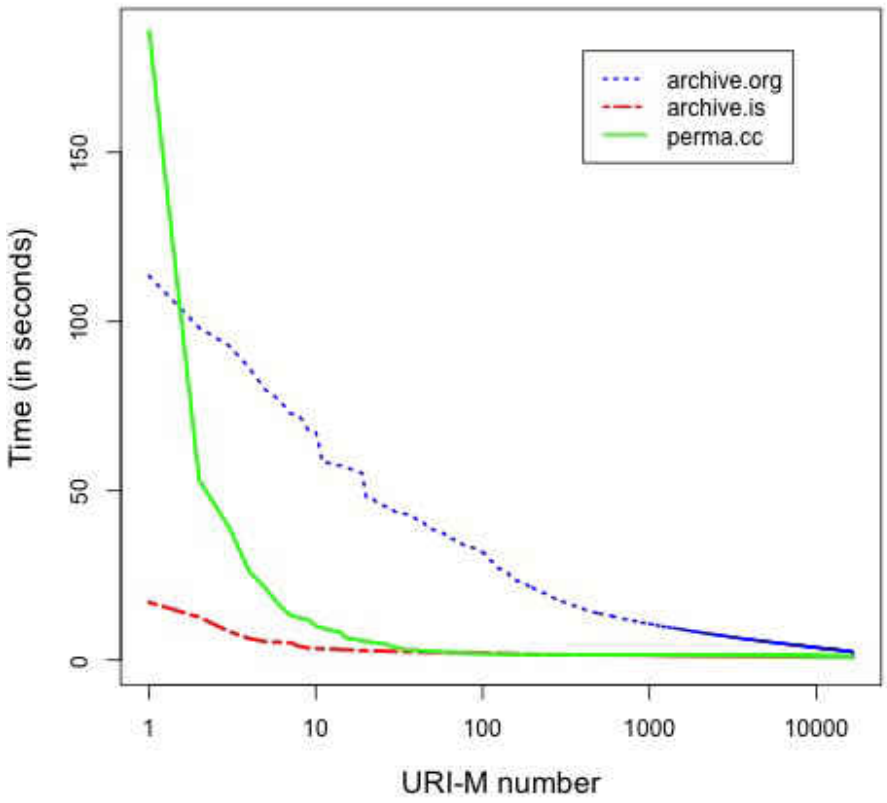


Fig. 159: Disseminating manifests to three archives.

`http://wayback.archive-it.org/all/20160701161239/http://www.catanddog firstaid.com/en/`

consists of 89 resources (or URI-Ms) including the URI-M of the base HTML file. Most of these resources, 72, have been verified using the following manifest files:

- `https://web.archive.org/web/20200413152403/https://manifest.ws-dl.cs.odu.edu/manifest/20200408010523/3c45ee1cd13dccc851b43bf5ab07584e47af9adcd9e21cc08ee511a3d002faea/http://wayback.archive-it.org/all/20160701161239/http://www.catanddogfirstaid.com/en/`
- `https://archive.li/wip/zfSvj`
- `https://perma.cc/WQ9Y-FKLN`
- `https://manifest.ws-dl.cs.odu.edu/manifest/20200408010523/3c45ee1cd13dccc851b43bf5ab07584e47af9adcd9e21cc08ee511a3d002faea/http://wayback.archive it.org/all/20160701161239/http://www.catanddogfirstaid.com/en/`

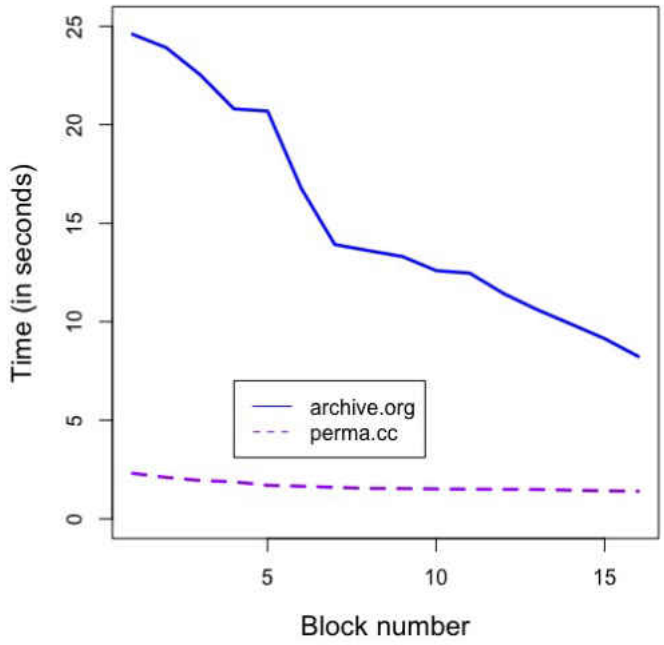


Fig. 160: Disseminating blocks to two archives.

The number of verified resources varies in Figure 163 because it depends on the hashing technique used to generate final hash values. For example, using the entity hashing technique, we have 49 different HTTP entity bodies, and only one of them has not been verified. The results also show the number of distinct URI-Ms (90 in Figure 163) that are previously seen when replaying the same composite memento in the past. Those URI-Ms are stored in the manifest files.

Table 21 shows the number of verified mementos (out of 16,608 mementos) by each hashing technique: the URIM-based hashing technique, entity-based technique, and the hash technique that is based on all components of a composite memento (i.e., complete hashing). The table indicates that without fulfilling the hashing guidelines (defined in Chapter 4), only 50.14%, 20.66%, and 14.21% of mementos could be verified using URIM-based, entity-based, and complete hashing technique, respectively. However, when we considered these predefined guidelines when generating final hash values, the percentage of verified mementos increased to 52.72%, 72.98%, and 48.72%. We excluded 1,302 mementos before we even started the process of verifying their fixity because the current download of the content of these composite mementos had transient errors (e.g., incomplete HTTP entity bodies), and based on one of the predefined hashing guidelines, we cannot generate hash values on

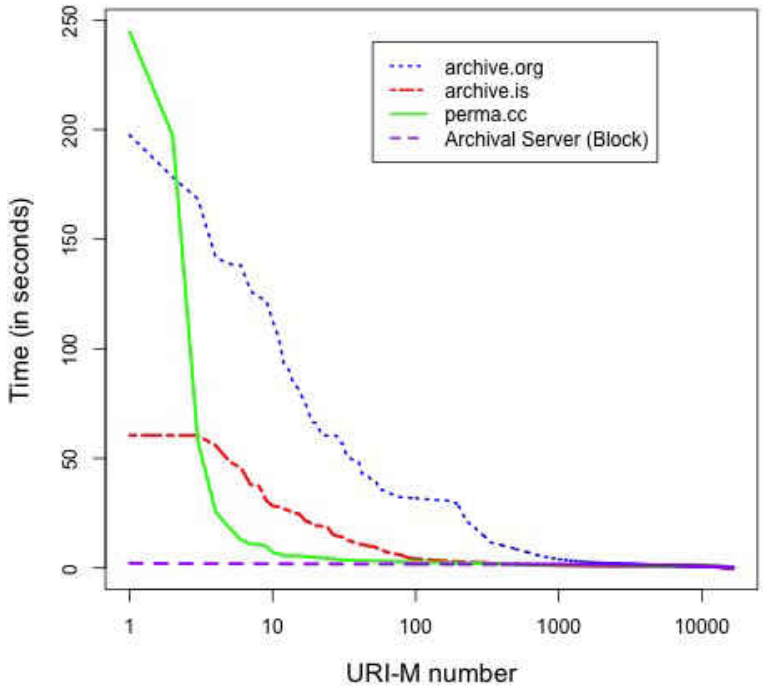


Fig. 161: Discovering and downloading manifest files in the *Atomic*/*Block* approaches per archive.

mementos with transient errors. The main reason why we still observe unverified mementos even after considering the predefined guidelines is because of JavaScript, which dynamically adds/loads different embedded resources each time the composite memento is replayed. Other reasons include archival redirects and the lack of raw mementos by some archives.

7.4 CHAPTER SUMMARY

In this chapter, we introduced two approaches, *Atomic* and *Block*, that can be used to disseminate fixity information to web archives and verify the fixity of mementos. 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. Therefore, the *Block* approach creates fewer web resources than the *Atomic* approach.

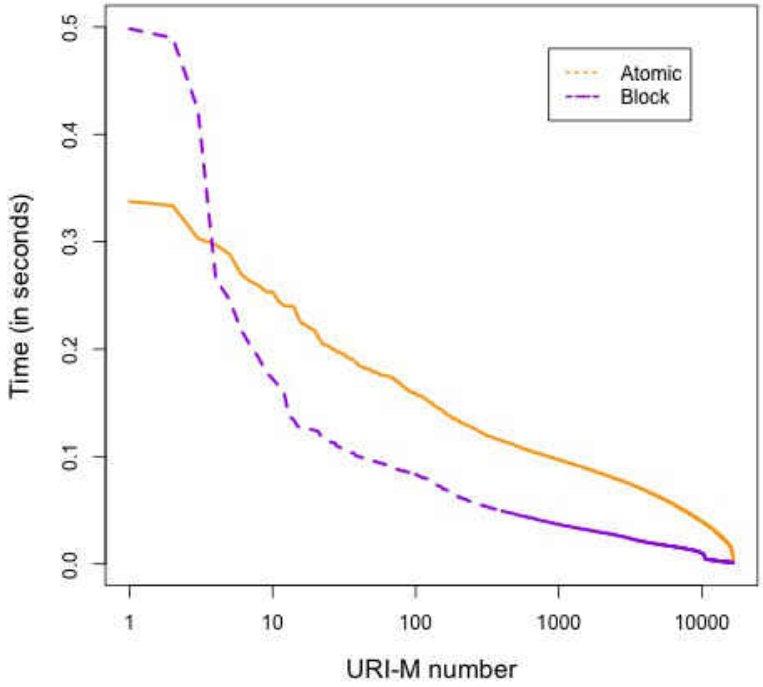


Fig. 162: Verifying mementos by both approaches.

TABLE 21: The number of verified mementos by each hashing technique with/without involving the pre-defined hashing guidelines.

With/Without guidelines	The number of mementos	Hashing technique	The number of verified mementos (based on final aggregated hash values)
Without guidelines	16,608	URIM-based hashing	8,328 (50.14%)
		Entity-based hashing	3,432 (20.66%)
		Complete hashing	2,360 (14.21%)
With guidelines	15,306	URIM-based hashing	8,069 (52.72%)
		Entity-based hashing	11,170 (72.98%)
		Complete hashing	7,457 (48.72%)
	1,302 mementos were not used for verifying the fixity because of transient errors		

```
{
    "Verified": "No",
    "time_in_seconds_to_verify_memento": 0.10734762391075492,
    "composite-memento-uri-m-hash": {
        "prev": ["74973ee7babc593c3992603f3840cbfd8c21e8309295fb134b1f7
                  224f6bfc9d8"],
        "current": "ac2fef76667ff2ade4fc6e791a79c3ee34859fb988dc580754e
                  36bb62c3631db",
        "resources": {
            "current_requested": 89,
            "current_verified": 85,
            "prev_requested": 90
        }
    },
    "composite-memento-entity-hash": {
        "prev": ["f8516f6c544b444f7791cab326c7fdd7786030d08ed7475071bce
                  6842787f0c7"],
        "current": "ea6bdb1a4bf3ccfea1adf5b0d4456084c144e9f7ac450f71fec
                    08fdb9e24f0b3",
        "resources": {
            "current_requested": 49,
            "current_verified": 48,
            "prev_requested": 48
        }
    },
    "composite-memento-overall-hash": {
        "prev": ["88c93d2e7b6025a294338e2ca9bc1086f57ee8c430fbbca71b618
                  8907051d549"],
        "current": "b06986478e9d536d57eab34b55c3c0fd4c4ca0e0a7a8b003011
                    0d0442845b0f2",
        "resources": {
            "current_requested": 89,
            "current_verified": 72,
            "prev_requested": 90
        }
    }
}
```


Fig. 163: The results of the verification process of a mementos shows the final hash values generated by the URIM-based, entity-based, and complete hashing techniques. For each hashing technique, the results also show the number of verified resources of the composite memento.

This chapter also describes the evaluation of the two approaches using 16,627 mementos and four different public web archives. The manifest generation time varies from one archive to another. For example, it takes about 15 seconds to generate fixity information on a memento from `bac-lac.gc.ca` on average, while it is 441 seconds by `Perma.cc`. We find that it takes about 1.09X and 4.54X longer to disseminate a manifest to `perma.cc`, `archive.org`, respectively, than `archive.is`, while it takes 9.20X longer to disseminate a block to `archive.org` than `perma.cc`. The *Block* approach performs 1.05X faster than the *Atomic* approach on verifying the fixity of mementos since the verification time of the fixity of a memento is 58.31 seconds by the *Atomic* approach on average and 55.32 seconds by the *Block* approach. Our study also shows that without fulfilling the hashing guidelines (defined in Chapter 4), only 50.14%, 20.66%, and 14.21% of mementos could be verified using the URIM-based, entity-based, and complete hashing technique, respectively. However, when we considered these guidelines when generating final hash values, the percentage of verified mementos increased to 52.72%, 72.98%, and 48.72%.

CHAPTER 8

CONTRIBUTIONS, FUTURE WORK, AND CONCLUSIONS

In this chapter, we summarize how we have addressed our three research questions. We also describe the contributions, future work, and conclusions.

8.1 RESEARCH QUESTIONS REVISITED

RQ1: Can we identify and quantify the types of changes on the playback of archived web pages that prevent generating repeatable fixity information?

Our goal was to introduce a framework that could be used to verify the fixity on the playback of archived web pages. In order to achieve that, we need to generate repeatable fixity information (e.g., hash values) based on the playback of mementos. Chapter 6 describes our 14-month study on a dataset of 16,627 mementos selected from 17 public web archives. Chapter 5 describes multiple methods we used to collect the dataset. We download each memento 39 times and compute their fixity information by following our initial predefined guidelines (Chapter 4.1). We identify and quantify several changes causing the same memento to have different fixity information over time. These changes may affect the HTTP status codes, the HTTP response headers and entity bodies, and the URIs-M of resources comprising an archived page.

We find that 88.45% of archived web pages produce more than one unique hash value when replayed at different times. About 16% of mementos always produce different hashes. Our study also shows that archived collections or archives may migrate to other archives. The migration process of mementos from one archive to other may cause multiple mementos to disappear.

RQ2: Given the types of changes identified in the playback of mementos, can we define the final guidelines for generating repeatable fixity information (defining an archive-aware hashing function)?

We first define initial guidelines for generating fixity information on the playback of archived web pages (Chapter 4.1), and based on the results of our study (Chapter 6) on 16k mementos, we define additional guidelines (Sections 4.2 and 6.7) for generating fixity information.

The new guidelines highlight the importance of using several aggregated hash values generated using multiple hashing techniques: URI-M-based and entity-based hashing techniques (Chapter 6.7.2). The new guidelines also indicate that the archive-aware hashing function should produce fixity information for each resource comprising the composite memento. These hash values can help us detect which resource in the composite memento has changed over time.

RQ3: How can we store and retrieve fixity information independently from the web archives from which the associated mementos are preserved?

Chapter 7 describes the framework that we use to store and discover fixity information of a memento independently from the web archive preserving the memento. The framework introduces two approaches, *Atomic* and *Block*, that can be used to disseminate fixity information to web archives and verify the fixity of mementos. 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 location before disseminating to archives.

8.2 CONTRIBUTIONS

This dissertation contributes to the field of web archiving by introducing a framework for verifying the fixity of archived web pages. Our contributions include the following:

- We defined several guidelines (Sections 4.1 and 4.2) that should be followed to generate repeatable fixity information on the playback of archived web pages [178].
- We described four methods for creating datasets of archived web pages. We used these methods to create a dataset of 16,627 mementos from 17 public web archives (Chapter 5) [170].
- We monitored and tracked 16,627 mementos for 14 months to identify and quantify types of changes on the playback of these mementos. We find that 88.45% of mementos produce more than one hash value, and only 11.55% of mementos always produce the same hash values [199].
- We found that 1002 mementos became permanently unavailable in their original archives (Chapter 6.5). Even though the original archives from which the mementos

have moved did not leave a machine readable method of locating the corresponding URIs in the new archives, we were able to manually discover the new archives to which the mementos have moved [179, 182–184].

- We found that about 54% of mementos that have moved from their original web archives, are missing. We call a memento "missing" if the values of the Memento-Datetime, the URI-R, and the final HTTP status code of the memento in the original archive are not identical to the values of the corresponding memento in the new archive or if we could not find a corresponding memento in the new archive. (Chapter 6.5) [179, 182–184].

- We created a heatmap to illustrate archive-level changes (Chapter 6.6). We used the heatmap to visualize the overall performance of each archive and to identify points in time where major changes occur, For example, through the heatmap we were able to identify archives that recently started to use a new version of their replay system .

- We introduced two hashing techniques (Chapter 6.7) for generating multiple aggregated hash values on composite mementos. These techniques are entity-based hashing technique and URI-M-based hashing technique [190].

- We developed ArchiveNow, a tool for disseminating web pages in public web archives (Chapter 7.1.2) [200, 201].

- We introduced two approaches, *Atomic* and *Block*, that can be used to disseminate fixity information to web archives and verify the fixity of mementos (Chapter 7) [191, 202].

8.3 FUTURE WORK

Future work includes the use of emerging standards, such as Web packaging [155, 156]. Web Packaging is introduced with the objective to distribute web pages packaged in a way that allows receivers to verify that the content is from a particular origin, and view the content offline. We were not able to use Web Packaging in our work because it is currently not supported by any web archive even though it may become an Internet standard in the near future [162]. Regarding our work of verifying the fixity of composite mementos, we want to explore how Web Packaging can be used by archives to deliver composite mementos. Some advantages of using web packaging might include:

- Web packaging may become an Internet standard in the near future [162]. Browsers that support web packaging can render composite mementos, without the need to load more resources from the archive.

- Using web packaging we can download a composite memento, packaged in a single file, with a single HTTP request. This should reduce playback-related changes, such as transient errors and TimeMap changes. In addition to the WARC format, web packaging can be viewed as another way of aggregating web resources in a single file. Web packaging can also be considered as replacement of WARC format in the future [160].

We want to compare our current archive-aware hashing approach with the web packaging based approach. We want to know the number (and size) of resources that each approach generates and the time required by each approach to (1) generate fixity information, and (2) verify the fixity of mementos. We also want to investigate how each approach performs in terms of generating repeatable fixity information and reducing the number of transient error and other replay-related changes.

8.4 CONCLUSIONS

We introduce a framework for checking the fixity on the playback of archived web pages. After conducting a 14-month study on 16,627 mementos from 17 public web archives, we identify and quantify changes causing the same archived pages to produce different fixity information over time. Changes may affect the HTTP status codes, the HTTP entity bodies and headers, and the URI-Ms of one or more resources comprising a composite memento. We learn that most changes are caused by JavaScript, the TimeMap inconsistency, the lack of raw mementos, transient errors, and others, which makes the process of generating repeatable fixity information complicated. We found that 88.45% of mementos produce at least two different hashes, including all mementos from seven archives. Only 11.55% of mementos always produce the same hash, while 16.06% of mementos always produce a different aggregated hash value after each replay.

Based on the results of our study, we define several guidelines for generating repeatable fixity information. These guidelines together create our archive-aware hashing function. One of the guidelines, for instance, indicates that we should generate multiple aggregated hash values on the same composite memento, and each of these aggregated hash values is generated using a different hashing techniques, such as entity-based hashing technique and

URI-M-based hashing technique.

Our framework also introduces two approaches, *Atomic* and *Block*, that we can use to disseminate fixity information to web archives. For each memento, the *Atomic* approach creates a single JSON file, or manifest, containing the fixity information. This file is then published on the web and disseminated to several on-demand web archives. The *Block* approach requires batching together fixity information of multiple archived pages to a single binary-searchable file, or block. Similar to the *Atomic* approach, the file then is published at a well-known web location before it is disseminated to multiple web archives. We use three web archives to evaluate the two approaches: `archive.org`, `perma.cc`, and `archive.is`. We find that it takes about 1.09X and 4.54X longer to disseminate a manifest to `perma.cc` and `archive.org`, respectively, than to `archive.is`, while it takes 9.20X longer to disseminate a block to `archive.org` than to `perma.cc`. The verification time of the fixity of a memento is 58.31 seconds on average using the *Atomic* approach and 55.32 seconds using the *Block* approach, so the *Block* approach performs 1.05X faster than the *Atomic* approach on verifying the fixity of memento (Chapter 7).

The framework is built based on well-known web archiving standards, such as the Memento protocol, and it does not require changes in the current web archiving infrastructure. The framework allows any user to generate fixity information on the playback of mementos at any time, preserve fixity information independently from the archive delivering the content, and verify the fixity of archived pages. Our framework allows web archives to monitor, track and verify the fixity of web archives.

REFERENCES

[1] G. Branwen, "Easy Cryptographic Timestamping." https://www.gwern.net/Timestamping, December 2015.

[2] S. G. Ainsworth, M. L. Nelson, and H. Van de Sompel, "Only one out of five archived web pages existed as presented," in *Proceedings of the 26th ACM Conference on Hypertext & Social Media (HT)*, pp. 257–266, 2015.

[3] A. Lerner, T. Kohno, and F. Roesner, "Rewriting history: Changing the archived web from the present," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, pp. 1741–1755, 2017.

[4] B. Gipp, N. Meuschke, and C. Breitinger, "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)*, 2016. https://www.gipp.com/wp-content/papercite-data/pdf/gipp2017a.pdf.

[5] J. F. Brunelle, "Zombies in the Archives." http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html, 2010.

[6] R. Kumar, "Sort-friendly URI Reordering Transform (SURT) python module." https://github.com/internetarchive/surt, 2017.

[7] I. Kreymer, "PyWb - Web Archiving Tools for All." https://github.com/ikreymer/pywb, December 2013.

[8] J. Cryer and Huddle, "Resemble.js - Image analysis and comparison." https://github.com/Huddle/Resemble.js, February 2012.

[9] M. Klein, H. Van de Sompel, R. Sanderson, H. Shankar, L. Balakireva, K. Zhou, and R. Tobin, "Scholarly context not found: One in five articles suffers from reference rot," *PloS one*, vol. 9, no. 12, 2014. e115253.

[10] Negulescu, K. C., "Web archiving @ the Internet Archive. Presentation at the 2010 Digital Preservation Partner Meeting." http://www.digitalpreservation.gov/meetings/documents/ndiipp10/NDIIPP072110FinalIA.ppt, 2010.

[11] Jackson, A. N., "UKWA Documentation." https://github.com/ukwa/ukwadocumentation/blob/master/README.md, 2018.

[12] J. Zittrain, K. Albert, and L. Lessig, "Perma: Scoping and addressing the problem of link and reference rot in legal citations," *Legal Information Management*, vol. 14, no. 2, pp. 88–99, 2014.

[13] H. Van de Sompel, M. L. Nelson, and R. Sanderson, "HTTP Framework for Time-Based Access to Resource States - Memento, Internet RFC 7089." https://doi.org/10.17487/rfc7089, 2013.

[14] M. Costa, D. Gomes, and M. J. Silva, "The evolution of web archiving," *International Journal on Digital Libraries*, vol. 18, no. 3, pp. 191–205, 2017.

[15] J. Bailey, A. Grotke, E. McCain, C. Moffatt, and N. Taylor, "Web Archiving in the United States: A 2016 Survey." http://ndsa.org/documents/WebArchivingintheUnitedStates_A2016Survey.pdf, February 2017.

[16] M. L. Nelson, "Why we need multiple web archives: The case of blog.reidreport.com." https://ws-dl.blogspot.com/2018/04/2018-04-24-why-we-need-multiple-web.html, 2018.

[17] B. Sopelsa, "MSNBC's Joy Reid apologizes for "insensitive" LGBT blog posts." https://www.nbcnews.com/feature/nbc-out/msnbc-s-joy-reid-apologizes-insensitive-lgbt-blog-posts-n826091, 2017.

[18] C. Ecarma, "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.

[19] C. Butler, "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-wayback-machine/, 2018.

[20] T. Owens, "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.

[21] PREMIS Working Group, "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. http://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf.

[22] R. L. Probert, B. Stepien, and P. Xiong, "Formal testing of web content using TTCN-3," in *Proceedings of TTCN-3 User Conference*, vol. 2005, 2005.

[23] S. Aljawarneh, C. Laing, and P. Vickers, "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)*, July 2008.

[24] P. Gao, H. Han, and T. Tokuda, "*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, ICUIMC'12*, (Kuala Lumpur, Malaysia), pp. 83:1–83:10, February 2012.

[25] J. Niu, "Functionalities of Web Archives," *D-Lib Magazine*, vol. 18, no. 3/4, 2012.

[26] The Internet Archive, "Internet Archive's Terms of Use, Privacy Policy, and Copyright Policy." https://archive.org/about/terms.php, 2014.

[27] D. R. Eltgrowth, "Best evidence and the Wayback Machine: toward a workable authentication standard for archived Internet evidence," *Fordham Law Review*, vol. 78, p. 181, 2009.

[28] J. Levine, "The Internet Archive Really Is Reliable." https://jl.ly/iarchive.html, October 2009.

[29] T. Cushing, "Federal Judge Says Internet Archive's Wayback Machine A Perfectly Legitimate Source Of Evidence." https://www.techdirt.com/articles/20160518/08175934474/federal-judge-says-internet-archives-wayback-machine-perfectly-legitimate-source-evidence.shtml, 2017.

[30] J. W. Lungstrum, "MARTEN TRANSPORT, LTD., Plaintiff, v. PLATTFORM ADVERTISING, INC, Defendant.." https://www.bloomberglaw.com/public/desktop/document/Marten_Transp_Ltd_v_PlattForm_Adver_Inc_No_142464JWL_2016_BL_1371?1462657373, 2016.

[31] A. Bright, "Web evidence points to pro-Russia rebels in downing of MH17," *The Christian Science Monitor*, 2014. https://www.csmonitor.com/World/Europe/2014/0717/Web-evidence-points-to-pro-Russia-rebels-in-downing-of-MH17.

[32] J. Lepore, "The COBWEB: Can the Internet be archived?," *New Yorker*, pp. 34–41, 2015.

[33] S. Kramarsky, "Archiving the Internet: The "Wayback Machine" in the Courts." https://www.law.com/newyorklawjournal/2018/07/16/the-wayback-machine-in-the-courts-introducing-evidence-of-the-past-state-of-the-web/, July 2018.

[34] A. Kaczynski, "Michael Flynn quietly deletes fake news tweet about Hillary Clinton's involvement in sex crimes," *CNN Politics*, 2016. http://www.cnn.com/2016/12/14/politics/kfile-flynn-deleted-tweets/.

[35] L. Qiu, "Fact Check: Trump Misleads About The Times's Reporting on Surveillance," *The New York Times*, 2017. https://www.nytimes.com/2017/03/23/us/politics/fact-check-trump-misleads-surveillance-wiretapping.html.

[36] J. Cushman and I. Kreymer, "Thinking like a hacker: Security Considerations for High-Fidelity Web Archives." http://labs.rhizome.org/presentations/security.html, May 2017.

[37] D. S. H. Rosenthal, T. Robertsoni, T. Lipkisii, V. Reichi, and S. Morabitoiii, "Requirements for Digital Preservation Systems," *D-Lib Magazine*, vol. 11, no. 11, 2005.

[38] M. L. Nelson, "Web Archives at the Nexus of Good Fakes and Flawed Originals." https://www.cni.org/events/membership-meetings/past-meetings/spring-2019/plenary-sessions-s19#closing, 2019.

[39] D. S. H. Rosenthal, "Michael Nelson's CNI Keynote: Part 1." https://blog.dshr.org/2019/06/michael-nelsons-cni-keynote-part-1.html, 2019.

[40] D. S. H. Rosenthal, "Michael Nelson's CNI Keynote: Part 2." https://blog.dshr.org/2019/06/michael-nelsons-cni-keynote-part-2.html, 2019.

[41] D. S. H. Rosenthal, "Michael Nelson's CNI Keynote: Part 3." https://blog.dshr.org/2019/06/michael-nelsons-cni-keynote-part-3.html, 2019.

[42] M. L. Nelson, "At the nexus of the CNI keynote and Rosenthal's response: "It's not an easy thing to meet your maker"." https://ws-dl.blogspot.com/2020/03/2020-03-07-at-nexus-of-cni-keynote-and.html, 2020.

[43] D. S. H. Rosenthal, "Attacking (Users Of) The Wayback Machine." https://blog.dshr.org/2017/09/attacking-users-of-wayback-machine.html, 2017.

[44] B. Tofel, "Wayback for accessing web archives," in *Proceedings of the 7th International Web Archiving Workshop (IWAW)*, pp. 27–37, 2007.

[45] "Telewizja Polska USA, Inc v. Echostar Satellite Corporation." Case No. 02 C 3293 (N.D. Ill. Oct. 14, 2004).

[46] "ST. Luke Cataract and Laser Institute V. Mark Erickson, Defendant-Counter-Claimant." Case No. 08-11848 (U.S. Court of Appeals Eleventh Circuit, Jul. 09, 2009).

[47] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–36, 2017.

[48] M. Rosenberg, "Trump Adviser Has Pushed Clinton Conspiracy Theories," *The New York Times*, 2016. https://www.nytimes.com/2016/12/05/us/politics/-michael-flynn-trump-fake-news-clinton.html.

[49] S. G. Ainsworth, M. L. Nelson, and H. Van de Sompel, "A framework for evaluation of composite memento temporal coherence," Tech. Rep. arXiv:1402.0928, 2014.

[50] Interent Archive, "Wayback CDX Server API - BETA." https://github.com/internetarchive/wayback/tree/master/wayback-cdx-server, 2019.

[51] Free Software Foundation, "GNU Wget - Introduction to GNU Wget," 2013.

[52] HTTP Archive, "State of the Web." https://httparchive.org/reports/state-of-the-web, March 2019.

[53] J. Berlin, "Squidwarc - A high fidelity archival crawler that uses Chrome or Chrome Headless." https://github.com/N0taN3rd/Squidwarc, July 2017.

[54] J. Berlin, "Replacing Heritrix with Chrome in WAIL, and the release of node-warc, node-cdxj, and Squidwarc." http://ws-dl.blogspot.com/2017/07/2017-07-24-replacing-heritrix-with.html, 2017.

[55] E. Bidelman, "Getting Started with Headless Chrome." https://developers.google.com/web/updates/2017/04/headless-chrome, 2019.

[56] T. Berners-Lee, "Information Management: A Proposal." https://www.w3.org/History/1989/proposal.html, 1989.

[57] I. Jacobs and N. Walsh, "Architecture of the World Wide Web." https://www.w3.org/TR/webarch/, 2004.

[58] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1, Internet RFC 2616." http://tools.ietf.org/html/rfc2616, 1999.

[59] R. T. Fielding and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, Internet RFC 7230." https://doi.org/10.17487/rfc7230, 2014.

[60] R. T. Fielding and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Internet RFC 7231."

[61] R. T. Fielding and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests, Internet RFC 7232." https://doi.org/10.17487/rfc7232, 2014.

[62] R. T. Fielding, Y. Lafon, and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Range Requests, Internet RFC 7233." https://doi.org/10.17487/rfc7233, 2014.

[63] R. T. Fielding, M. Nottingham, and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching, Internet RFC 7234." https://doi.org/10.17487/rfc7234, 2014.

[64] R. T. Fielding and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication, Internet RFC 7235." https://doi.org/10.17487/rfc7235, 2014.

[65] "curl - command line tool and library for transferring data with URLs." https://curl.haxx.se/, 2020.

[66] K. Holtman and A. H. Mutz, "Transparent Content Negotiation in HTTP, Internet RFC 2295." https://doi.org/10.17487/rfc2295, 1998.

[67] T. Berners-Lee and D. W. Connolly, "Hypertext Markup Language - 2.0, Internet RFC 1866." http://tools.ietf.org/html/rfc1866, 1995.

[68] P. L. Hegaret, "Document Object Model (DOM)." https://www.w3.org/DOM/, 2005.

[69] B. Kahle, " Wayback Rising! now 731,667,951,000 web objects (counting images and pages) active on https://web.archive.org . 731 billion! Thank you for all the support, it makes a difference. go @internetarchive." https://twitter.com/brewster_kahle/status/1118172506777509890, April 2019.

[70] S. Bailey and D. Thompson, "Building the UK's First Public Web Archive," *D-Lib Magazine*, vol. 12, no. 1, 2006.

[71] Search Console Help, "Googlebot." https://support.google.com/webmasters/answer/182072, 2019.

[72] K. Sigurdsson, "Incremental crawling with Heritrix," in *Proceedings of the 5th International Web Archiving Workshop (IWAW)*, 2005.

[73] International Organization for Standardization, "WARC file format," no. ISO 28500:2017, 2017. https://www.iso.org/standard/68004.html.

[74] M. Kelly and M. C. Weigle, "WARCreate: Create Wayback-Consumable WARC Files from Any Webpage," in *Proceedings of the 12th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 437–438, 2012.

[75] H. Van de Sompel, M. L. Nelson, R. Sanderson, L. L. Balakireva, S. Ainsworth, and H. Shankar, "Memento: Time Travel for the Web," Tech. Rep. arXiv:0911.1112, 2009.

[76] "OpenWayback Users." https://github.com/iipc/openwayback/wiki/OpenWayback-Users, 2017.

[77] "Public Projects using pywb." https://github.com/webrecorder/pywb/wiki/Public-Projects-using-pywb, 2018.

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

[79] A. N. Jackson, "The CDX file format." https://iipc.github.io/warc-specifications/specifications/cdx-format/cdx-2015/, 2015.

[80] Sawood Alam, "CDXJ: An Object Resource Stream Serialization Format." http://ws-dl.blogspot.com/2015/09/2015-09-10-cdxj-object-resource-stream.html, September 2015.

[81] S. Alam, M. L. Nelson, H. Van de Sompel, and D. S. H. Rosenthal, "Web archive profiling through fulltext search," in *Proceedings of Theory and Practice of Digital Libraries (TPDL)*, pp. 121–132, 2016.

[82] Rossi, Alexis, "If You See Something, Save Something - 6 Ways to Save Pages In the Wayback Machine." https://blog.archive.org/2017/01/25/see-something-save-something/, January 2017.

[83] M. L. Nelson, "Wayback Machine Upgrades Memento Support." https://ws-dl.blogspot.com/2013/07/2013-07-15-wayback-machine-upgrades.html, 2013.

[84] J. A. Berlin, "To Relive the Web: A Framework for the Transformation and Archival Replay of Web Pages," Master's thesis, Old Dominion University, 2018.

[85] M. Kelly, J. F. Brunelle, M. C. Weigle, and M. L. Nelson, "On the change in archivability of websites over time," in *Proceedings of Theory and Practice of Digital Libraries (TPDL)*, pp. 35–47, 2013.

[86] J. F. Brunelle, M. Kelly, M. C. Weigle, and M. L. Nelson, "The impact of JavaScript on archivability," *International Journal on Digital Libraries*, vol. 17, no. 2, pp. 95–117, 2016.

[87] HTTP Archive, "State of JavaScript." https://httparchive.org/reports/state-of-javascript, June 2020.

[88] S. Withee, "First, right off the bat sites seem to fall into 3 categories: - Blank white screens (45%) - Sites with some things but most content doesn't load (50%) - Sites that still mostly work (extremely rare, maybe 4%) - Sites that fully work (less than 1%) (2/11)." https://twitter.com/geekygirlsarah/status/1260409691030663171, May 2020.

[89] International Internet Preservation Consortium (IIPC), "OpenWayback." https://iipc.github.io/openwayback/2.1.0.RC.1/administrator_manual.html, 2015.

[90] I. Kreymer, "Rewriter." https://github.com/webrecorder/pywb/blob/master/docs/manual/rewriter.rst, 2018.

[91] S. M. Jones and H. Shankar, "Rules of acquisition for mementos and their content," Tech. Rep. arXiv:1602.06223, 2016.

[92] S. M. Jones, H. Van de Sompel, and M. L. Nelson, "Mementos in the Raw." http://ws-dl.blogspot.com/2016/04/2016-04-27-mementos-in-raw.html, 2016.

[93] H. Van de Sompel, M. L. Nelson, L. Balakireva, M. Klein, S. M. Jones, and H. Shankar, "Mementos In the Raw, Take Two." `http://ws-dl.blogspot.com/2016/08/2016-08-15-mementos-in-raw-take-two.html`, 2016.

[94] M. Nottingham, "Web Linking, Internet RFC 8228." `http://tools.ietf.org/html/rfc8228`, 2017.

[95] Internet Assigned Numbers Authority (IANA), "Link Relation." `https://www.iana.org/assignments/link-relations/`, August 2005.

[96] N. J. Bornand, L. Balakireva, and H. Van de Sompel, "Routing Memento requests using binary classifiers," in *Proceedings of the 16th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 63–72, 2016.

[97] S. Alam and M. L. Nelson, "MemGator-A portable concurrent memento aggregator: Cross-platform CLI and server binaries in Go," in *Proceedings of the 16th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 243–244, 2016.

[98] S. Alam, "A Memento Aggregator CLI and Server in Go." `https://github.com/oduwsdl/MemGator`, 2016.

[99] UK Web Archive, "Mementos - Finding historical archives across the world wide web.." `https://www.webarchive.org.uk/mementos/search`, 2019.

[100] A. Alsum, M. C. Weigle, M. L. Nelson, and H. Van de Sompel, "Profiling web archive coverage for top-level domain and content language," *International Journal on Digital Libraries*, vol. 14, no. 3-4, pp. 149–166, 2014.

[101] A. Alsum, M. C. Weigle, M. L. Nelson, and H. Van de Sompel, "Profiling web archive coverage for top-level domain and content language," in *Proceedings of Theory and Practice of Digital Libraries (TPDL)*, pp. 60–71, 2013.

[102] J. F. Brunelle and M. L. Nelson, "An evaluation of caching policies for Memento TimeMaps," in *Proceedings of the 13th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 267–276, 2013.

[103] R. L. Dale and B. Ambacher, "Trustworthy repositories audit and certification: Criteria and checklist. Report of the RLG-NARA Task Force on Digital Repository Certification." `http://www.crl.edu/sites/default/files/d6/attachments/pages/trac_0.pdf`, 2007.

[104] T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA), Internet RFC 4634." `http://tools.ietf.org/html/rfc4634`, 2006.

[105] Chumbley, Alex and Moore, Karleigh and Khim, Jimin, "Merkle Tree." `https://brilliant.org/wiki/merkle-tree/`, 2020.

[106] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing*, pp. 380–388, 2002.

[107] G. S. Manku, A. Jain, and A. Das Sarma, "Detecting near-duplicates for web crawling," in *Proceedings of the 16th International World Wide Web Conference*, pp. 141–150, ACM, 2007.

[108] S. Evans, "A Python Implementation of Simhash Algorithm." `https://github.com/leonsim/simhash`, 2015.

[109] A. Z. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings* (B. Carpentieri, A. D. Santis, U. Vaccaro, and J. A. Storer, eds.), pp. 21–29, 1997.

[110] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004* (J. Snoeyink and J. Boissonnat, eds.), pp. 253–262, 2004.

[111] A. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th International Conference on*

World Wide Web, WWW, Banff, Alberta, Canada, May 8-12, 2007 (C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, eds.), pp. 271–280, 2007.

[112] M. R. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006* (E. N. Efthimiadis, S. T. Dumais, D. Hawking, and K. Järvelin, eds.), pp. 284–291, 2006.

[113] A. Shrivastava and P. Li, "In defense of minhash over simhash," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, vol. 33 of *JMLR Workshop and Conference Proceedings*, pp. 886–894, JMLR.org, 2014.

[114] N. Krawetz, "The Hacker Factor Blog." `http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html`, 2011.

[115] E. Klinger and D. Starkweather, "pHash- The open source perceptual hash library." `https://www.phash.org`, 2020.

[116] L. Daigle, "WHOIS Protocol Specification, Internet RFC 3912." `http://tools.ietf.org/html/rfc3912`, 2004.

[117] J. Cushman, "WARCgames." `https://github.com/harvard-lil/warcgames`, May 2017.

[118] D. S. H. Rosenthal, "WAC2017: Security Issues for Web Archives." `https://blog.dshr.org/2017/06/wac2017-security-issues-for-web-archives.html`, 2017.

[119] I. Kreymer, "Webrecorder - a web archiving platform and service for all." `https://webrecorder.io`, 2015.

[120] "PhantomJS - Scriptable Headless Browser." `https://phantomjs.org/`, 2020.

[121] E. Rescorla, "HTTP Over TLS, Internet RFC 2818." `http://tools.ietf.org/html/rfc2818`, 2000.

[122] E. Rescorla and A. M. Schiffman, "The Secure HyperText Transfer Protocol, Internet RFC 2660." `http://tools.ietf.org/html/rfc2660`, 1999.

[123] J. E. Smith and R. Nair, "The architecture of virtual machines," *IEEE Computer*, vol. 38, no. 5, pp. 32–38, 2005.

[124] T. Kuhn and M. Dumontier, "Trusty URIs: Verifiable, immutable, and permanent digital artifacts for linked data," in *European Semantic Web Conference*, pp. 395–410, Springer, 2014.

[125] T. Kuhn and M. Dumontier, "Making digital artifacts on the web verifiable and reliable," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2390–2400, 2015.

[126] S. P. Kurian and V. S. Sekhar, "Distributed digital artifacts on the semantic web," *International Journal of Computer Applications Technology and Research*, vol. 5, pp. 99–103, 2016.

[127] G. Carothers, "RDF 1.1 N-Quads," *World Wide Web Consortium*, 2014. `https://www.w3.org/TR/n-quads/`.

[128] C. Bizer and R. Cyganiak, "RDF 1.1 TriG," *World Wide Web Consortium*, 2014. `https://www.w3.org/TR/trig/`.

[129] A. Swartz, "Application/RDF+XML Media Type Registration, Internet RFC 3870." `https://doi.org/10.17487/rfc3870`, 2004.

[130] R. Cyganiak, D. Wood, and M. Lanthaler, "RDF 1.1 Concepts and Abstract Syntax," *World Wide Web Consortium*, 2014. `https://www.w3.org/TR/rdf11-concepts/`.

[131] T. Kuhn, "trustyuri-python." `https://github.com/trustyuri/trustyuri-python`, 2014.

[132] T. Kuhn, "trustyuri-java." `https://github.com/trustyuri/trustyuri-java`, 2020.

[133] T. Kuhn, "trustyuri-perl." `https://github.com/trustyuri/trustyuri-perl`, 2014.

[134] M. Aturban, M. L. Nelson, and M. C. Weigle, "Quantifying orphaned annotations in hypothes.is," *Proceedings of Theory and Practice of Digital Libraries (TPDL)*, pp. 15–27, 2015.

[135] J. Benet, "Multihash." `https://github.com/multiformats/multihash`, 2017.

[136] Protocol Labs, "IPFS is the Distributed Web." `https://ipfs.io`, 2017.

[137] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf, 2008.

[138] A. Wright and P. De Filippi, "Decentralized blockchain technology and the rise of lex cryptographia," *SSRN Electronic Journal 2580664*, 2015.

[139] B. Gipp, "Free trusted timestamping." https://originstamp.org, 2019.

[140] W. Vaughan, J. Bukowski, and S. Wilkinson, "Chainpoint-A Scalable Protocol for Anchoring Data in the Blockchain and Generating Blockchain Receipts." https://github.com/chainpoint/whitepaper/blob/master/chainpoint_white_paper.pdf, 2016.

[141] ChainPoint, "ChainPoint." https://www.chainpoint.com, 2020.

[142] P. Todd, "OpenTimestamps." https://opentimestamps.org/, 2017.

[143] P. Todd, "OpenTimestamps: Scalable, Trust-Minimized, Distributed Timestamping with Bitcoin." https://petertodd.org/2016/opentimestamps-announcement, 2016.

[144] B. Gipp, N. Meuschke, and A. Gernandt, "Decentralized trusted timestamping using the crypto currency Bitcoin," Tech. Rep. arXiv:1502.04015, 2015.

[145] C. Adams and C. Renaud, "Proof of Existence." https://github.com/proofofexistence/proofofexistence, 2020.

[146] R. Merkle, *Secrecy, authentication and public key systems*. PhD thesis, Stanford University, 1979.

[147] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.

[148] J. Collomosse, T. Bui, A. Brown, J. Sheridan, A. Green, M. Bell, J. Fawcett, J. Higgins, and O. Thereaux, "ARCHANGEL: Trusted Archives of Digital Public Documents," Tech. Rep. arXiv:1804.08342, April 2018.

[149] "Bibliotheca Alexandrina." https://www.bibalex.org/en/default, 2019.

[150] K. Barrett, "FAQs about the Internet Archive Canada." https://blog.archive.org/2016/12/03/faqs-about-the-internet-archive-canada/, 2016.

[151] L. Bailey, "Brief of Internet Archive Canada and the Internet Archive to the Standing Committee on Industry, Science and Trade (Indu) Pursuant to the Statutory Review of the Copyright Act." https://www.ourcommons.ca/Content/Committee/421/INDU/Brief/BR10268191/br-external/InternetArchiveCanada-e.pdf, December 2018.

[152] V. Reich and D. S. H. Rosenthal, "LOCKSS: A permanent web publishing and access system," *D-Lib Magazine*, vol. 7, no. 6, 2001.

[153] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker, "The LOCKSS peer-to-peer digital preservation system," *ACM Transactions on Computer Systems (TOCS)*, vol. 23, no. 1, pp. 2–50, 2005.

[154] D. Gomes, J. Miranda, and M. Costa, "A survey on web archiving initiatives," *Proceedings of Theory and Practice of Digital Libraries (TPDL)*, pp. 408–420, 2011.

[155] J. Yasskin, "Packaging Websites." https://github.com/WICG/webpackage, 2017.

[156] J. Yasskin, "Use Cases and Requirements for Web Packages." https://tools.ietf.org/id/draft-yasskin-webpackage-use-cases-00.html, 2017.

[157] J. Yasskin, "Signed HTTP Exchanges." https://wicg.github.io/webpackage/draft-yasskin-http-origin-signed-responses.html, 2019.

[158] J. Yasskin, "Bundled HTTP Exchanges." https://wicg.github.io/webpackage/draft-yasskin-wpack-bundled-exchanges.html, 2019.

[159] J. Yasskin, "Loading Signed Exchanges." https://wicg.github.io/webpackage/loading.html, 2019.

[160] A. Jackson, "This is interesting. The web packaging format is coming along nicely, and may be a useful complement or even replacement for WARC. (link: https://github.com/WICG/webpackage/blob/master/explainer.md) github.com/WICG/webpackag...." https://twitter.com/anjacks0n/status/950861384266416134, Jan 2018.

[161] S. Alam, M. C. Weigle, M. L. Nelson, M. Klein, and H. Van de Sompel, "Supporting Web Archiving via Web Packaging," Tech. Rep. arXiv:1906.07104, June 2019.

[162] J. Yasskin, "Web Packaging." https://tools.ietf.org/id/draft-yasskin-dispatch-web-packaging-00.html, June 2017.

[163] "Archive.is blog - Blog of http://archive.is/ project." https://blog.archive.today/post/618635148292964352/what-scraper-or-headless-browser-are-you-using-it, May 2020.

[164] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins, "Sic transit gloria telae: Towards an understanding of the web's decay," in *Proceedings of the 13th International Conference on World Wide Web*, pp. 328–337, 2004.

[165] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on circuits and systems for video technology*, vol. 11, no. 3, pp. 301–317, 2001.

[166] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, and M. Dietz, "ISO/IEC MPEG-2 advanced audio coding," *Journal of the Audio engineering society*, vol. 45, no. 10, pp. 789–814, 1997.

[167] D. S. H. Rosenthal, T. Lipkis, T. S. Robertson, and S. Morabito, "Transparent Format Migration of Preserved Web Content," *D-Lib Magazine*, vol. 11, no. 1, 2005.

[168] K. Collins, "How Adobe Flash, once the face of the web, fell to the brink of obscurity and why it's worth saving." https://qz.com/863467/how-adobe-flash-once-the-face-of-the-web-fell-to-the-brink-of-obscurity-and-why-its-worth-saving/, December 2016.

[169] Free Software Foundation, "Why There Are No GIF Files on GNU Web Pages." https://www.gnu.org/philosophy/gif.html, 2006.

[170] M. Aturban, M. L. Nelson, M. C. Weigle, M. Klein, and H. Van de Sompel, "Collecting 16K archived web pages from 17 public web archives," Tech. Rep. arXiv:1905.03836, May 2019.

[171] "The Moz Top Pages." https://moz.com/top500, 6 2017. Accessed on 2017 June 8.

[172] J. F. Brunelle, M. Kelly, H. SalahEldeen, M. C. Weigle, and M. L. Nelson, "Not all Mementos are created equal: Measuring the impact of missing resources," *International Journal on Digital Libraries*, vol. 16, no. 3-4, pp. 283–301, 2015.

[173] "The HTTP Archive Tracks How the Web is Built." https://httparchive.org/downloads.php, 2017.

[174] N. Ruest, I. Milligan, R. Deschamps, J. Lin, and Library and Archives Canada, "Web Archives for Historical Research Group Dataverse (WAHR)." https:/dataverse.scholarsportal.info/dataverse/wahr, 2017.

[175] Cremona, Rebecca Lynn, "Remove pywb playback; reimplement memento support #2699." https://github.com/harvard-lil/perma/pull/2699, 2020.

[176] R. Viscomi, "Are HTTP(S) requests that return 4xx/5xx counted?." https://discuss.httparchive.org/t/are-http-s-requests-that-return-4xx-5xx-counted/1640, April 2019.

[177] J. F. Brunelle, M. C. Weigle, and M. L. Nelson, "Archival Crawlers and JavaScript: Discover More Stuff but Crawl More Slowly," in *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 1–10, IEEE, 2017.

[178] M. Aturban, M. L. Nelson, and M. C. Weigle, "Difficulties of Timestamping Archived Web Pages," Tech. Rep. arXiv:1712.03140, December 2017.

[179] M. Aturban, "Where did the archive go? Part 1: Library and Archives Canada." https://ws-dl.blogspot.com/2019/08/2019-08-30-where-did-archive-go-part1.html, 2019.

[180] Jae Duk Seo, "Simple-Merkle-Tree-in-Python." https://github.com/JaeDukSeo/Simple-Merkle-Tree-in-Python, 2017.

[181] Cloudflare Support, "What is Email Address Obfuscation?." https://support.cloudflare.com/hc/en-us/articles/200170016-What-is-Email-Address-Obfuscation-.

[182] M. Aturban, "Where did the archive go? Part 2: National Library of Ireland." https://ws-dl.blogspot.com/2019/09/2019-09-10-where-did-archive-go-part-2.html, 2019.

[183] M. Aturban, "Where did the archive go? Part 3: Public Record Office of Northern Ireland." https://ws-dl.blogspot.com/2019/09/2019-09-25-where-did-archive-go-part-3.html, 2019.

[184] M. Aturban, "Where did the archive go? Part 4: WebCite." https://ws-dl.blogspot.com/2019/10/2019-10-21-where-did-archive-go-part-4.html, 2019.

[185] Internet Archive, "Internet Archive's Terms of Use, Privacy Policy, and Copyright Policy." https://archive.org/about/terms.php, December 2014.

[186] K. Perisic, "Internet Archive Removes Evidence That Companies Sold Stalkerware, but Not ISIS Propaganda." https://dailycaller.com/2018/05/23/internet-archive-flexispy/, May 2018.

[187] M. L. Nelson, "REST, HATEOAS, and Follow Your Nose." https://ws-dl.blogspot.com/2013/11/2013-11-19-rest-hateoas-and-follow-your.html, 2013.

[188] The Apache Software Foundation, "Apache HTTP Server Project." https://httpd.apache.org/, 2019.

[189] The Apache Software Foundation, "Apache Module mod_rewrite." https://httpd.apache.org/docs/current/mod/mod_rewrite.html, 2019.

[190] M. Aturban, "Comparing the three hasing techniques." https://github.com/oduwsdl/mementos-fixity/tree/master/hashing_techniques, 2020.

[191] M. Aturban, S. Alam, M. L. Nelson, and M. C. Weigle, "Archive Assisted Archival Fixity Verification Framework," in *Proceedings of the 19th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 162–171, 2019.

[192] D. S. H. Rosenthal, "SHA1 is dead." https://blog.dshr.org/2017/03/sha1-is-dead.html, 2017.

[193] M. Aturban, M. Kelly, A. Sawood, J. Berlin, M. L. Nelson, and M. C. Weigle, "Archivenow: Simplified, extensible, multi-archive preservation," in *Proceedings of the 18th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 321–322, 2018.

[194] S. Alam, "Unified Key Value Store." https://github.com/oduwsdl/ORS/blob/master/ukvs.md, January 2019.

[195] S. Alam, M. C. Weigle, and M. L. Nelson, "MementoMap framework for flexible and adaptive web archive profiling," in *Proceedings of the 19th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 172–181, 2019.

[196] A. Narayanan and J. Clark, "Bitcoin's Academic Pedigree: The concept of cryptocurrencies is built from forgotten ideas in research literature," *ACM Queue*, vol. 15, no. 4, 2017.

[197] K. Sigurðsson, M. Stack, and I. Ranitovic, "Heritrix User Manual: Sort-friendly URI Reordering Transform." `http://crawler.archive.org/articles/user_manual/glossary.html#surt`, 2006.

[198] S. Alam, "HTTP Mailbox - Asynchronous Restful Communication," Master's thesis, Old Dominion University, 2013.

[199] M. Aturban, "Mementos Fixity." `https://github.com/oduwsdl/mementos-fixity`, 2019.

[200] M. Aturban, M. Kelly, S. Alam, J. A. Berlin, M. L. Nelson, and M. C. Weigle, "ArchiveNow: Simplified, Extensible, Multi-Archive Preservation," in *Proceedings of the 18th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 321–322, 2018.

[201] M. Aturban, "Archivenow - A Tool To Push Web Resources Into Web Archives." `https://github.com/oduwsdl/archivenow`, February 2017.

[202] M. Aturban and S. Alam, "Fixity Manifest Server." `https://github.com/oduwsdl/manifest`, 2019.

VITA

Mohamed Aturban
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

EDUCATION

Ph.D. Computer Science, Old Dominion University, 2020

M.S. Computer Science, New Mexico State University, 2011

B.S. Computer Science, University of Tripoli, Libya, 2002

EMPLOYMENT

2016 - 2020 Research Assistant at Old Dominion University, USA

2015 - 2016 Teaching Assistant at Old Dominion University, USA

2011 - 2012 Research/Teaching Assistant at New Mexico State University, USA

2003 - 2008 Teaching Assistant at University of Tripoli, Libya

2003 - 2006 Software Engineer at Tatweer (Acer Authorized Service Provider), Libya

PUBLICATIONS

A complete list is available at `scholar.google.com/citations?user=XdGvjXEAAAAJ&hl=en&oi=ao`

Typeset using LaTeX.