Center for Research Libraries
Auditing and Certification of Digital Archives Project

# LOCKSS Audit Report

November 2007

Report prepared by Robin Dale, with contributions by Bernard Reilly and Marie Waltz.

*Note: This report was produced as part of a test of the RLG/NARA Draft Audit Checklist for the Certification of Trustworthy Digital Repositories and other metrics developed by the Center for Research Libraries under a grant from the Andrew W. Mellon Foundation. Because the metrics and methodologies applied were still in development at the time of the audit, this report should not be considered a definitive assessment of the repository described.*

# TABLE OF CONTENTS

## 1 Summary Statement

The audit of the Lots of Copies Keeps Stuff Safe (LOCKSS) open source software system and the accompanying LOCKSS program based at Stanford University, took place over the course of two months, between September and October 2006, with a brief review and update in January 2007. Further clarification of some points was solicited and received from LOCKSS in November 2007. The goal of the audit was to form an overall risk analysis of the LOCKSS' software and system's ability to provide long-term access to scholarly electronic journals and other digital resources. LOCKSS was evaluated on three aspects of the program's operation:

a. characteristics of the archiving software that affect performance, accountability, and business continuity of LOCKSS collections;
b. technologies and technical infrastructure employed by the archiving software; and
c. processes and procedures enabled by the software and implemented at local LOCKSS participating institutions.

This "sidewalk audit" differed from the other audits conducted by CRL as part of the same project. Unlike the 2006 audits of the e-Depot at the Koninklijke Bibliotheek, Portico, and ICPSR, the LOCKSS audit did not involve an on-site visit by CRL auditors. This was in part because LOCKSS is a distributed system, rather than a central repository. In addition it was not possible within the time frame of the CRL project to arrange a time mutually acceptable to LOCKSS and CRL for the prescribed two-day, on-site session.

Hence this report is based primarily upon publicly available documentation, information from publications and presentations about LOCKSS by a member of the LOCKSS team, information and documentation obtained from libraries utilizing the LOCKSS software, and information provided by Victoria Reich, Director of the LOCKSS Program.

The audit did reveal a robust, low cost Open Source software system, which at the time of the audit was used to preserve primarily electronic journals in more than 100 institutions. As an e-journal archiving system, the software functions as intended and is widely used to preserve and provide persistent access to e-journal content. This core functionality has been tested and validated through myriad minor, anticipated failures such as network access interruptions, data corruption, and individual LOCKSS box failure. In all instances of which the auditor was aware, the individual LOCKSS boxes performed as anticipated and content and/or access was automatically restored.

Moreover, the system is decentralized; allowing local custody and control of comprehensive copies of journal content by participating institutions and thus avoids a single point of failure. The system also continually monitors and validates archived content, providing high assurance of the continued presence and integrity of that content.

Concerns include the present "migration on access" strategy and the public perception that LOCKSS can be used as a digital preservation system for any and all types of Web-based digital content. The chief concern raised by the audit, however, was the sustainability of the LOCKSS

Program and LOCKSS Alliance.  Although the Alliance was formed to provide stability and financial support for the LOCKSS Program, at the time of our audit it had yet to reach subscription levels that would allow long-term sustainability.  Recent developments at LOCKSS, however, suggest that the service could soon become self-sustaining.   It was not possible on the basis of the information provided for the auditor to render an informed assessment of LOCKSS' prospects for economic sustainability.

In concept and in practice, the LOCKSS system has proved to be an effective approach to preserving Web-based content.  While not the only option enabling persistent access to Web-based collections, it is one of the few that allows institutions -- especially smaller ones—local capture and control of the resulting digital collections.

## 2 Executive Summary

In March 2005, the Andrew W. Mellon Foundation funded the Center for Research Libraries (CRL) Auditing & Certification of Digital Archives project, an endeavor to develop an audit and certification process for digital repositories and archives. Rigorous auditing and certification are necessary to determine the level of assurance that particular archiving arrangements provides to publishers/depositors and users, and to ensure that the valuable digital resources archives will continue to be available and functional over the long-term. As a part of the CRL project, Project Director Robin Dale performed a "sidewalk" audit and assessment of the Lots of Copies Keeps Stuff Safe (LOCKSS) open source software system as well as its organizing bodies, the LOCKSS Program and LOCKSS Alliance.

Among the goals of this audit was to evaluate LOCKSS to form an overall risk analysis as it relates to long-term access to scholarly electronic journals and other digital resources being managed by LOCKSS software. The audit of the various components of LOCKSS took place over the course of two months, from September – October 2006. A brief review and update took place in January 2007, and further clarification of some points was solicited and received from LOCKSS in November 2007.

This "sidewalk audit" differs in methodology and scope from the other test audits, and is not as rigorous. The assessment is based primarily upon publicly available documentation about LOCKSS, including information from the LOCKKS Website, publications or presentations about LOCKSS provided by a member of the LOCKSS team, and documentation produced by projects utilizing the LOCKSS software. The audit also involved extensive interviews with the director of LOCKSS, Victoria Reich.

Audit findings revealed a robust, but low cost Open Source software system used to preserve primarily electronic journals in more than 100 institutions. Moreover, the LOCKSS system is a decentralized one, allowing individual participating libraries to maintain local custody and control of a comprehensive set of copies of journal content. The LOCKSS software is also open source and thus can be maintained by participating institutions, thus avoiding a single point of failure. The system also continually monitors and validates archived content, providing high assurance of the continued presence and integrity of that content.

The LOCKSS system is also being used in a variety of projects to test its use and capability to preserve private networks of Web-based content (although not necessarily e-journal content). Further development and testing to support this new functionality were ongoing and taking place both in cooperation with LOCKSS staff, and within the broader open source community.

Designed as a system incorporating a risk-based, economical approach to persistent access to certain content types, LOCKSS appeared to function well. As an e-journal archiving system, the software functioned as intended and at the time of the audit was in use to preserve and provide persistent access to e-journal content. This core functionality has been tested and validated through a myriad of minor though anticipated failures such as network access interruptions, data corruption, and individual LOCKSS box failure.  In all instances of which the auditor is aware, the individual LOCKSS boxes performed as anticipated and content and/or access was automatically restored.  This is one of the only archiving systems to experience and recover from anticipated, real-time failures.

The audit found that LOCKSS tends to be most effective when applied to archiving e-journal content.   Uses of LOCKSS as a specialized preservation network of other Web-based content were underway at the time of the audit, but it would be premature to render judgment on LOCKSS functionality for that purpose.  As long as other uses of LOCKSS conform to its intended functionality (generally preserving static, Web-based materials that can be rendered by Web browsers), experiences similar to e-journal archiving with LOCKSS software should be anticipated.

Despite LOCKSS wide distribution and success to date, a few concerns emerged during the course of the audit.  These include LOCKSS' "migration on access" or "migration on demand" strategy.  This strategy has been demonstrated for images through a proof-of-concept test, but may prove much more difficult and costly for other formats.  However, the LOCKSS team has expressed their intention to consider incorporating functionality with emerging format registries that provide links to registered format converters, when such registries are available for production use, potentially saving time and the expense of other potential solutions.

The chief concern raised was the economic sustainability of the LOCKSS Program and LOCKSS Alliance.  Although the Alliance was formed to provide stability and financial support for the LOCKSS Program, at the time of the audit it had yet to reach annual revenue levels that will guarantee long-term sustainability.  LOCKSS Alliance membership (and corresponding dues) will have to increase well beyond their 2006 level to support the LOCKSS Program and its dedicated support and developmental staff.

Although the LOCKSS software is Open Source and the Open Source community has been encouraged to take on and/or contribute to ongoing LOCKSS development, a sizable, interested community had not yet materialized at the time of the audit.  LOCKSS recently reported, however, that in 2007 the Alliance generated sufficient income to "cover all the LOCKSS-related activities of the Stanford team." If such a development proves persistent it would suggest that LOCKSS has become self-sustaining, independent of grants and other soft money subsidies -- a very promising development.

In concept and in practice, the LOCKSS system has proved itself to be a solid approach to digital preservation of Web-based content.  While it is not the only option enabling persistent access of Web-based collections, it is one of the few that affords institutions -- especially smaller ones—onsite capture and control of the resulting digital collections.  Its low-cost, automated, digital preservation process should be considered a viable option for institutions wanting to collect and preserve materials appropriate to the LOCKSS software design.

# 3 Full Report

## 3.1 Introduction

As a part of the Center for Research Libraries Auditing & Certification of Digital Archives project, Project Director Robin Dale performed an audit and assessment of the LOCKSS (Lots of Copies Keeps Stuff Safe) archiving software, as well as the supporting LOCKSS program based at Stanford University. The Andrew W. Mellon Foundation-funded project is engaged in developing a complete audit and certification process for digital repositories and archives. Rigorous auditing and certification are necessary to determine the level of assurance that particular archiving arrangements provide to publishers/depositors and users, and to ensure that the valuable digital resources archives will continue to be available and functional over the long-term. As a component of the project, the LOCKSS archiving software capabilities and the supporting LOCKSS Program were assessed through a "sidewalk audit" – an audit which utilized public information about the software in addition to extensive interviews with LOCKSS Director, Victoria Reich, rather than a two-day onsite audit to complete the process. The results of that test audit are the subject of this report.

### 3.1.1 LOCKSS system & background

Unlike other test audit subjects in the project, LOCKSS is not a digital repository. Rather, LOCKSS is software written to function on standard desktop computers as a digital preservation system or digital preservation "appliance." The LOCKSS system software is managed and distributed by the LOCKSS Program based at Stanford University. Conceptualized in the late 1990s, the software was designed to address the then-emerging set of problems associated with libraries transitioning from purchased journals to leased e-journal content. At the time, it was difficult to obtain local copies of e-journal content, even for back-up purposes. Additionally, even if an institution was able to secure back-up copies of content as a part of the licensing agreement, there was little local capability to reliably load and manage the material in case access to publisher content was interrupted. To address these related problems, LOCKSS was conceived to enable librarians to have an "easy and inexpensive way to collect, store, preserve, and provide access to their own, local copy of authorized content they purchase."

The technology development and original system coding began in 1999 at Stanford. At its core, the LOCKSS software is the foundation of a seemingly simple, yet sophisticated peer-to-peer file system that stores and consistently verifies and updates Web caches. Each physical unit – referred to as a LOCKSS box -- runs on a fairly low-cost, mid-range computer in which the LOCKSS software is installed. The LOCKSS box then functions as a server to obtain content from appropriate Web sites (generally publisher Web sites which contain the content to which the institution subscribes), store the content, and interact with other LOCKSS boxes. This interaction among other LOCKSS boxes within the greater LOCKSS network is designed to enable a polling, verification, and replication mechanism that is one of the keys to LOCKSS data preservation. In essence, LOCKSS boxes within a defined network poll each other and compare content. If content is found to be damaged or missing from a particular box, the software can obtain and/or repair the missing or damaged content. The key here is that multiple copies of content are stored widely in a distributed fashion and then systematically and repeatedly verified against each other. A fairly sophisticated polling system manages this process and based on many comparisons of the replicated content, can detect a copy or copies which are different from its peers and intervene in an appropriate way to restore authentic, valid content. Access to the stored content is not “granted” unless a specified “trigger event” occurs, generally the loss of access to the content through the normally specified access point (i.e., a publisher’s Website).

In the intervening years, LOCKSS technology has undergone testing and further development. The alpha test ran through 2000, and an early beta version was successfully deployed to 50 libraries worldwide between 2000 and 2002. Testing and development continued with partners from 2002-2004 and LOCKSS was released for production use in April 2004. Until 2004, the LOCKSS Program endorsed its use primarily for its intended purpose: enabling libraries to obtain and store local copies of content which libraries purchase, but which are normally accessed through publisher Websites. In 2004 however, several projects, including the MetaArchive project[1] and the ASERL Electronic Thesis and Dissertation Project[2], began to test LOCKSS as a tool for preserving content other than e-journals. Exploiting the principle of capturing content from the Web and utilizing Web caches as backups, these new projects use LOCKSS to capture other Web-based materials and preserving it in dedicated LOCKSS networks. Further projects, such as the Alabama Digital Preservation Network recently adopted such LOCKSS networks to create special “collections” of important Web-based content.

During this same time, the LOCKSS Program began further development to support its assertion of being a digital preservation system. By early 2005, LOCKSS programmers had designed and tested an initial implementation of format migration (migration on access) for Web content being managed by LOCKSS boxes.[3] This proof of concept migration strategy was a solid step towards proving that content managed by LOCKSS boxes would not only be available, but would continue to be renderable over time.

In parallel to system development LOCKSS Director, Victoria Reich, has worked with publishers to develop and obtain agreements allowing local system implementations of LOCKSS (“LOCKSS boxes”) to capture e-journal content directly from publishers’ Websites. These agreements will permit the development of further content-specific plug-in modules that drive the processes of collecting, preserving, and providing access to specific e-journals since each online publishing platform requires a separate module.[4] In 2006, Reich announced the formation of the CLOCKSS – Controlled LOCKSS – Initiative which is a direct partnership between publishers and libraries to implement a community maintained, dark archive, of participating publishers’ content.

### 3.1.2 LOCKSS Philosophy

The purpose of LOCKSS is to preserve scholarly literature published in electronic form and to ensure that these materials remain accessible to future generations of scholars, researchers, and

[1]MetaArchive <http://metaarchive.org/>
[2] Association of Southeastern Research Libraries (ASERL) <http://www.aserl.org/>
[3] David Rosenthal, et al. “Transparent Format Migration of Preserved Web Content.” *D-Lib Magazine* 11:1 (January 2005).
[4] *LOCKSS: A Distributed Digital Archiving System: Progress Report for the Mellon Electronic Journal Archiving Program*. Stanford, CA: Stanford University Libraries, 8 October 2002.

students.[5] It intends to provide “effective preservation” of the archived content and provides end user access to content in a limited set of situations (see Section 2.3.2.3, *Usability of Information*).

The LOCKSS Program is also motivated by a fervent belief in keeping content in the possession of the academic institutions rather than under the control of publishers or third-party archiving operations. LOCKSS was intentionally designed to enable libraries not only to obtain copies and archive materials, but also to have immediate access to them in the event of an authorized trigger event, large or small. This approach differs from other archiving and access models such as that of Portico or the Koninklijke Bibliotheek which work with publishers to identify and validate trigger events through a series of contracted communications, but could potentially leave library users temporarily without access to authorized content.

**Publishers**
The benefits of LOCKSS to publishers include reducing (or eliminating) the publisher’s internal archiving costs, meeting library demand for a trusted, third-party archive, and meeting library demand for perpetual access without negative impact on a publisher’s operations. At the same time, many publishers also participate in other e-journal preservation projects such as Portico, and the Koninklijke Bibliotheek’s e-Depot. Hence the LOCKSS value proposition of saving costs is undermined if publishers are required to participate in multiple initiatives. Moreover, by replicating a publisher’s content and placing copies in multiple locations, LOCKSS increases the risk to publishers of unauthorized access to or theft of that content.

**Libraries**
According to the LOCKSS Website, the benefits to libraries include securing protection against eventual loss of access to important scholarly source materials and providing a practical way to maintain continuity of library collections. The main difference articulated by the LOCKSS Program is that LOCKSS allows institutions to continue to collect and preserve – that is, maintain a usable copy of purchased content – locally rather than having that content managed and maintained by a separate repository. The latter example is true of the Koninklijke Bibliotheek and Portico models.

With the expansion of LOCKSS functionality to include persistent caches of other Web-based content, it is arguable that investing in a LOCKSS box or a collaborative LOCKSS project might enable long-term access to a wider array of Web or network based digital content beyond e-journals.

### 3.1.3 Organizational Structure (Brief Overview)

The LOCKSS Program was initiated by Stanford University and remains as an organizational unit housed within the university. Early funding for the program and software development came from a variety of grant sources, most notably the Andrew W. Mellon Foundation, the National Science Foundation, and National Digital Information Infrastructure and Preservation Program (specifically for CLOCKSS). LOCKSS has received support from a variety of other funders, as well as in-kind support from several organizations.

[5] Portico, an electronic journal archiving service <http://www.portico.org/index.html>

In recent years, there has been an effort to move away from grant-based funding and to make the program sustainable by creating the LOCKSS Alliance. The LOCKSS Alliance is a membership organization "open to libraries interested in LOCKSS as part of their strategy for building and preserving digital collections of e-journals and other Web-based content."[6] At the time of the audit the LOCKSS Alliance consisted of 125 members. It was governed by a Board of Directors, and staffed by project team members, including Program Director, Victoria Reich and Chief Scientist, David S.H. Rosenthal.

Membership fees support ongoing technical development of the software, as well as Alliance activities. Salaries of the staff and periods of intense development have typically been covered by temporary, grant funding. There is some uncertainty whether the current Alliance membership is large enough to sustain staffing costs and the degree of ongoing development needed for the software. (There is an extensive and impressive list of libraries using LOCKSS at http://www.lockss.org/lockss/Libraries, but not all of these institutions have opted to join the LOCKSS Alliance and pay the ongoing support fees.) Like other Open Source initiatives incubated within the library community (D-Space, FEDORA, etc) , there is some thought to finding a new organizational home, perhaps with a non-profit that would commit to covering the funding gaps, as necessary. Thus far, this has not occurred and it is unclear what the consequences will be if the Alliance dues do not completely cover all its anticipated costs. More information about this can be found in Section 2.3.1, *Organizational Analysis*.

#### 3.1.4 Technical Architecture (Brief Overview)

The LOCKSS technology is that of a network appliance on a peer-to-peer network of persistent Web caches. Its technical architecture can be broken into two parts: the software and the hardware.

**Software:**
The first version of LOCKSS software was based on a boot-floppy distribution of Linux. After three years of testing at over 50 libraries world-wide, this appliance level of the system was replaced by a second version, based on a modified version of the OpenBSD install CD-ROM. [7] It is a specially configured version of OpenBSD which boots and runs from a CD, downloading updates automatically. It relies upon daemons to both get content to LOCKSS boxes, as well as to cooperate to detect and repair damage across the LOCKSS network to which it belongs.

The software collects HTTP delivered content from Web sites based upon plug-ins designed for the content and Website from which it will be harvested. When working with e-journal publisher sites, LOCKSS must have specially authorized access to enable it to crawl and harvest content appropriately. Once a publisher's content is collected, the software daemons mentioned above consistently audit the integrity of the files and cooperate with (poll) other LOCKSS boxes on the network to validate caches against one another and repair problems and resolve gaps, as

[6] The LOCKSS Alliance <http://www.lockss.org/lockss/LOCKSS_Alliance>
[7] David S.H. Rosenthal. "A Digital Preservation Network Appliance Based on OpenBSD." *Proceedings of BSDCon*. San Mateo, CA September 8-12, 2003.
<http://www.usenix.org/publications/library/proceedings/bsdcon03/tech/full_papers/rosenthal/rosenthal.pdf>

necessary. (The more organizations on the particular LOCKSS network, the higher the level of assurance in the polling/authentication process among LOCKSS boxes.) The software is coded to make each of these processes occur on a regular basis without human intervention.

**Hardware:**
Some basic design principles were factored into the creation of LOCKSS, including reliance on low cost PCs combined with widespread replication. In doing this, the start-up cost to utilize LOCKSS is low and the potential failure rate of individual inexpensive PCs is offset by the content being replicated on many similar devices. Malfunctioning PCs simply need to be replaced with a newer PC, have the LOCKSS software reloaded and configured, and the programs will accomplish the rest in regaining and loading the appropriate content back onto the local LOCKSS box. The configuration and machines are deliberately kept simple.

The current minimum hardware requirements for LOCKSS boxes are as follows:

- *A specified amount of CPU and memory.* 1GHz VIA CPUs are the minimum recommended, while a 2.4GHz Celeron is "lavish." LOCKSS recommends 1GB of memory. There is a bug in the current software that causes it to fail on machines with more than 2GB of memory; it was to be fixed in the subsequent release.
- *A CD drive and optionally either:*
    - a floppy disk drive
    - or a USB flash memory drive with a hardware write-protect switch.
- *Specified disk capacity.* 250GB is enough to get started. The current CD supports both parallel ATA (PATA) drives and serial ATA (SATA) in native mode. Some adjustment of BIOS settings may be needed to handle SATA drives.[8]

Further information about LOCKSS technical architecture can be found in Section 2.3.3, *Technical Analysis*.

[8] Installing LOCKSS – Computer Specifications <http://www.lockss.org/lockss/Installing_LOCKSS>

## 3.2 Objectives, Scope, & Methodology

### 3.2.1 Scope

This audit evaluated and provides information on the following topics:

- Organizational Infrastructure
- Technical Analysis (Digital Object Management, Technologies and Technical Infrastructure)
- Content
- Vulnerabilities

In all areas, the focus was on identifying and describing issues that could affect the viability and stability of the repository and the digital objects stored within the system.

At the time of the audit, LOCKSS had been released in its production system for more than two years. The software was stable and only minor development issues were underway, most of which were to address the plug-ins required by new publisher participation.

### 3.2.2 Method of Work

The work performed in this audit was different than the full-scale audits performed at other organizations. It consisted almost entirely of a review of publicly available documentation, in addition to several conversations with LOCKSS Project Director Victoria Reich. Based upon available information, the RLG-NARA Checklist for the Certification of Digital Repositories was completed to the best of the auditor's abilities and is reflected in this report. An onsite visit, however, did not take place. And no detailed technical testing was conducted although several scenarios were communicated to V. Reich and D. Rosenthal and responses were received via email. These scenarios were designed to detect potential vulnerabilities in policies and functionality (ingest, processing, archival package creation, data loss detection & resolution, access, etc). Such scenario testing cannot ascertain the integrity of the digital objects stored within the LOCKSS system or any particular LOCKSS box, but did provide insight into threat detection and risk management capabilities of the system. Finally, a brief analysis of content was made to determine the extent of content available, as well as discovery and delivery options that would be accessible to users should a trigger event occur and "unlock" any LOCKSS content.

### 3.2.3 Standards against Which the Audit Was Completed

The RLG-NARA Checklist for the Certification of Digital Repositories (August 2005) provided the metrics for this audit. The checklist was developed by an international task force of experts in digital preservation, digital repositories, and data archives. While the Checklist is not yet an international standard itself, it is based upon and references a number of international standards and best practices such as the Reference Model for an Open Archival Information System (ISO 14721:2004), Control Objectives for Information and related Technologies (COBIT) 4.0, Information Technology—Security techniques—Code Of Practice For Information Security Management (BS ISO/IEC 17799:2005), PREMIS Preservation Metadata (2005), and Trusted Digital Repositories: Attributes and Responsibilities (2002).

## 3.3 Findings

As an acronym for Lots of Copies Keeps Stuff Safe, LOCKSS has come to be used in a variety of ways, having implications for understanding the organization, its activities, its sustainability, and its technical capabilities. The core concepts are briefly described here to enhance understanding of the full report details.

**LOCKSS Program**: the term used to describe the umbrella for the myriad activities developing and/or utilizing LOCKSS. Based at Stanford University, this is seen as the organizational home for LOCKSS.

**LOCKSS Software**: the software code driving the LOCKSS peer-to-peer, networked preservation system.

**LOCKSS Box**: a local implementation of LOCKSS software, run on a PC, and part of a larger LOCKSS network

**LOCKSS Team**: the group of people responsible for LOCKSS software development and LOCKSS deployment.

**LOCKSS Alliance**: the relatively new, membership organization established to make the LOCKSS program sustainable over time. Members pay annual fees to participate and the Alliance is governed by a Board.

**CLOCKSS**: Controlled LOCKSS, a new initiative of publishers and libraries to have a community-controlled, dark archive of e-journal content.

### 3.3.1 Organizational Analysis

#### 3.3.1.1 *Governance*

The LOCKSS Program was initiated by Stanford University and remains as an organizational unit housed within the university, rather than a separate corporate entity. Early funding for the program and software development came from a variety of grant sources, most notably the Andrew W. Mellon Foundation, the National Science Foundation, and National Digital Information Infrastructure and Preservation Program. Other grantors and organizations contributing in-kind support include:

- The UK's Joint Information Systems Committee
- Sun Microsystems
- HP Labs
- Intel Research Berkeley
- Stanford University Libraries and Academic Information Resources
- Stanford Computer Science Dept.
- Harvard Computer Science Dept.

Despite funding origination, the responsibility for managing LOCKSS remained with the LOCKSS Program and Stanford University. In recent years, a drive to move away from grant-based funding and establish a more sustainable program resulted in the creation of the LOCKSS Alliance, a new organizational body for LOCKSS management. The Program remains based at

Stanford and under the directorship of Victoria Reich, but is now under the control of a separate organization: the LOCKSS Alliance.

The LOCKSS Alliance is a membership organization "open to libraries interested in LOCKSS as part of their strategy for building and preserving digital collections of e-journals and other Web-based content."[9] It is governed by a Board of Directors, and staffed by project team members, including Program Director Victoria Reich and Chief Scientist David S.H. Rosenthal. Technical development is supported by a small team of engineers.

LOCKSS Alliance membership fees support ongoing technical development of the software, as well as Alliance activities. Salaries of the staff and periods of intense development have typically been covered by temporary, grant funding.

#### 3.3.1.2 ***Staff***

The LOCKSS staff is small, reflecting the highly decentralized nature of the LOCKSS program. In general, the team is primarily responsible for software development and has some technical support responsibilities to LOCKSS Alliance members. The day-to-day support required for LOCKSS boxes is the responsibility of the local LOCKSS implementers though by design, this responsibility requires little time commitment per month. LOCKSS is designed to be as automated as possible, so local tasks tend to be error alert resolution, basic machine checks, and occasional monitoring of software upgrades.

LOCKSS original engineering staff was highly qualified, due to the core group designing and implementing LOCKSS early on and a geographical location in the Silicon Valley. As time progressed, supporting these highly qualified staff with commensurate salaries posed a challenge for long-term sustainability. In many cases however, some original staff moved on to senior positions within local high technology companies but continue to advise and/or consult on LOCKSS issues as Friends of LOCKSS. The current engineering team is highly qualified and represents engineers as well as affiliated users of the system (including Chris Rusbridge in the UK).

At first glance, the staff appears to be small compared to the size of the Alliance, but LOCKSS's highly automated environment, as well as the fact that many responsibilities are managed at the local LOCKSS box level, actually make this a manageable set of tasks for the current team. Of course further development of the software could be advanced more quickly with a larger and dedicated team, but the LOCKSS Program does want the software to live in the Open environment, gathering contributed utility from institutions using LOCKSS. For those reasons, the current team size and skill sets they possess are suited to the program.

#### 3.3.1.3 ***Policies and Procedures***

The policy and procedures framework is appropriately lightweight. High-level policies are managed by the LOCKSS Alliance Board and the LOCKSS Technical Policy Committee. A set of "day-to-day" procedures to be managed by the local institution implementing LOCKSS are specified by the LOCKSS Program. Most of this information is available from the LOCKSS

[9] *About the LOCKSS Alliance*. <http://www.lockss.org/lockss/LOCKSS_Alliance>

Website (http://lockss.stanford.edu). This Website is actually a wiki, implemented in MediaWiki, so that changes made to the site are also tracked.[10] Policies related to collection, retention, etc., are the responsibility of the local institutions and the LOCKSS networks in which they participate.

There is a formal procedure for periodic review at the programmatic and Alliance level. Issues related to implementations at the local level are unknown to the auditors and are local responsibilities.

On the software level, there is a defined set of policies and procedures for code management. According to David Rosenthal, SourceForge is utilized as the sole source code management system and contains every change that has ever been made to the LOCKSS source, together with notes briefly justifying each change. These notes also correspond to entries in the LOCKSS Roundup bug tracking database, completing the cycle from change request through code source change.[11]

#### 3.3.1.4 ***Financial Analysis***

As a part of the sidewalk audit, only minimum financial information was available from the LOCKSS Program. Necessary information on LOCKSS finances was gathered from publications and conversations with LOCKSS staff and the following comprise the basic expenses: LOCKSS team salaries (almost exclusively supporting technical development and support) and activities associated with Alliance activities (travel, meetings, as well as meetings with publishers). Machine expenses for LOCKSS boxes are borne by the institutions supporting the local implementations of LOCKSS boxes.

**Business Model**

There appear to be two components of the current LOCKSS business model. The first is to achieve sustainability and allow continued centralized development through the formation of the LOCKSS Alliance. The goal is to move LOCKSS to self-sufficiency via membership fees. According to the LOCKSS Website, membership fees for the LOCKSS Alliance are based on Carnegie Classification for UA libraries and equivalent measures are used for non-US libraries. LOCKSS Alliance fees for 2006 were:

| *(Institution size)* | *(USD)* |
|---|---|
| Research universities (very high research activity) | 10,800 |
| Research universities (high research activity) | 9,600 |
| Doctoral/research universities | 8,200 |
| Master’s colleges and universities (larger programs) | 5,200 |
| Master’s colleges and universities (medium programs) | 4,443 |
| Master’s colleges and universities (smaller programs) | 3,685 |
| Baccalaureate colleges | 2,160 |
| Associate’s college | 1,080 |
| Specialized | (Quote available on request) |

[10] Email from David Rosenthal via Victoria Reich, 19 December 2006.
[11] Ibid.

At the time of the audit the current size of the Alliance was 125 institutions and by February 2006 membership dues had grown enough to cover “two thirds of the community support needed to be self-sustaining.”[12] Based on information available at the time of the audit, the Alliance had not yet reached its target membership level and was therefore still seeking further membership growth.

LOCKSS has since reported, however, that in 2007 the Alliance generated sufficient income to “cover all the LOCKSS-related activities of the Stanford team.” If true, this would indicate that LOCKSS has become self-sustaining, independent of grants and other soft money subsidies -- a very promising development.

It will be increasingly important for membership to continue to grow substantially, or for LOCKSS to find an organizational sponsor to provide additional funding to supplement the income from Alliance membership fees.

The second component of the business model is locating and securing other resources to assist with continuing technical development. According to Victoria Reich, “a key to the LOCKSS Program business model is building an open source technical community.” The JISC [Joint Information Systems Committee in the UK] was funding technical development staff through the Digital Curation Centre, and the Library of Congress NDIIPP program recently committed $700,000 in funding to the CLOCKSS dark repository program. (See Section 2.3.1.6 below.) According to Reich, “The LOCKSS Alliance community partakes in technical workshops held every month or two. The software is fully documented and available on SourceForge. We are encouraging communities to use the software for varied applications thereby building a robust user base and community of expertise.”

At the time of the audit it was not clear how the second component of the business model would translate into longevity for the LOCKSS Program, but it did address some of the issues of sustainability of the software should the LOCKSS Program exhaust its funding. Many users of the LOCKSS software are not members of the LOCKSS Alliance. While their institutions do not contribute to the financial well-being of LOCKSS, they do keep the software in use and so will potentially have an interest in its further development as an Open Source tool with its own longevity separate from the program.

Regardless of the status of the LOCKSS Program or Alliance, owners of LOCKSS boxes should be able to continue to independently keep the content gathered by LOCKSS boxes and even potentially gather new content. In these cases, technical support would need to come from the Open Source community that Reich asserts should support LOCKSS. Further development would also have to come from the community.

**Budget Direction**
The budget of the Alliance is controlled ad monitored by the LOCKSS Alliance Board of Directors.

[12] Victoria Reich. “Editors Interview with Victoria Reich, LOCKSS Program.” *RLG DigiNews*, 15 February 2006. <http://www.rlg.org/en/page.php?Page_ID=20894#article0>

#### 3.3.1.5 *Contracts (Submission Agreements) & Licenses*

LOCKSS was designed to work in a rights-compliant and rights-controlled environment. Specific technical components make sure that only authorized content can be collected and distributed among LOCKSS boxes in any LOCKSS network.

**Appropriate Contracts and Deposit Agreements**

Publishers participating in the LOCKSS Alliance must make their titles "LOCKSS compliant." LOCKSS compliance means that participating publishers must grant two specific types of permissions:

1. permission for libraries to collect, preserve, and provide access to content that they have individually licensed from the publisher; and
2. permission for LOCKSS software to crawl, collect, and preserve that content from the publisher Websites.

The LOCKSS Program (and Website) did not specify the rights language the publisher must use, but does provide suggested publisher declarations. By virtue of these grants of rights by the publisher, libraries may then:

- Cache and maintain through the LOCKSS system currently accessible materials that they have licensed from the participating publisher;
- Use such material in ways consistent with the original license terms;
- Provide copies to other appliances for purposes of audit and repair. [13]

One potential weakness of this arrangement is that publishers grant archive rights to the licensees individually, rather than to the LOCKSS Alliance. Hence technically the LOCKSS Alliance has no standing with the publishers through which it might effectively represent or enforce the rights of participating libraries should a publisher default on its agreement.

**Electronic Implementation of Agreements**

In order to collect content from a Web site, the LOCKSS software needs positive evidence that it has permission to do so. This is provided in the form of a permission page that contains one of the supported permissions statements. (Sample permission statements are available.)**[14]** This permissions component is handled through a Web-based LOCKSS publisher manifest (http://www.lockss.org/lockss/Publisher_Manifest_Page). A publisher manifest permits the software to crawl, collect, and preserve the content. It also lists the top level URLs so that the crawler knows where to start the collection process. A publisher manifest is needed for each Archival Unit (typically a volume) to be preserved through the LOCKSS system and LOCKSS provides both XML templates and a Web form to generate the necessary manifests.

In addition, the LOCKSS system requires the manifest page to include either a suitable Creative Commons license (http://www.creativecommons.org/), or the following declaration: "LOCKSS system has permission to collect, preserve, and serve this Archival Unit." Therefore, the right to preserve the content is *explicitly given* in all licensing arrangements made with publishers and is verified by publishers before content can be obtained.

[13] *For Publishers: Making your Titles LOCKSS Compliant.* <http://www.lockss.org/lockss/For_Publishers#Making_Your_Titles_LOCKSS_Compliant>

[14] *Supported Permissions Statements.* <http://documents.lockss.org/pub-wiki/SupportedPermissionStatements>

Finally, and significantly, agreements made by publishers come with perpetual access implications. Once content is harvested via LOCKSS (with publisher permissions) and resides within a LOCKSS network, the content is permanently available to authorized participants in the network. Even if a publisher withdraws its support for LOCKSS participation, it cannot remove the copies of content already ingested and under the management of LOCKSS participants.

#### 3.3.1.6 ***Succession Planning***

The LOCKSS Program and the LOCKSS Alliance are not repositories, but rather a programmatic effort and membership initiative operating under the auspices of Stanford University Libraries. Stanford provides assistance with publisher relations and software support. Formation of the Alliance was intended to keep the LOCKSS Program (support and development) sustainable. LOCKSS staff members actively seek other organizational support – potentially a new organization home which can commit to stable funding in addition to Alliance fees, although at the time of the audit a new organization had not been located and grants and Alliance membership revenues appeared to be the main sources of financial support.

Succession planning for the LOCKSS software is arguably just as critical as for the Alliance itself. Since the software was designed and released into the Open Source community, all users are technically capable of continued development and support of the software. Nonetheless, at the time of the audit a large Open Source LOCKSS community had yet to develop even though it has been a major part of the LOCKSS business model. If such a community does not develop and if the Alliance failed, it appears that no likely successor is in place to manage the central technical support and development roles currently managed by the LOCKSS team.

Regarding long-term content accessibility, by design the LOCKSS software places copies of the archived content at the authorized (subscriber) participating institutions. Should the LOCKSS Alliance fail for any reason, the content stored on local LOCKSS boxes would not be affected and the institutions would continue to have access to the content based upon the original publisher agreements.

NOTE: CLOCKSS is an implementation of a LOCKSS network designed as a dark archive. CLOCKSS partners – primarily publishers -- contribute financially to this effort. In June 2006, the Library of Congress entered into a three-year cooperative agreement with Stanford University to provide approximately $700,000 in support of Stanford’s CLOCKSS (Controlled Lots of Copies Keep Stuff Safe) pilot and related technical projects. This agreement was intended to provide three solid years of initial financial sustainability for CLOCKSS.

### 3.3.2 Content Analysis

#### 3.3.2.1 *Logical and Physical Content*

Content collected during a LOCKSS crawl is what is available on a participating publisher’s Website. The digital files collected are bit-preserved within the LOCKSS system along with their appropriate, though minimal metadata. The system preserves content in a repository defined by a set of Java classes. The Archival Information Package (AIP) consists of instances of these classes, representing the content itself, metadata obtained from the publisher manifest page and the HTTP headers, and an instance of a Java class implementing the LOCKSS plug-in API encapsulating metadata not obtained from these sources. This instance is normally driven by externalized metadata in the form of XML files.

Underlying LOCKSS design is the belief that preserving the presentation form of Web content rather than the source databases used to generate it showed promise of being a pragmatic, practical approach to long-term preservation. It leverages and preserves a form of the publisher's content made available directly by the publisher without any alterations or huge processes having to be followed by the publisher. It was also believed that making this content the target of preservation rather than any additional information or value-added options available from the publisher through a content management system or full source database would lead to agreements more acceptable to both librarians and publishers. (NOTE: there is a potential exception to this design in CLOCKSS. As a community dark archive, CLOCKSS will also accept publisher source files if they are offered, but does not require them.)

Content preserved on LOCKSS boxes also remains accessible at the original publisher's URL. Links and bookmarks, searches through indexing and abstracting databases, etc. resolve either to the publisher's site or to the locally cached content. So the physical and logical e-journal articles are the same on both systems, but the default is always to resolve to the publisher site unless the LOCKKS system detects a problem.

#### 3.3.2.2 *Extent of Content*

**Committed Content**
At the time of the audit more than 40 publishers allowed their content to be harvested and archived on LOCKSS boxes. Those forty publishers represented over 1,000 titles and an additional eleven publishers responsible for many other titles are in the process of agreeing to have their content preserved via LOCKSS. Some content is available to any LOCKSS box in the system, other content is available but its availability is restricted to LOCKSS Alliance members only.

Because of the design of the LOCKSS software and system, the content is ingested into the system as soon as the publisher signals agreement and has made the appropriate publisher manifest available to the software (including appropriate capture URL). At that point boxes on the LOCKSS network begin ingesting, auditing, comparing, and replicating the content along the greater LOCKSS network. Unlike other e-journal repositories like Portico and the Koninklijke Bibliotheek's e-Depot, libraries running LOCKSS boxes do not need to manipulate or dataload content. As soon as the appropriate conditions exist and the authorized URL is provided, content can be ingested automatically by the LOCKSS software. This prevents the delay of content additions common to other types of digital repositories and makes it unlikely that libraries will ever face a situation where committed content varies significantly from ingested content.

#### 3.3.2.3 ***Usability of the Content***

Content harvested and preserved on LOCKSS boxes is in the same form as the content exposed on the publisher Websites and is renderable with a Web browser. This generally means HTML-encoded content, but also includes the relevant PDFs, image and data files, as appropriate. Because of the requirement to be renderable with a Web browser (delivered as it was on the publisher site, excluding the value added content that might have been available through a publisher's surrounding content management system), the acquired contents becomes renderable with a widely-held, basic kind of software package – a Web browser. As long as a Web browser can continue to render the content as captured, it remains fully usable within the LOCKSS system, awaiting any possible trigger event.

There has been some preservation planning on the part of LOCKSS staff to make sure that what is captured and renderable in today's browsers will be able to be rendered and/or supported in later browsers. In the last year, LOCKSS technical staff completed a proof-of-concept formation migration capability with LOCKSS. The basis for this is a "migration on demand" or "migration on access" strategy which will essentially identify content problems via MIME-Type values (much like a format registry) and seek available converters to invoke and perform on-the-fly conversion of the digital object so that it can be rendered.[15] The LOCKSS software is currently being enhanced to allow an API to work across the system in this fashion. As format registries (e.g. GDFR) become available, the LOCKSS Program plan is to use them to locate, and to use the LOCKSS system to preserve copies of, suitable converters whose input formats match those found in LOCKSS boxes. These converters would then be invoked using the "migration on access" framework described in 2005 D-Lib paper to create temporary access copies on demand.

The LOCKSS team has agreed to consider incorporating functionality with emerging format registries that provide links to registered format converters, when such registries are available for production use, potentially saving time and the expense of other potential solutions.

**Content Accessibility**
In almost all instances, access to content occurs through the content owner's (publisher's) site exclusively. Copies of content preserved in LOCKSS box are not available to, or utilized by users, unless a trigger event occurs.

---

[15] David S.H. Rosenthal, et al. "Transparent Format Migration of Preserved Web Content." *D-Lib Magazine*, January 2005. <http://www.dlib.org/dlib/january05/01rosenthal.html>

**Trigger Events**
LOCKSS agreements with publishers, subscription (licensed) stipulate that content cached in LOCKSS boxes is to be made available to the local user community only in the event of a trigger event. Trigger events include:

- Publisher ceases operation or fails and content is no longer available from another source;
- Publisher no longer offers back issues to everyone and content is no longer available from another source;
- Journal ceases publication and back content is not longer available from another source;
- Expiration of the publisher's copyright;
- Catastrophic failure of publisher's traditional access mechanism; and
- Temporary failure of publisher's traditional access mechanism (or loss of access to it).

**Triggered Content**
Content preserved in LOCKSS boxes (and including that in the CLOCKSS archive) will be available after a trigger event. In the LOCKSS scenario, access to preserved content is triggered when the publisher (rights holder system) is unable to resolve a specific URL request from a user's system. When that URL is not available at a particular moment in time, LOCKSS content is "triggered" and the content on the LOCKSS box associated with that URL is served from the LOCKSS box to the reader. In the CLOCKSS scenario, unavailability of content would trigger what the LOCKSS Program refers to as "a collaborative process" between publishers, librarians, and representing societies "to determine whether materials should be made generally available to all for an unlimited or an indefinite amount of time."

**Availability (Timing) of Triggered Content**
As described above, traditional triggers within LOCKSS systems would result in immediate access to the content on the LOCKSS box. This allows users to have uninterrupted access to content, and quickly resolve instances of a when a publisher is temporarily unable to serve the content due to power failure or even greater system failure.

Content within the CLOCKSS archive is likely to take a longer period of time since the decision about releasing content widely must be addressed by all participating parties – publishers, libraries, societies, etc.—before the CLOCKSS content can be released.

**Perpetual Access**
As a component of LOCKSS agreements, subscribers are allowed access to appropriate licensed content and may harvest and store that content as a part of a LOCKSS network. If a publisher or subscriber ceases the contract, the harvested content remains the property of the previously subscribing institution, providing immediate and uninterrupted access to the content to which they had subscribed. This perpetual access via local access is an appealing component of LOCKSS. By nature of not only possessing a physical copy of the files, but being immediately able to allow user access, LOCKSS perpetual access option for e-journal collections is a powerful incentive for potential LOCKSS Alliance members.

#### 3.3.2.4 ***Link Management Solutions***

Link management and referential integrity are core components of LOCKSS functionality. As a part of LOCKSS content ingest, LOCKSS metadata is collected and maintained. This includes the URL from which the file was obtained. This serves as not only provenance metadata, but allows the link management to serve up content to the user when the publisher (content owner's) Website is unavailable.

#### 3.3.2.5 ***Termination of Service Policy***

This has been explained indirectly through the Perpetual Access and Licensing and Agreements sections. Termination of service means two things to a LOCKSS user. When a publisher terminates its LOCKSS agreement all content already harvested is still in the possession of the local LOCKSS box owner and may be served up under the original access terms of the institutional subscription agreement. If an institution decides to terminate LOCKSS Alliance membership, there are two implications. First, an institution will be responsible for the local support of the LOCKSS box and the broader LOCKSS network to which it belongs. Technical support from the LOCKSS team is only available to Alliance Members. The second implication is the potential loss of content. Certain agreements with publishers provide LOCKSS content permissions only to LOCKSS Alliance members. Terminating a LOCKSS Alliance membership could potentially result in the loss of access to future Alliance content.

#### 3.3.2.6 ***Threats to Content***

Content stored on LOCKSS boxes is fairly secure from threats to that content, especially if the LOCKSS network is fairly large. By nature of the LOCKSS automatic authentication and repair mechanism, content is consistently polled and verified against several other LOCKSS boxes. Detection of any anomaly results in further polling (if applicable) and repair. Failure of any one LOCKSS box is easily remedied by reconfiguring a replacement box and connecting it to the network. Content that is supposed to be on the previously damaged/malfunctioning LOCKSS box is replaced through network repair.

Moreover, the LOCKSS system is far more automated than most other repository or archiving options, reducing the chance of intentional or unintentional human error into the process.

It is unclear whether the migration on access policy will be a problem for all future LOCKSS content. The recent proof-of-concept project to enable migration on access was successful for the kinds of content tested. That said, with new uses of LOCKSS arising frequently, it is quite possible—in fact highly probable -- that this particular strategy may not be effective for all types of content. This would pose a risk to certain kinds of content and therefore this functionality should be the focus of future LOCKSS development.

### 3.3.3 Technical Analysis

#### 3.3.3.1 *Architecture*

The LOCKSS technology is that of a network appliance on a peer-to-peer network of persistent Web caches. Its technical architecture can be broken into two parts: the software and the hardware.

**Software**

The first version of LOCKSS software was based on a boot-floppy distribution of Linux. After three years of testing at over 50 libraries world-wide, this appliance level of the system was replaced by a second version, based on a modified version of the OpenBSD install CD-ROM. [16] It is a specially configured version of OpenBSD which boots and run from a CD, downloading updates automatically. It relies upon daemons to both get content to LOCKSS boxes, and to cooperate in detecting and repairing damage across the LOCKSS network to which it belongs.

The software collects HTTP delivered content from Web sites based upon plug-ins designed for the content and Website from which it will be harvested and saves the content as Web caches. When working with e-journal publisher sites, LOCKSS must have authorized access via an included publisher manifest (including pointers to appropriate content URLs) to enable it to crawl and harvest content appropriately.

Once content is collected, software daemons mentioned above consistently audit integrity of the files against other LOCKSS boxes on the network to validate caches against one another and repair problems or resolve gaps, as necessary. This integrity and authentication process is accomplished via a specially designed LOCKSS "opinion poll" that enables peer LOCKSS boxes to "vote" on large archival units (AUs) of content. According to a technical paper on LOCKSS, because each peer holds a different set of AUs, the protocol treats each AU independently. If a peer loses a poll on an AU, the protocol treats each sequence of increasingly specific partial polls within the AU to locate damage. (LOCKSS recently reported that during the first quarter of 2007, the system migrated to use the V3 protocol, which is capable of resolving all damage to an AU in a single poll.) Other peers cooperate with the damaged peer if they "remember" that it agreed with them in the past about the AU, by offering it a good copy, in the same way they would for local readers.[17] The software was coded to make each of these processes occur on a regular basis without human intervention.

The more organizations on the particular LOCKSS network, the higher the level of assurance in the polling/authentication process among LOCKSS boxes. As well, because peers can and will only help others they recognize from participating previously, this prevents free-loading. An institution's LOCKSS box must have downloaded appropriate content from publisher Websites and participated with other peers before [re]gaining content from other LOCKSS boxes.

**Hardware**

[16] Rosenthal, "A Digital Preservation Network Appliance Based on OpenBSD."
[17] Petros Maniatis, et al. The LOCKSS Peer-to-Peer Digital Preservation System. *ACM Transactions on Computer Systems,* February 2005. <http://www.eecs.harvard.edu/~mema/publications/TOCS2005.pdf>

Some basic design principles were factored into the creation of LOCKSS, including reliance on low cost PCs combined with high replication. In doing this, the initial costs of implementing LOCKSS are low and the potential failure rate of individual low-cost PCs is offset by the content being replicated on many similar devices. Malfunctioning PCs simply need to be replaced with a newer PC, have the LOCKSS software loaded and configured, and the programs will accomplish the rest in regaining and loading the appropriate content back onto the local LOCKSS box over a period of time. The configuration and machines are deliberately kept simple.

The current minimum requirements for LOCKSS boxes [from LOCKSS Website] are as follows:

- *A specified amount of CPU and memory.* 1GHz VIA CPUs are the minimum recommended, while a 2.4GHz Celeron is "lavish." LOCKSS recommends 1GB of memory. There is a bug in the current software that causes it to fail on machines with more than 2GB of memory; it was to be fixed in the subsequent release.
- *A CD drive and optionally either:*
    - a floppy disk drive
    - or a USB flash memory drive with a hardware write-protect switch.
- *Specified disk capacity.* 250GB is enough to get started. The current CD supports both parallel ATA (PATA) drives and serial ATA (SATA) in native mode. Some adjustment of BIOS settings may be needed to handle SATA drives.[18]

The hardware and networking access required to run LOCKSS are the responsibility of the local institution implementing LOCKSS. LOCKSS box caches need static, globally routable IP addresses and communicate via User Datagram Protocol (UDP), essentially in a broadcast mode. LOCKSS boxes can be run behind firewalls, if desired. (The V3 protocol now in use by LOCKSS uses TCP/IP, not UDP.)

The LOCKSS program does not provide hardware or network access, but instead provides only the software necessary to configure and drive the system. This frees the LOCKSS Program from having to budget these costs internally or charge for hardware and software. It also allows institutions to choose whatever PC system they prefer or that meets local requirements, as long as the fit the minimal requirements outlined above.

A final note about software, hardware, and networks: for the proper polling functionality which is so critical to LOCKSS functionality, the LOCKSS Program has recommended that a minimum of 6-7 replicas (LOCKSS boxes) exist within a private LOCKSS network (e.g., the MetaArchive or GPO projects) and participate in all polls. The production system of LOCKSS has over 100 boxes but not all have all the possible content collections. LOCKSS system parameters are set so that a typical poll in the system has 7-10 votes.[19]

## Architecture Options Considered

The original LOCKSS software was based on the Linux Router Project (LRP) platform, a boot floppy containing a minimal, but functional Linux system in a RAMdisk.[20] It was capable of downloading and installing the LOCKSS daemon and the software on which it depended, such as

[18] Installing LOCKSS – Computer Specifications <http://www.lockss.org/lockss/Installing_LOCKSS>
[19] Reich, "Editors' Interview with Victoria Reich, Director, LOCKSS Program."
[20] Rosenthal, "A Digital Preservation Network Appliance Based on OpenBSD."

the Java Virtual Machine that would not fit on a floppy.  To begin running LOCKSS, the local institution had to also download a Windows program that formatted, wrote, and checked a generic version of the floppy.  When the PC was booted from this floppy, the local operator had to input necessary configuration information that personalized the floppy, partitioned the disk, and created the necessary file systems on it.  UNIX configuration skills were also needed to recover from a compromised or damaged LOCKSS system, making the task of recovery much more complicated and vulnerable to human error.

In order to eliminate the human error factor and make the system easier to run, the system was redesigned to utilize an OpenBSD install CD rather than using Linux.  The modified OpenBSD system could fit on a single floppy (now CD) and be write-locked.  Moving to this system enabled LOCKSS to continue functioning as designed, but simplified the operator side of two key LOCKSS tasks performed at the local level:  installing and configuring a new system and recovering from a compromised system.  New system installation can now be done by almost anyone – programmer experience is no longer necessary.  In addition, recovering from a LOCKSS local system problem can now generally be managed by rebooting from the floppy/CD.

#### 3.3.3.2 ***Data Security (Access Controls)***

**Authorization and Authentication**
Authorization and authentication of users to access LOCKSS content are managed by the local institution's authorization and authentication systems.  LOCKSS boxes have to be integrated into the local institutional network in order for LOCKSS to work properly.  If done correctly, authorized readers from an institution can access content stored within a LOCKSS box when the publisher's (content owner's) Website is not available for any reason (subscription canceled, network traffic, publisher server down, etc.).  When access to the content owner's Website is interrupted, the LOCKSS software steps in and redirects user requests to the LOCKSS cached content, in effect acting as a proxy server for the content owner's site.

**Physical Security**
Physical security, i.e.., controls and protections against unauthorized access to the LOCKSS box, is the responsibility of local implementing institutions since these are deployed locally.

#### 3.3.3.3 ***Data Deposit & Ingest***

**Content Acquisition, Source Files & File Formats**
LOCKSS software was designed to be format-agnostic.  Content is collected as it exists in the Web-based environment via Web harvesting (crawling) and Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH).  Original formats are ingested as is (or as published if it is an e-journal).  Where plug-in software is required to render a format, that software is ingested as well.  Finally, in cases where content is published in several formats (HTML and PDF), LOCKSS will ingest all formats and copies.  This information is bit preserved, as received.  The goal is to preserve the "look and feel" of the e-journal content when it was captured.  LOCKSS does not preserve interactivity that is delivered via Web services from publishers' Websites (e.g. AJAX, SOAP).

In the LOCKSS system SIPs are created by the publisher, who places a "publisher manifest page" containing metadata on their Website and publishes the URL. A participating organization's LOCKSS system will then, as directed by the authorized administrator, collect via HTTP the entire SIP containing the content and the "publisher manifest page" and store it together with all available HTTP header information (including MIME type). This information is sufficient at the time of collection for a browser to render the content. Packaging Information is encoded in instances of Java classes implementing the LOCKSS plugin API. In most cases this is a generic implementation driven by XML files.[21]

**Unique Identifiers**
LOCKSS software consistently names all harvested content with the exact URL from which it was collected. This applies both internally (within an AIP) and externally (in a SIP and a DIP). Multiple versions of content collected from the same URL are disambiguated using the time of collection.

**Validation**
Validating ingested content has two aspects. First, there is a validation that the LOCKSS or CLOCKSS box has collected everything that the publisher made available. The set of files the LOCKSS box will collect is controlled by a set of crawl rules created specifically for the publisher's site, part of what is called the" LOCKSS plug-in" for that site, and by the publisher's manifest page. The release process for the plug-in includes rigorous tests including auditing the set of files collected from the publisher's site. Failure to collect files specified by the rules generates an alert to the host institution.

A second and more difficult validation is whether the publisher has made everything available that was published. The LOCKSS team is developing techniques to perform this validation, using the crawler's ability to find all user-visible URLs, OAI-PMH queries and other sources of information.

Finally, a correctness determination is based on integrity checks of what LOCKSS collected versus what is available on the publisher's Website and other LOCKSS boxes. Differences are resolved by re-crawling the publisher's site; any difference that cannot be resolved in this way generates an alert to the host (participating) institution.

**Metadata Capture**
As a proxy Web cache system, LOCKSS is not designed to be an independent repository with searchable content. As such, the types and extent of metadata captured differs from traditional digital preservation repository implementations.

Descriptive metadata in the LOCKSS system consists of the URLs at which the information was originally published, and the searchable information they contain including metadata and the full text, all provided directly by the publisher. Information such as title, publisher, date, subject, coverage, etc., is collected within the "LOCKSS core metadata." (See analysis at http://dlibcenter.iei.pi.cnr.it/Metadatadictshort22703%20a4.pdf). Other descriptive information

[21] Email from David Rosenthal.

may be compiled and made available to local OPACs, but that responsibility falls to the local implementers. LOCKSS is not a searchable system, but a proxy replacement server to the content normally provided via the publishers' Websites.

LOCKSS does capture some representation information. Defined as all of the additional information (metadata) required to convert bit sequences into more meaningful information, LOCKSS relies on its browser-renderable design to determine what is collected and recorded in the Archival Information Packages (AIPs). Renderability from LOCKSS boxes relies upon a browser (which renders known file formats), so LOCKSS does capture and record MIME type, as well as package the LOCKSS browser plug-in API into the SIP. Other types of detailed file-level representation information are not recorded or utilized within the LOCKSS system.

#### 3.3.3.4 ***Archival Storage***

Archival storage functionality is partially defined within LOCKSS software capabilities, as well as the PC and network environment on which a LOCKSS box resides.

Internally, the LOCKSS system stores content in a repository defined by a set of Java classes. The AIP consists of instances of these classes, representing the content itself, metadata obtained from the publisher manifest page and the HTTP headers, and an instance of a Java class implementing the LOCKSS plug-in API encapsulating metadata not obtained from these sources. This instance is normally driven by externalized metadata in the form of XML files.

Physical archival storage capability within the PC is determined by the size and processing capacity of the device. As mainly text-based and small image files at this time, a LOCKSS box can start as small as 250 Gb. This is a vast difference from the multi-terabyte servers in use in full digital preservation repositories. It is important to underscore that the LOCKSS software can support both parallel ATA (PATA) drives and serial ATA (SATA) in native mode to allow for increases in storage size, but does not support the use of RAID.

#### 3.3.3.5 ***Preservation Planning (Strategies)***

**General**

The LOCKSS team has chosen to incorporate and address preservation strategies in several ways, including through the system design.

First, the LOCKSS approach deploys a large number of independent, low-cost computers with persistent Web caches that cooperate to detect and repair damage by voting in "opinion polls" on their cached document. This is the truest application of the "Lots of Copies Keeps Stuff Safe" principle. Low-cost computers can fail, but the LOCKSS software polling system would detect this and alert local institutions to their problem. Once replaced, the LOCKSS software communicates with other LOCKSS boxes to replace the content which had previously been a part of the auditing/polling circle. This ensures that only authorized content is reloaded in the even of damage detection or replacement.

Secondly, LOCKSS controls preservation complexity by controlling formats ingested. LOCKSS is designed to work with certain kinds of content. Specifically, it works for content that is available on the Web and renderable through a browser. At the time of collection, the content harvested as the SIP is sufficient for a browser to render the content; because it is collected in exactly the same way that a browser would access it. The LOCKSS system's DIP replicates the SIP exactly by acting as a proxy for the original SIP to the Designated Community, so the information preserved is independently understandable (i.e. can be rendered in any browser without the need for user intervention).

Migration of AIPs is not a part of the LOCKSS functionality. Instead, LOCKSS will incorporate a migration upon access strategy for the long-term rendering of the bits preserved in LOCKSS boxes. This migration on access strategy was demonstrated in a proof-of-concept project in 2005 though a series of APIs need to be created and maintained to enable links to known format converters to enable to conversion on the fly capability.

Because of the nature of LOCKSS content, these strategies seem appropriate. It would be advantageous for LOCKSS to enable linking to international format registries when these are available. The LOCKSS team has agreed, however, to consider incorporating functionality with emerging format registries that provide links to registered format converters, when such registries are available for production use, potentially saving time and the expense of other potential solutions.

### 3.3.3.6 *Data Management (Metadata, Logs, etc.)*

**Management of Files**

The LOCKSS software creates and records an extensive list of process logs. For purposes of management and storage, some of the most important logs generated as related to LOCKSS box polling. Logs with large numbers of poll requests from previously unknown peers might lead to potential attackers who should be blacklisted from the network polling. Logs and alerts also enable local LOCKSS administrators to identify, determine, and remedy potential problems with local content caches.

Some logs/statistics are managed at the network level, but each LOCKSS box logs all its own activities locally. Access denials to the dissemination and management interfaces are logged through the logs that are currently preserved for a limited period. LOCKSS is working to provide usage and content delivery statistics to LOCKSS box administrators although this was not yet fully implemented at the time of audit.

**Metadata**

LOCKSS collects a minimal amount of descriptive and technical metadata associated with the content. This metadata is “packaged” as a component of SIP and maintained in the AIP. This information is described in 2.3.3.3, *Data Deposit and Ingest: Metadata*.

**Fixity**

LOCKSS software implements content cache fixity through the mutual auditing protocol which supplies regular assurance that the content agrees with other replicas. Regular hash checks do not occur on individual boxes, but take place within the network polling. Since the “fixed” content for any title or Archival Unit is the same on all LOCKSS boxes (as harvested from the publisher’s Website), this fixity is the same for all LOCKSS box AU instances (caches).

#### 3.3.3.7 ***Access Management***

**Accessible content**

The LOCKSS system disseminates information locally by acting as an HTTP proxy, making it appear to users that the SIP is still available from its original URLs (with any changes required by preservation operations such as format conversion).[22] The entire SIP, including the publisher manifest page with its metadata, is available.

**Access Management Functionality**

The LOCKSS software implements the strict access requirements agreed to by publishers and the libraries subscribing to the publisher's content and caching that content on various LOCKSS boxes. LOCKSS boxes do not implement "LOCKSS-wide" access in the case of publisher failure. In fact, caching rights are determined by institutional subscriptions with publishers. If an institution has a subscription to a publisher's content, it is able to cache that content on a LOCKSS box. In the event of any of the specified trigger events, those subscribing institutions with LOCKSS boxes would be able to provide their own communities authorized access to their local copies of the content.

#### 3.3.3.8 ***Business Continuity, Environmental Management, and Disaster Planning***

LOCKSS is a highly decentralized repository system. While LOCKSS was designed to support business continuity to the e-journals offered on the publisher Websites, all other aspects of business continuity are applicable only at a local level, not at the broader network or LOCKSS Program level.

Environmental management issues and disaster planning responsibilities also occur at the local institution level.

[22] *LOCKSS and OAIS: Formal statement of conformance to ISO 14721:2003*. <http://www.lockss.org/lockss/Oais>

### 3.3.4 Vulnerabilities

#### 3.3.4.1 ***Significant Repository Events***

Publications by Reich and Rosenthal reveal detected failures at certain levels of the system over the last few years. These include failures related to ingest, disk storage, and the local PCs. That being said, the system proved its capability of recovery from such failures, owing to the distributed nature of the collection (available directly from publisher sites) and the design principles behind the LOCKSS Program and LOCKSS software. Multiple instances of the same content allowed for polling and identification of failure points, and permitted recovery of full archived content from system participants.

#### 3.3.4.2 ***Threats and Liabilities***

While LOCKSS has competently addressed the issues of system design, functionality and security an important question raised by the audit was the financial sustainability of the LOCKSS Program and the LOCKSS Alliance. Although the fee-based membership Alliance was formed to provide stability and financial support for the LOCKSS Program, at the time of the audit it had yet to reach a level sufficient to put the program on a sound economic footing. To support the ongoing costs of LOCKSS' dedicated support and developmental staff and continued negotiation of new agreements with publishers to increase LOCKSS-compatible content, LOCKSS membership revenue will have to grow well beyond its 2006 level.

While a key component of the LOCKSS business plan has been the development of a supportive Open Source community to assist with further technology development and support, a community sizable enough to do so has yet to materialize. Should the LOCKSS Program cease to exist, content archived to that point, however, will still remain available on the distributed LOCKSS boxes, but less tech-savvy institutions would likely be incapable of continuing to manage that content.

LOCKSS recently reported, however, that in 2007 the Alliance generated sufficient income to "cover all the LOCKSS-related activities of the Stanford team." If true, this would indicate that LOCKSS has become self-sustaining, independent of grants and other soft money subsidies, a very promising development.

### 3.3.5 Final Observations and Recommendations

Overall, the LOCKSS software appears to be a solid technology enabling the preservation of a range of digital materials, although adopting institutions should be aware that LOCKSS technology is not a panacea for preserving all types of digital content. That being said, for the content it currently targets – Web-delivered e-journals and some Web site content– it has proved capable of providing persistent access to collections within LOCKSS boxes and the broader LOCKSS networks.

The concept behind LOCKSS – a collaborative partnership enabling digital preservation – is one of its strong points. As an inexpensive implementation (less than $3000 per instance, in general, including hardware costs), it also enables smaller institutions to combine resources and begin to preserve fragile, born digital content in a way they would otherwise be incapable of accomplishing.

In using LOCKSS to preserve e-journal content LOCKSS is distinguishable from other e-journal archiving solutions in another respect: the content is kept and maintained locally. Institutions can use LOCKSS to provide perpetual access to licensed content in a way that would enable an immediate and seamless transition, continuing user access – even on a temporary basis – in the event that content became unavailable from the publisher. Other options such as the Portico e-journal archiving service and the Koninklijke Bibliotheek's e-journal archive in their e-Depot have agreements with publishers which preclude access to their archived content unless significant trigger events occur. Even after a significant trigger event occurs, opening access through these other services requires a waiting period and notification of publishers prior to opening the archives. Since both of these services provide a "once opened, always opened" clause, the negotiation to open the archive services may be protracted, prohibiting user access to content for a significant period of time. From a user perspective, quick access through a local LOCKSS box proxy server seems like a preferable arrangement.

Because LOCKSS and private LOCKSS networks are becoming more important to the adopting institutions, the absence of solid, sustainable funding to keep the LOCKSS Program and LOCKSS software development ongoing is troubling. The development of the LOCKSS Alliance has created the foundation for a sustainable business model, and LOCKSS reports that in 2007 it realized adequate revenue to cover all of the LOCKSS maintenance and development costs of the Stanford team. It was not possible during our audit, however, to independently determine the financial status or condition of the LOCKSS program.