CHAPTER 2

BACKGROUND AND STATE OF THE ART

Understanding why the goals of this research would be useful requires one to examine the state of archiving as a whole, particularly in the non-disjoint realms of personal digital archiving, web archiving and personal web archiving. Each of these has outstanding tasks to be resolved or considered, many of which are intractable. The importance of preserving digital content lies in that the content is largely ephemeral. Missing web pages, for example, are ubiquitous in today’s browsing experience [24]. Efforts like the Firefox add-on “Synchronicity” that support the user in (re-)discovering missing webpages [23] through access to other archives and caches would be more effective if more content were preserved. Preserving more content in the ways considered to be “best practice” enables recollection of a digital resource once it is discovered and in need of retrieval. Various software endeavors and their respective services (e.g. IA’s Wayback) have driven forward momentum of digital archiving but not all aspects have been translated over to personal digital archiving.

One of the advances in recent web archiving practices is the standardization and adoption of the WARC [19] format by the Internet Archive. While some attributes and merits of this format are discussed in Sections 1.1 and 2.3, the intention of this work is to be applied independent of any certain format. Because the WARC format is considered the current standard, however, this paper assumes conformance to be a prerequisite in obtaining satisfactory preservation of digital objects.

2.1 BACKGROUND IN PERSONAL WEB ARCHIVING

Three realms of applicability for this research are personal digital archiving, web archiving and personal web archiving. Issues that reside in a more specific realm might find resolve in more general realms. This is especially the case in realms that are encapsulated by another (e.g. personal web archiving within the realm of web archiving). Because of this, the discussion of these realms will get progressively less abstract.
2.1.1 State of Personal Digital Archiving

Personal digital archiving spans a wide range of applications from assuring that stored content is well-backed up and easily findable to verifying that the information can be recovered if an original is ever lost. In the context of digital assets, many are aware of good practice yet few institute a consistent backup regimen or only backup on an ad hoc basis [28]. Still then, users assume that the ethos of the LOCKSS [27] (Lots of Copies Keeps Stuff Safe) system is sound but until a resource needs to be accessed, the availability of the resource is often not verified. LOCKSS’ ethos rather than the system itself is alluded to here because of its unlikely pertinence to personal web archiving (it was built to ensure continued access to scientific publications) though its emphasis on redundancy is largely the rationale that laymen take in ensuring preservation. An individual is seldom aware of this digital brinkmanship [28] and in practice, few do much to hedge against this loss [30] in which a LOCKSS-like system would be helpful in preventing. Individuals frequently use sub-par strategies like using system backups as archives, moving files from one machine to another, moving files to another medium (e.g. CDs, floppy disks) as their archiving strategy [1], all of which are problematic and reduce the ability to recall information when needed. Sound personal archiving practices are not commonplace both because users are seldom able to implement their current strategies consistently [1] and many of their strategies give a false assurance of their soundness due to user ignorance.

With proper life cycle management, digital objects are more likely to be preserved. Best practices in preserving digital content come down to a system of proper creation, acquisition, cataloging/identification, storage, preservation and access. [18] For example, practices used when a digital object is created ultimately impact the ease with which the object can be digitally archived and preserved. [18] Further, best practice requires that metadata is created as objection-creation stage [18] and that the archiving process is made more efficient when attention is paid to issues of consistency, format, standardization and metadata description in the very beginning of the information life cycle. [18]

2.1.2 State of Web Archiving

Generally speaking, web archiving consists of creating archives of any content that resides on the web. The Internet Archive et al. have been major players in facilitating
the preservation of content on the web and assuring metadata is attributed. The Internet Archive and Nordic National Libraries created a web crawler, Heritrix [42], to crawl websites for inclusion into the Wayback Machine. The tool is written in Java and was originally created with the intention of being open source so as to promote collaboration between institutions that were interested in archiving the web [42]. Prior to their efforts, no one had tried to capture a comprehensive record of the text and images contained in the documents that appeared on the web [20]. For content that has not been successfully archived by a particular organization, methods like utilizing the lexical signatures of lost web pages [22] to find content not in an archive, referencing other archiving institutions to supplement periods of time when a page was not archived [52], and referring to search engines caches that were created as a result of the indexing process [37, 40] can be used to restore and repair incompletely preserved content.
FIG. 2: Replaying the most recently archived version of Craigslist returns the unexpected result of the crawler's original locale instead of the user's current locale.

Little has been done in the way of assuring that an archive is replayed in the manner originally intended. Archives generated from Heritrix are replayed as if being viewed by Heritrix. An example of this problem can be observed with the Internet Archive's crawler of Craigslist\(^1\). The archived version retained by Heritrix of Craigslist is based on the perspective of the crawler, i.e. Heritrix as run from the Internet Archive in San Francisco. A user that wished to recall the content on craigslist.org (Figure 2 upper left) at a certain date would not have the luxury of simply entering the domain but instead is required to be familiar with the site-specific redirection scheme (Figure 2 progression from top left clockwise). This is one

\(^1\)http://craigslist.org
example where WYSIWYG archiving not being maintained potentially compromises the content targeted to be archived. When originally generated, code that appeals to the user's perspective (in this case, the web crawler; in other cases, the user's web browser) may result in unexpected output on replay. Emulation is needed so the replay of web resources may exactly imitate legacy software [33].

2.1.3 State of Personal Web Archiving

Curation of personal digital materials in online storage bears some striking similarities to the curation of similar materials stored locally [30]. The aforementioned neglect stems from a lack of current need for resource recall, the inability to sufficiently archive because of an unsound or poorly implemented processes and no standard medium to assure that the output format will be able to be read in the future. These are only a few of the numerous reasons why the practice of web archiving and particularly personal web archiving is in disarray. Users will often use the circular reasoning of a service supplying the backup for data they have stored on the web yet resort to poor archiving practices in assuring that the content is preserved. For example, pictures on photo sharing websites\(^2\) retain more metadata than those stored in a directory on a hard drive. While one usually will reference the more comprehensive collection of photos on a local machine, in the event of system failure, only then, will the attempt to recover photos from the photo sharing website. Little redundancy is put in place and where it is, the integrity of data being backed up is rarely verified until the data has been lost. Users are unwilling to put forth any curatorial effort to ensure their work is not lost [30].

The extent to which a digital object is preserved corresponds to the methods and medium used to accomplish the preservation. People archive their personal digital belongings by relying on a combination of benign neglect, sporadic backups and unsystematic file replication [30]. Tools created to alleviate the process are only as reliable as the methods and medium that the tools employ. Archive Facebook, for instance, preserves content in a directory structure navigable by a web browser. While this content is saved to disk and retained, without appropriate measures (\textit{e.g.} storing of metadata, appealing to standardized preservation formats) and a standard medium (the tool suffers from the issue describes in Section 2.2.3), the content is not sufficiently preserved.

\(^2\)\textit{e.g. http://www.flickr.com}
2.2 CONCERNS UNIQUE TO PERSONAL WEB ARCHIVING OF THIS NATURE

Personal web archiving exhibits some features not present in conventional web archiving that should be addressed. Considering these prior to moving forward with new methods will allow the methods to have a foundation in both accomplishing their task while taking into account some of the caveats that are likely to come about. These caveats arise when the task of personal web archiving is addressed in a naive manner similar to conventional web archiving. Though there are likely to be further concerns as the processes of archiving this content matures in the future, the issues of URI bleed over, authentication, and privacy and security in regards to personal web archiving will be addressed. Topics relevant to web archiving in general that will be discussed are of archive integrity, archiving vs. backing up and the inconsistency in methods of obtaining data that plague web archiving as a whole.

2.2.1 Bleed Over

Once a user is authenticated with a target website, the contents of a single URI will likely vary from user to user. In the sample case of Facebook, a user’s news feed is composed of the contents of the recent updates from a respective user’s friends. Being that it is unlikely that two users have an identical set of friends (and even if they do), the contents of two users’ news feeds will be different while still being accessed with the same URI (i.e. www.facebook.com). Much of the content on the web is assumed to contain the same data when accessed by two different users. Even content within the Deep Web [7] with the same URI and explicit (GET request) parameters will likely result in identical content, especially if the process of accessing the content has no side effects. As content on the web becomes more dynamically generated [25, 45], hidden behind web forms and other kinds of query interfaces [54], it becomes less accessible to crawlers.\(^3\) For example, if two users access a URI that is not indexed by search engines (through robots.txt enforcement or simply because it is not linked from elsewhere) residing at http://www.example.com/?key=secretKey, the content could be different because consistency of content between accesses is not guaranteed contrary to the specification stating that a URI will consistently return

\(^3\)Though content on social media websites fits Raghavan’s definition of “dynamic” [45] and thus is the target for the crawler, it is not, “form generated” and so contradictorily does not fit the criteria.
the same content [26]. This is due to pages including code that executes on the client machine to retrieve further tailored content [45] (frequently implemented as AJAX calls). If this content requires authentication, it is still possible that the content will remain the same between different users, but it is the nature of social websites to tailor content to its users. Even if the content when accessing http://www.example.com/?key=secretKey while authenticated is nearly identical, any tailoring to the user (even a content variation as subtle as a “Hello Username” message) will result in unique content. One cannot assume that a URI will result in the same content or even the same design when accessed by different users. Figure 3 shows two different Facebook users’ returned content after having accessed the same URI (http://www.facebook.com/profile.php?sk=info). Normally, only the content would be tailored to the user but because of the website’s personalization, the page is completely different though temporally equivalent. On the surface web, only a temporal difference would result in this drastic of a difference.

FIG. 3: URIs can not be used to guarantee what content is returned when different users access the URI because of site personalization.

Less subtle differences in the content displayed to different users when accessing the same URI are becoming more commonplace as websites enable users to control privacy settings. By using these privacy settings, a user is able to restrict or allow specified content to be displayed with a scope that can span any of “publicly accessible”, “only friends”, “only me” or any ad hoc subset. Though two users might be friends (in the context of social media websites) with a third, the common friend can potentially tailor what each user sees on a per user basis. This implies that, for the most part, the content displayed to a user from a system that allows tailoring, is almost always guaranteed to be unique.

As the content can greatly vary between users and the same URI when accessed by different users can result in different content; in order to archive the content from
the perspective of a user, some unobtrusive identifier must be added to the means of accessing the archive to assure that the content from the desired perspective is served. Such an identifier will assure that when two users access the same URI, a secondary identifier will provide the facility to serve the user the appropriate concept.

One way to accomplish this requirement of assuring consistent access to a unique archive is to add “perspective” data to the URI in the location where the username would normally reside\(^4\) [26]. The overloading of this attribute remains semantic, as the “username” field in the URI scheme is still representative of a unique identifier but expanded to be a signifier to other tools to handle URIs with this addition differently. In the case of http, this attribute is rarely included in the user scheme and different means of authentication are normally used (e.g. requiring credentials on access of the URI that does not contain the username:password@urischeme form). An implementation-based issue with this approach is that in the default implementation (at archive.org), Wayback Machine’s crawler (Heritrix) uses a filter [3] to strip away certain information including the username field preceding the hostname. Because of this, a different approach must be used to accomplish perspective designation of an archive. This is discussed further in section 2.2.2. As is evidenced here, Heritrix’s default functionality is too generic to be applied to personal web archiving of content requiring authentication. The lack of context and stripping of external means to represent this context make it unwieldy for a casual user. A user that who would like to archive certain content on a specific website would need to hold the crawler’s hand via explicitly defining URIs to crawl.

### 2.2.2 Context

When viewing a webpage from various perspectives, be it one of the variety of web browsers available or from different devices (e.g. mobile phone, PC), it is possible that different content is displayed based on the user’s choice of device (Figure 4). Some websites serve a completely separate, often optimized version of a website to those on mobile devices that might be restricted by bandwidth, screen real estate or any number of limitations that a mobile device imposes. The code and markup behind a webpage might also be tailored to serve certain information only appropriate to users of a certain browser, potentially even limiting which browser may view

\(^4\)An example for a user with a unique identifier of 12345 would embed this data with a result like \texttt{http://12345@socialmediawebsite.com/path/to/further/resources.php?p1=foo}. 
a website, as was common in the browser wars [53]. This behavior is still found on systems where reliability of experience is crucial and the website administrators have taken the route of excluding rather than being accessible. For these and other reasons, it is easy to imagine two webpages (even those on the surface web behind no authentication) being rendered differently when viewed by two people or by a single person on multiple devices. As personal archives are much more susceptible to this tailoring because of the desire of social media websites’ users for ubiquitous access to a service, it is important to address these deviations in user experience and how they relate to personal web archiving.

FIG. 4: When accessing facebook.com from a mobile device (left), the content supplied to the user is tailored to the user’s available screen width. Where the screen width is less predictable but often wider, as is the case with a PC running Internet Explorer (right), the user is supplied content with much more detail.

Information about a user’s perspective when visiting a website is identified by a user agent string, representative of identifying information (e.g. choice of browser, current platform) and a few other pieces of information, essentially a digital fingerprint [13], to allow the sniffing of a user’s browsing attributes [17]. As was more common in the past, when a user was prevented from accessing a website because of one of these attributes (e.g. blocking any users that are not on a Macintosh from a Macintosh Fan Club website), this user agent information could be spoofed (or falsified) to circumvent the restriction. Spoofing, as in Figure 5, also has the constructive use in testing to assure no restrictions of this sort are being accidentally imposed by
the webmaster. Even with spoofing, how the page appears from the spoofed perspective cannot be accurately observed without the further assistance of a corresponding rendering engine. An example of constructively using browser spoofing would be in testing code or markup that is tailored to a specific version of Microsoft’s web browser, Internet Explorer (IE). Since version 5 of IE has allowed unobtrusive HTML comments of a certain form to be rendered only by those that satisfy the condition. As an example:

```
<!--[if lt IE 5]>
Your browser is too old and cannot render this content.
<![endif]-->  
<!--[if gte IE 9]>
...features not supported by version of IE prior to 9...
<![endif]-->  
```

This content, enclosed in an HTML comment tag from the perspective of any browser but IE, will render differently between users that are using IE version 4 and version 9 and will show no content generated from within the comment for users of browsers other than IE. In this instance, IE versions less than version 9 are unable to natively render the content within the second conditional, and so are appropriately not shown this content so as to not confuse users with non-functional content. This filtering is performed client-side and could be overcome with user agent spoofing. In addition to content exclusion, a user may be served what the webmaster believes is a more appropriate display of the content, potentially leveraging features that are only available and are appropriate on a certain platform. An example of this would be forwarding a user to a mobile version of a website that utilizes the GPS functionality in a smart phone that would not be appropriate on a user accessing the same website from a PC without this functionality. Content might also be dynamically generated based a user’s choice of browser, potentially utilizing a reliable browser detection library (e.g. QuirksMode’s BrowserDetect) to serve only appropriate media that the webmaster believes will be compatible and optimally experienced by the user. The webmaster could also potentially exclude access to content regardless of user agent spoofing, as the above library does not rely on the value being spoofed to reliably detect the user’s browser.

6http://www.quirksmode.org/js/detect.html
FIG. 5: Websites like web-sniffer.net allow a user to spoof their user-agent to determine if different results are produced when various browsers are visited. Browser-based plugin approaches also exist but by using web-sniffer, a user is able to see the method used (modification of HTTP headers) to accomplish the spoofing. Note the spoofing of the Opera web browser while Google Chrome is being used.

If a user wished to archive two versions of a website from two different perspectives (e.g. browser, platform) and view them from either in the future, the content might not be displayed properly yet that might not matter. The device from which a user views his archive should not limit whether the archive is viewable, but an attempt should be made to display the archive that contains the most similarity to the user’s current perspective. Retaining this information to be included with the archive should be as simple as capturing this metadata at the time of archiving, but the Wayback Machine does not natively support user agent switching.

As we saw with Heritrix’s stripping of portions of the URI it considered superfluous to archiving the page, encoding user agent information into the URI would be a dangerous move because of the implicit coupling of Heritrix and Wayback. Instead, a more reliable way to preserve this information would be to encode it as metadata.
in the WARC records. Metadata is information that enables and documents the long-term preservation and access to digital objects [51]. Retaining this information at the creation stage of the digital objects is preferential for good practice, as described in Section 2.1.1. Because the implementation-agnostic case (i.e. appealing to the WARC format independent of wayback and Heritrix) is useful to explore for further developing the applications that utilize the WARC format, encoding user agent information in the URI will be discussed further here. Doing so will reinforce the lower degree of qualitative optimality of considering an encoding-based scheme over one that appeals more to the format itself.

With encoding the user agent into the URI, we wish to retain semantic, simple URI schemes of the resulting archive and generalize the scheme to be applicable to a variety of social media websites. Encoding too much information within the URI scheme might be counterproductive to these goals. To consider a possible encoding scheme while leveraging the perspective specification described above and ignoring the implementation-specific filtering by Heritrix, this information can be encoded in what would normally be used as the password field in a URI. With the Facebook example: \texttt{http://perspective:useragentinfo...@facebook.com}, it is unclear how to encode this information to be comprehensive of all of the information in the user-agent string and still be succinct. Extracting all of the content and appending them in a fashion akin to appending variable values to the end of a URI string (e.g. \texttt{http://www.example.com/index.php?user=john&color=blue}) is not conducive to our goals, as even a scheme with limited browser attributes would require URIs like “\texttt{http://myusername:engine=Mozilla&platform=AppleWebKit&platformversion=7B405&...”}.

Unlike overloading the username portion of the URI scheme, this sort of information loading is an abuse of the original intention of the password attribute and would not scale well when integrating the archives with other systems that use the field for its intended purpose. Because of these issues, it would be difficult to represent all possible permutations of perspective without a reference to external encoding and that would still then contain a combinatorial number of variants.

Two non-mutually exclusive alternatives to this scheme would be to either include the user agent information as WARC metadata (suggested above) and/or inject the information into the target page for later reference. The first case is optimal, as it is the more semantic option and the WARC format is designed to be extended in this
way. Unfortunately, from the client side perspective and the current offering of the Wayback Machine and the open source wayback\textsuperscript{7} package, this user agent information is not accessible to the end-user, which makes storing it in a WARC metadata record useless without a custom Wayback build. An additional offering from the Internet Archive’s Archive-It\textsuperscript{8} website has implemented a way to attach additional metadata to an archive by using an external database but because of the overhead, this would be impractical for a casual user.

The second alternative is more obtrusive on the target document and less semantic than the aforementioned ideal approach but is very accessible to the end-user, who is the target of this study. At time of archiving, the user agent can be collected and injected into the target HTML page as HTML metadata, rather than WARC metadata. This information is directly accessible on the client-side via Javascript. Because of this obtrusion, maintaining archive integrity is quite important. Because this injection technique is the most accessible of the techniques discussed, considering it as a use case for maintaining archive integrity will also be discussed further in Section 2.2.7.

*Encoding the User Agent as WARC Metadata Does Wayback support this? If not, can this be implemented in the add-on in a fashion similar to Archive-It’s “category” attribute?*

### 2.2.3 Archiving Versus Backing Up

As documented in Section 2.1.1, users are often confused as to what constitutes an archive over a backup, or they perform their backups using sub-par methods that make recall of resources difficult and assurance of the resources’ existence difficult to verify. While institutional archiving efforts are making great strides forward, consumers are unintentionally flirting with digital brinkmanship in regards to method of archiving [28]. It is not unusual for consumers to write the most valuable of their files to external media [28], which is prone to decay and is unwieldy when the media turns into a stack of media with little-to-no metadata and immense overhead for one to recall a desired resource. Some even move files from one machine to another, presuming the data is stored and safe in a folder with a label like, “My Old Documents” yet moving files from one PC to its successor is not actually creating an

\textsuperscript{7}\url{https://github.com/internetarchive/wayback}
\textsuperscript{8}\url{http://archive-it.org}
archive [28].

Many are moving away from physical storage and relying on social media, free unlimited storage e-mail space and other online services (particularly in social media) to be a redundant means of backup, again finding appeal in LOCKSS. Individuals use these services as safety net for rescuing their digital belongings [28] where they once referred to a stack of discs. These services often format the information in a way fitting to the service (a 300 pixels per inch (ppi) image might be scaled down to 72ppi to reduce file size), which leads to a new problem of deciphering, “which copy is the best” and, “what are my options”. The difference here should be clear. While the LOCKSS ethos assures that some backup is retained (assuming users occasionally assure that the content still exists, as previously discussed), without metadata, these copies still exhibit the same problems as backups, which archiving would prevent in attributing metadata. A further requirement of a backup being an archive is data portability. Utilizing the WARC format is a means to assure this.

2.2.4 Privacy & Security

Some people feel that everything on the web is in the public domain [32] though the means one uses to obtaining this data is controversial [41]. With the approach of obstructing this public access to information on the web via the walled garden of authentication, social media websites are attempting to assure that only the information that a user of the service wants exposed will be exposed. With this assumption, users rely on the service to protect the data, but this protection scheme is the root cause of the difficulty in archiving the information when the user wants to liberate it for the purposes of service-independent archiving.

Google, among others, have made strides in assuring that a user can liberate or wipe the data contributed to the service, but this sort of access-through-liberation breaks the WYSIWYG expectation of a user when replaying an archive and further, is only a backup of the data rather than an archive. The data that is able to be liberated from services that provide such a feature is often provided in a sanitized format and is often incomplete of facets on the website that would make replaying the page representative of the archive more genuine.

Retaining privacy and security for this data and still making it quick and easy

\footnote{Facebook provides a means of acquiring one’s data at \url{http://www.facebook.com/setting} but it is first sanitized by the service, so is not WYSIWYG.}
to retrieve and replay requires a scheme of protection. Various approaches can be considered with two dimensions: degrees of encryption and centralization. The sweet spot for these two is discussed here.

Transparency of implementation is debatable, as often the level of protection a security scheme provides is inversely proportional to the amount of information that is known about the scheme being used. Ideally, advances in encryption could be applied here but many of these schemes are expensive, require a central server and are impractical from the context of a browser. As we hope to overcome the barrier of authentication by leveraging the browser, we will consider approaches toward security and privacy that can take advantage of the context of the web browser by the user and emphasize decentralization. By emphasizing decentralization, archives will retain a greater degree of portability and the process of implementing the specification will remain more accessible without the undue hindrance of an external service. The goal is to explore the optimal degree of security while still making it easy to obtain and tailor the level of security that a user desires.

The initial approach explored is that of symmetric-key-based cryptography. Through providing this simple means of encryption, a user will be able to assure some (albeit small) degree of protection is used. We can leverage this scheme even further by providing a relevant key to the data prior to encryption with a user-specified key. This key would consist of a unique identifier representative of the user on the network, e.g. a user ID. Other secure alternatives [43] are currently being developed to accomplish the goals of generating asymmetric key pairs and built-in, i.e. not implemented by an external library. The merits and pitfalls of using this symmetric key based approach this approach are discussed below.

The difference between encoding and encryption should be clarified, as both are utilized by the tools built and manipulated in this thesis. Encoding consists of transforming data with the intention of usability and is utilized so that it can be consumed or transporting to a target system. An example of this is in image data, which is frequently transferred as an ASCII string representing the encoded form (usually base64) of the data needed for the image to be reconstructed. Encryption, however, has the intention of keeping data secret while utilizing encoding and some other security measure.

For the use cases (Section 3.1) described in this paper, a symmetric key-based approach is used. This simple approach usually relies on a shared key by multiple
parties. This key is used for the initial encryption and the eventual decryption of
the data. In the case of personal archives, both parties are frequently the same user.
Transmitting this key is frequently the downfall of symmetric key approaches, as a
man-in-the-middle attack can be used to intercept the data and brute force the key
to expose the data. The approach performed by WARCreate never transmits this
key but only the data, so does not suffer from this issue. Data is encrypted with
the key prior to transmission. When the encrypted data is to be retrieved, it is
largely gibberish if the user does not know the key. The key is entered by the user
on the client side once the gibberish is received (or supplied to the tool beforehand
and retained) and used as the symmetric key for decryption, just as if the data were
being sent to another user. The implementation details of accomplishing symmetric
key based encryption are described in Section 2.2.5.

2.2.5 Overhead Analysis and What is Lost By Using Encryption

With conventional key-based systems, the intention is to supply the key (e.g. a
user’s password) once, process that data and make it difficult to reverse the processing
and obtain the original data. Verification that this data is correct is a matter of taking
new input, running the same processing procedure and verifying that the results
match. Because we hope to retrieve the encrypted data, this one-way encryption
scheme is unsuitable.

The more suitable approach is to use a symmetric key. Using symmetric keys
allows the content, represented in a WARC file as ciphertext, to be decoded only
if the key is known by the user. This key is best implemented by way of using a
standard, consistent hashing function on the concatenation of the key and the data.
An extension to this would be to include the hashed key at a location in the ciphertext
based on the length of the original key. A thorough analysis, which is beyond the
scope of this research, would have to be done to assure that this method is reasonably
secure against rainbow table attacks\textsuperscript{10} at deciphering the content.

No browser-based decentralized approach will be as strong as a server-based so-
lution. A solely client-side approach would use Javascript. Javascript suffers from
issues of runtime malleability, shortcomings in system primitives needed for true

\textsuperscript{10}Simpler encryption schemes, like those using symmetric keys, are especially susceptible to rainbow table attacks. The gist of the attack employs using pre-computed hashes to match up with encrypted keys.
cryptography and a variety of problems [36] that make it less than ideal for cryptographic implementations. However, adopting a server-based solution too tightly couples a potential personal web archiving tool to a single point of failure. The overhead required to encrypt the data is $O(n)$ if a simple combination of base64 encoding and RC4 encryption is used. Such implementations are natively available in Javascript\textsuperscript{11} and sufficiently secure for further discussion on less implementation specific issues.

### 2.2.6 Further Discussion on Centralized Versus Decentralized Approaches

Aside from the ease of enforcing privacy and security, taking either a partially/fully centralized or a completely decentralized approach at personal web archiving has pros and cons for each. The ability to utilize external resources weighs in favor of having some element of centralization or some form of external server access to accomplish the process. This comes at the expense of loss in privacy and increases the potential for breakage as tools on external systems need to communicate. On the other side of the spectrum, a completely decentralized approach is too extreme to be applicable to web archiving, where the resources are almost always located on a remote machine. A server-based approach could use protocols like OAuth\textsuperscript{12} or OpenID\textsuperscript{13} to assure that only the user that created the archive can subsequently access it. If these services were to get compromised, go down or suffer from data loss of authentication credentials, personal archives created might become inaccessible. Relying on external services for the core function of accessing an archive increases the potential for these problematic scenarios. The specification proposed in Chapter 3 is a guide for the tools. If it were to become outdated, removed from the web or otherwise become inaccessible, the cached version of the specification would remain suitable for as long as the previously existing specification were applicable (e.g. until it would normally be updated). It is recommended that tools based on the specification cache the respective (to the target service) implementation of the specification for an incident such as the one described. Unlike a tool relying on the external service of the specification, the issues with tools relying on authentication services (if implemented correctly) would not be overcome with caching due to the

\textsuperscript{11}http://code.google.com/p/crypto-js/
\textsuperscript{12}http://oauth.net/
\textsuperscript{13}http://openid.net/
inherent reliance on such authentication systems on maintaining the allowance of external access.

2.2.7 Archive Integrity

Preservation and archival of the digital born media is not trivial and can contain data quality issues [49]. Two considerations should be addressed in regard to archive integrity, one abstract and one implementation-specific. When data is collected by a crawler, it is normally not transformed in any way, as a conventional web crawler like Googlebot only archives metadata and sometimes a cached copy that is hardly sufficient to be considered an archive (Section 2.2.3). From an end-user perspective, Heritrix appears to tailor crawled and archived pages to be replayed in the Wayback Machine by appending additional archive metadata and graphical user elements as in Figure 6. However, as documented in Wayback’s administrator manual [4], the content originally archived can be viewed by the end-user by adjusting the parameters queried to the Wayback Machine. This behavior also exists in the open source wayback **TODO: verify**. Often because archived content no longer exists in its archived form, there is not a way to verify that content archived at a particular time, even by Heritrix, is identical to the representation in a WARC file. Assuming tools created based on the standard are not as unverifiably diligent at preserving the content’s original form, a list of modifications that have been performed on the content’s original form (a digital paper trail [48]) should be presented before storing the data in a WARC file.
FIG. 6: Upon replay, it would appear (left) that the archive has been decorated with user interface elements by the Internet Archive to allow users to navigate from between temporally different versions of the same archived page. The source code (right) seems to confirm this with the addition of various scripting that compromises the integrity of the archive so that a user cannot be sure they are experiencing the content in its original form.

Providing a means to represent how content has been modified only increases the likelihood and at worst does not affect the likelihood (i.e. it is not hindered) that the content archived is consistent with its original form. If it is necessary to first transform content in any way prior to preservation, as is done in converting images to their binary representation, this should be documented in the WARC file and attributed to the archived content. How? An example use case where documenting the changes made to the webpages that were archived can be supposed in Archive Facebook. In the add-on, URIs on a webpage are transformed from their absolute references (e.g. http://www.facebook.com/resource.html) to a reference relative to other pages (e.g. ./resource.html) that are archived in the same session to assure that the “replay” of the backed up content is navigable by the end-user. Because Archive Facebook does not currently utilize the WARC format, the application of documenting changes for the sake of archive integrity is largely moot. If Archive Facebook were to utilize the WARC format, rewriting URIs to allow navigation between pages archives would be unnecessary, as wayback prepends all URIs with the hostname on which the wayback instance resides, making references to pages that should be accessible from one page to another a process handled by the replay system, usually wayback itself.
As relatively few tools implement the WARC standard at this time, little has been done in taking into account these considerations. The reference implementation of WARCreate [21], built for this research (discussed more in section 2.3), provides a way for a user to manipulate an archive while still inherently retaining the original content through a rollback sequence that is common in other practices (namely, version control) that use differentials.

This section requires further research and might be beyond the scope of this paper though provides an additional theoretical aspect and potential extension in formalization beyond the thesis

- Address header fabrication, techniques used to transpose data into a form to be read locally and fit for WARC conversion
- Potentially write section on rationale of various HTTP and WARC header attributes and values
- Propose way to store differentials about how the archived content was modified so that the original can be restored.

2.3 PRIOR ATTEMPTS AT CAPTURING CONTENT BEHIND THE WALLED GARDEN OF AUTHENTICATION

The idea of circumventing service-provided APIs and leveraging the user’s context of the web browser is not a new idea. Tools like the aforementioned Archive Facebook and WARCreate, a Google Chrome extension used to create a standard format of digital archive from any arbitrary webpage, have succeeded in providing a means for a user to retain this content but capture the content in an ad hoc fashion. Though the WARC file format is widely used, it is just a specification for a container and says nothing about the formats or semantics of the objects contained within WARC files nor about their relationships to each other [51]. While other formats exist (e.g. FOXML\textsuperscript{14}, METS\textsuperscript{15}, MPEG-21\textsuperscript{16}, etc) that attempt to accomplish web resource bundling, none have been adopted by the Open Archival Information System Reference Model (OAIS)\textsuperscript{17}. The objective of the OAIS model is to provide a

\textsuperscript{14}\url{http://www.fedora-commons.org/download/2.0/userdocs/digitalobjects/introFOXML.html}
\textsuperscript{15}\url{http://www.loc.gov/standards/mets/}
\textsuperscript{16}\url{http://mpeg.chiariglione.org/standards/mpeg-21/mpeg-21.htm}
\textsuperscript{17}\url{http://public.ccsds.org/publications/archive/650x0b1.pdf}
framework for the use of these bundling specifications [51]. Because of the WARC format’s adoption of the model and its utilization by the most prominent web archiving organization (Internet Archive), further discussion assumes this to be the optimal format on which to base future web archiving efforts.

The model used by WARCreate to accomplish its objective can be applied to other tools. By leveraging the user’s perspective of the web browser (Figure 2.3, marker 1), a user interacts with a tool (marker 2) that serves as a bridge for converting viewed content into an archived form. In the case of WARCreate, the process described in Figure 14 is executed (marker 3) and outputs the file representing the archived content (marker 4) into a local repository. The concrete example of WARCreate outputting WARCs couples with the consumption of this archived format (marker 5) by a system created to read the format, a local instance of wayback in respective to WARCreate. The user can then access this local instance (marker 6) and view the result of the processing instigated by the initial interaction (marker 2) of the tool (e.g. WARCreate) via the browser.

FIG. 7: A higher level view of an archival tool built upon the browser platform gives perspective on how all of the components of archive creation, consumption and replay can be experienced by the user. Displayed here is the process that WARCreate uses though any tool that uses the browser medium for perspective could reuse this model.

Archive Facebook, originally developed by the Web Sciences and Digital Libraries (WS-DL) Research Group at Old Dominion University operates by appealing directly to the breakup of content sections on Facebook, a social media website. Though archivists have made previous strides in preserving websites like YouTube [8] and MySpace [11], a growing amount of personal (and what will be historically significant) information is locked behind the walled garden of Facebook [39]. Through a system of scraping, the add-on is able to capture the content in the viewport of the user’s browser to disk and resolve references to all downloaded resources. The final steps
of the tool’s operation link all of the sections together and provide a starting point for a user to replay the archive via an entry in the sidebar of the user’s browser. The archived content suffers from being merely a backup and not an archive (discussed in section 2.2.3) yet excels in that it is truly WYSIYWG and navigable as if a user was viewing the live webpage. As the archive creation process is prone to breakage when Facebook redesigns its layout, the embedded nature of the scraping parameters is insufficient to make the tool adaptive.

The intention of the creation of the WARCreate Google Chrome extension was to sacrifice direct WYSIWYG viewability at the advantage of appealing to a standardized archiving format. The WARC format [19] was developed by the Internet Archive (IA) as a medium for preserving web archives. WARC files are normally generated by IA’s web crawler, Heritrix, to be consumed by the WARC processing and replay engine Wayback Machine. Wayback Machine is open source in the form of the wayback[18] project. With the assumption that a user would be able to leverage the open source nature of a tool (wayback) that was created to consume a format of digital archive that was developed to be robust and extensible (i.e. the WARC format), WARCreate generates this format of archive from any arbitrary web page and works toward bridging the gap that currently exists between conventional web archiving and web archiving of content behind the walled garden. WARCreate suffers from a problem beyond its original scope. A user is unlikely to have a private instance of wayback running for personal use, so the output, though standard, is of little use without the means for replay.

Though both of these tools circumvent the issue of context and authentication, they do so in different ad hoc fashions. Archive Facebook does not have relevance to websites outside of Facebook, and WARCreate is merely a general purpose bridge of getting content from one form to another. Both of these tools only work in the browser for which they were respectively created. Through my work on both of these projects and the lack of extensibility that comes from these types of one-off tools, I am providing a standard for addressing the hierarchy and means for capturing content on social websites residing behind authentication so that this content can be captured and treated in the same way and with similar tools as conventional web archives.

[18]https://github.com/internetarchive/wayback