Cyber Fast Track: Redundant Array of Independent Clouds

DARPA-RA-11-52 Cyber Fast Track

- Principal Investigators:
  - Zooko Wilcox-O'Hearn <zooko@LeastAuthority.com>
  - Daira Hopwood <daira@LeastAuthority.com>
- Organization Name:

  Least Authority Enterprises OTHER SMALL BUSINESS
- Technical & Administrative Contact:

  Zooko Wilcox-O'Hearn 3450 Emerson Ave. Boulder, CO 80305-6452 Phone: 303.543.2301
- Place Performed:

  Boulder, CO

Executive Summary

Cloud computing is critical for a growing number of applications. It offers cost advantages, it allows the outsourcing of technology management to specialists, and it simplifies remote access and shared access to the service.

These advantages are so compelling that cloud computing is rapidly spreading into more fields of use, despite increasing concern about the dangers. Even organizations that have traditionally prioritized strict control over their computer resources are struggling to determine how they can take advantage of the cloud without introducing unacceptable vulnerability.

In addition, cloud computing is spreading into organizations ahead of official approval. The great benefits of the cloud, its accessibility via the Internet, and its ease of use mean that employees are adopting it and coming to rely upon it without the approval or knowledge of their IT managers.

In the proposed research, we focus on the *storage* component of cloud computing services.

We propose to develop a practical way to gain *provider-independent security*—also known as "host-proof" security—while leveraging widely-used commodity cloud storage services from companies such as Amazon, Rackspace, Google, and Microsoft.

By extending existing open-source storage software to use multiple cloud storage services, we will construct a novel distributed data structure that we term *Redundant Array of Independent Clouds* (*RAIC*). With *RAIC*, even if a subset of the services were to simultaneously fail or be taken over by an attacker, the availability of the array would not be compromised. In addition, even if *all* of the cloud services in the array were taken over by an attacker, the integrity, confidentiality, and access-control properties of the array would not be compromised.

Upon completion of the project, we will release an implementation of *provider-independent* secure access to a Redundant Array of Independent Clouds, with four connectors, allowing back-end service on Amazon Simple Storage Service (S3) [1], OpenStack Object Storage [2] (sold commercially as Rackspace Cloud Files [3]), Google Cloud Storage [4], and Windows Azure Storage [5]. We will also release the interface, documentation, and tools that can be used to extend the technology to other back-end services.

In addition to releasing the source code and documentation for these tools to the public under the terms of an Open Source/Free Software licence, we intend to launch a new commercial product, leasing a *Redundant Array of Independent Clouds* to customers.

The research outlined in this proposal will be conducted by two principal investigators: Zooko Wilcox-O'Hearn and Daira Hopwood, both well-known security researchers with extensive experience in secure online storage. This research is projected to span 24 weeks with a combined cost of $XXX including labor, materials, and transportation.

Technical Description

The Problem

Cloud storage services reached the mass market with the introduction of Amazon S3 in 2006. Since then growth has been rapid. There are many commercial cloud storage service providers now operating, including Amazon, Rackspace, Google, and Microsoft. The total amount of data on cloud platforms has been increasing rapidly year after year. Many popular new web sites and applications were designed for cloud storage and have never been deployed with any other kind of storage.

It is widely understood that this introduces security issues—users of cloud storage typically rely utterly on the host to protect the confidentiality and integrity of their data. If the host is compromised—such as by an *Advanced Persistent Threat*, by a social engineering attack (e.g. bribing or coercing an employee of the hosting organization), or by a software exploit—then the attacker gains the ability to undetectably read all of the user's data and also to modify or delete that data.

A demonstration of the vulnerability inherent in this approach was seen on June of 2011 when Dropbox (built on top of the Amazon S3 cloud storage platform) accidentally turned off user verification, making it possible for all of the files of all of their 25 million users to be read or altered by anyone over the Internet. This window of opportunity lasted for four hours before the error was discovered and corrected [6].

Not only are users of cloud storage unable to prevent such failures or attacks, but they are also excluded from effective insight into the service provider's operations and internal policies, records of known incidents, and other information which could help them to estimate and manage the security risks to their data.

Cloud storage concentrates assets from many different users. This makes it a more valuable target for attackers, who stand to gain more from penetrating a cloud computing provider which hosts the assets of many users than from penetrating any one user's own infrastructure. Incident response may be complicated by the fact that a number of users may have been hit simultaneously.

The unsolved security issues have not prevented an aggressive push for cloud computing from the highest levels of government.

In 2011, the Office of the Federal CIO announced the Federal Cloud computing Strategy [7], requiring all new programs (unless specifically exempted) to use private-sector clouds, and allocating *$20 B* for cloud computing migration out of a total budget of $80 B. It was estimated that the new focus on cloud computing would *save $5 B* per year.

A few months later, the risk of *added costs* were illustrated when the Department of Defense became a defendant in a lawsuit asking *$4.9 B* in damages. The Department of Defense had contracted the storage of health data to a private sector enterprise, which stored it unencrypted and thus allowed it to be exposed when it was subsequently stolen [8].

In December of 2011 the National Defense Authorization Act for Fiscal Year 2012 [9] became law. It requires all departments of the Department of Defense to migrate "Defense data and government-provided services from Department-owned and operated data centers to cloud computing services generally available within the private sector that provide a better capability at a lower cost with the same or greater degree of security."

Arguably, the "same or greater degree of security" criterion can be met only by a storage system with

provider-independent security, since a system without this property would necessarily have additional vulnerabilities to storage providers relative to the current situation.

Is it possible to gain the benefits of cloud storage without losing control over who can read and edit your files? In the proposed research, we attempt to answer this question in the affirmative.

Our Approach

Tahoe—the Least-Authority File System

We propose to extend the Tahoe "Least-Authority File System", an open-source platform that implements remote storage with *provider-independent security*. We are major contributors to the design and implementation of Tahoe-LAFS and we already understand its design and implementation.

Tahoe-LAFS performs encryption and integrity-checking of all data on the client side. It includes fine-grained and dynamic cryptographic access-control which allows the sharing of specified subsets of files and directories with explicitly chosen recipients. It implements immutable files, read-only access to mutable files, and transitive-read-only "views" into a subtree of directories and files [10].

Tahoe-LAFS is a known and respected secure storage system. It is distributed by popular open source operating systems such as Debian, Ubuntu, Slackware, and NetBSD [11]. Academic research papers describing Tahoe-LAFS have been cited more than 30 times [12]. Tahoe-LAFS received an unsolicited recommendation when the *National Cyber Leap Year* program, organized by NSA and including researchers sponsored by DoD and DARPA, praised it and stated:

> "As a specific example, we wish to highlight the Tahoe grid file system, a cross-platform open-source software solution which demonstrates both secure chunking and redundant data decentralization.
>
> …
>
> Tahoe promotes an explicitly secure, fault- tolerant model: stored files are broken into pieces, encrypted, and the pieces are redundantly stored across arbitrarily many servers.
>
> …
>
> Wider deployment of this type of file storage system would have an immediate impact on the quality of modern data protection. Built-in fault tolerance lowers server costs by allowing any machine with an excess of unused disk space to join the storage grid; because files are encrypted prior to storage, the individual storage grid nodes need not be trusted. Most important, by spreading data across a number of (potentially heterogeneous) machines and coupling the process with strong encryption, data storage as a whole is transformed into a moving target.
>
> …
>
> Systems like Tahoe are making these methods immediately usable for securely and availably storing files at rest; we propose that the methods be further reviewed, written up, and strongly evangelized as best practices in both government and industry." — National Cyber Leap Year Summit 2009—Co-Chairs' Report [13]

Redundant Array of Independent Clouds

The current implementation of Tahoe-LAFS uses custom servers for persistent storage of the ciphertext. They are denoted "Tahoe-LAFS storage servers" in this diagram of the architecture:

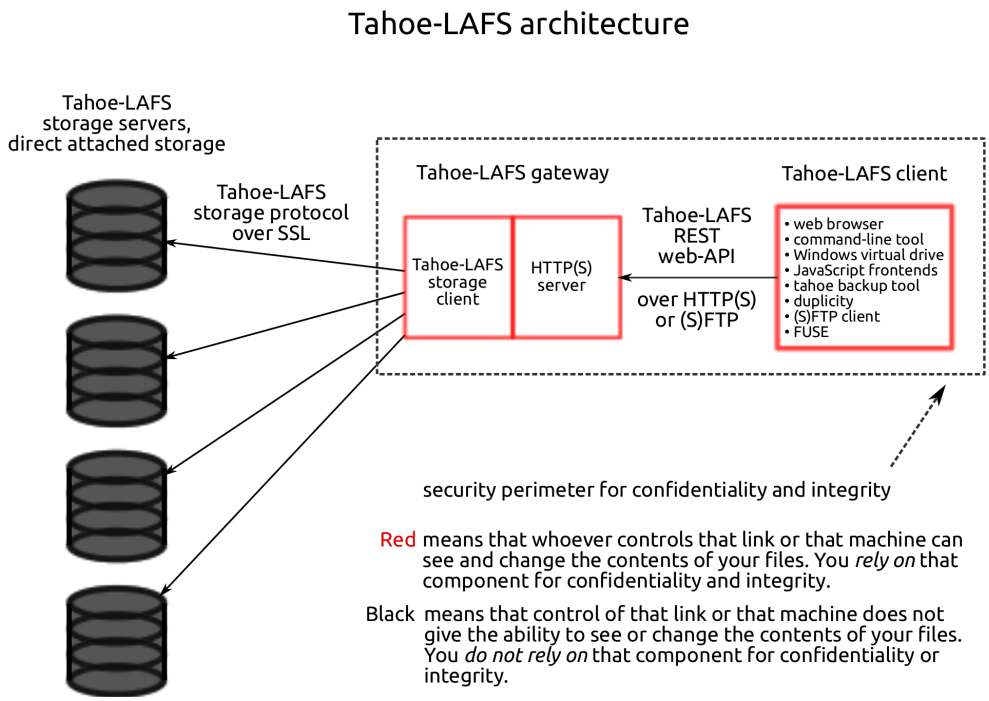

Figure 1: diagram of Tahoe-LAFS architecture

We propose to replace those components with connectors to established cloud storage service providers. This allows users to choose cloud service storage providers based on business and administrative considerations such as cost, scalability, service level agreements, and legal mandates, while retaining the security properties uniquely offered by Tahoe-LAFS.

RAIC architecture

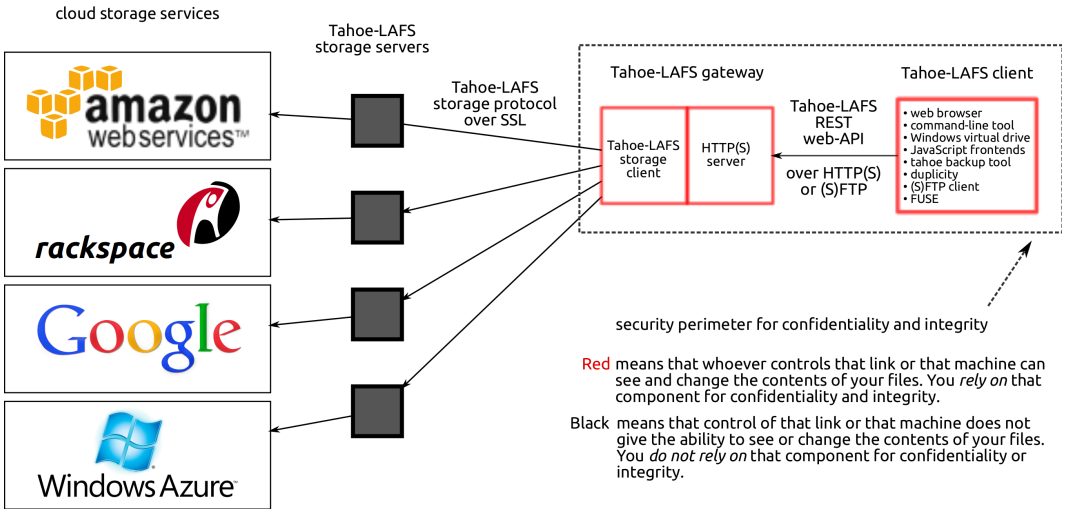

Figure 2: diagram of proposed RAIC architecture

In this approach the "Tahoe-LAFS storage server" nodes still exist, but they no longer store data persistently. Instead they serve as proxies between the Tahoe-LAFS storage protocol (on their right in this diagram) and the specific protocol of their cloud service (on their left).

This architecture creates a novel kind of fault-tolerance across multiple clouds. If a subset of the cloud service providers suffers an outage of availability, whether due to accident or attack, the RAIC continues to provide full availability to its users.

At the same time, it preserves all of the integrity and confidentiality properties of the original Tahoe-LAFS architecture, without which this sort of cross-cloud fault tolerance would not be possible.

Capability / Technology Information

This is the first solicitation for which this capability and technology have been proposed.

Interactions with the Ad-Hoc Cyber Research Community

Principal Investigators

The proposal is led by two principal investigators, each with significant research and commercial security expertise, particularly in the realm of cloud security.

- Zooko Wilcox-O'Hearn is a well-known security expert and researcher. While his research interests span many topics within the security domain, he has deep expertise in cloud storage. He is known for his work on DigiCash, Mojo Nation, ZRTP, and more. He is one of the co-founders of the Tahoe-LAFS free/open-source software project.

- Daira Hopwood participated in the standardization of the TLS protocol and Internationalized Domain Names, found security bugs and design flaws in Java Virtual Machines, wrote code for the Cryptix cryptography library, and did security auditing for the Caja Secure JavaScript project. She is a major contributor to the Tahoe-LAFS project. In her spare time, she is designing a capability-secure programming language codenamed "Noether".

Prior Interaction

Both PIs have experience participating in the cyber security research community. The following represents the subset of the PIs recent research presented in the ad-hoc cyber research community that is relevant to this proposal in the field of cloud storage:

- **Strange Loop: Emerging Languages Camp 2013** [14] St. Louis, Missouri, USA Daira Hopwood
- **Applied Cryptography and Network Security 2013** [15] Banff, Alberta, Canada, Zooko Wilcox-O'Hearn
- **USENIX Vail Computing Elements Workshop 2013** [16] Vail, Colorado, USA Zooko Wilcox-O'Hearn
- **ECRYPT II Crypto For 2020** [17] Tenerife, Spain Zooko Wilcox-O'Hearn and Daira Hopwood
- **7th International Digital Curation Conference (IDC11) Domain names and persistence workshop 2011** [18] Bristol, U.K. Daira Hopwood
- **CONFidence 2010** [19] Kraków, Poland Zooko Wilcox-O'Hearn
- **RSA Conference 2010** San Francisco, USA Brian Warner and Zooko Wilcox-O'Hearn
- **USENIX Conference on File And Storage Technologies (FAST) 2009** [20] San Francisco, USA James Plank, Jianqiang Luo, Catherine D. Schuman, Lihao Xu, Zooko Wilcox-O'Hearn

Future Interaction

We anticipate that the results of the research performed as described in this proposal will result in presentations at top tier conferences in the boutique cyber security research community, software releases available to the security community and general public, and published reports and other materials detailing the findings of the research.

Metrics

How Many Cloud Storage Services Are Supported?

The primary quantitative measure of success for this program is the number of cloud storage plugins that are fully implemented. The goal is to implement four cloud storage backends: Amazon S3, OpenStack Object Storage / Rackspace Cloud Files, Google Cloud Storage, and Windows Azure Storage.

A cloud storage backend plugin is considered complete when it meets all of the functional requirements and quality requirements listed below.

Functional Requirements

- *Upload*: an encrypted data share can be uploaded to a Tahoe-LAFS storage server configured with the plugin and the data is stored to the cloud storage backend.
    - *Scalable shares*: there is no hard limit on the size of encrypted share that can be uploaded.

      If the cloud storage backend offers scalable files, then this could be implemented by using that feature of the specific cloud storage backend. Alternately, it could be implemented by mapping from the LAFS abstraction of an unlimited-size immutable share to a set of size-limited files in the cloud storage backend. See Task 1—Design mapping between LAFS shares and cloud files, below, for more detail.
    - *Streaming upload*: the size of the encrypted data share that is uploaded can exceed the amount of RAM and even the amount of direct attached storage on the storage server. I.e., the storage server is required to stream the data directly to the ultimate cloud storage backend while processing it, instead of to buffer the data until the client is finished uploading and then transfer the data to the cloud storage backend.
- *Download*: an encrypted data share can be downloaded from a Tahoe-LAFS storage server configured with the plugin, and the data is loaded from the cloud storage backend.
    - *Streaming download*: the size of the encrypted data share that is downloaded can exceed the amount of RAM and even the amount of direct attached storage on the storage server. I.e. the storage server is required to stream the data directly to the client while processing it, instead of to buffer the data until the storage backend is finished serving and then transfer the data to the client.
- *Modify*: an encrypted data share can have part of its contents modified.

  If the cloud storage backend offers scalable mutable files, then this could be implemented by using that feature of the specific cloud storage backend. Alternately, it could be implemented by mapping from the LAFS abstraction of an unlimited-size mutable share to a set of size-limited files in the cloud storage backend. See Task 1—Design mapping between LAFS shares and cloud files, below, for more detail.
    - *Efficient modify*: the size of the encrypted data share that is being modified can exceed the amount of RAM and even the amount of direct attached storage on the storage server. I.e. the storage server is required to download, patch, and upload only the segment(s) of the share that are being modified, instead of to download, patch, and upload the entire share.
- *Tracking leases*: The Tahoe-LAFS storage server is required to track when each share has its lease renewed so that unused shares (shares whose lease has not been renewed within a time limit, e.g. 30 days) can be garbage collected. This does not necessarily require code in every backend because the lease tracking can be performed in the storage server's generic component rather than in each backend.

Quality Requirements

- *Unit tests*: all code contributed to the Tahoe-LAFS project is required to have thorough unit tests. To meet this standard, we develop the code and the unit tests together, using a code coverage tool to show us visually which lines of code are executed by the unit tests.

- *Documentation and open source publication*: We will contribute all of the implementation source code to the Tahoe-LAFS project under the terms of its Free Software/Open Source Licences. This maximizes the opportunities for peer review including security auditing by open source contributors, for benefit to the public, and for other works to be built on top of this one. It also eliminates barriers to government use of the product.

  To have a chance of acceptance into the Tahoe-LAFS project, we have to follow that project's coding standards and quality standards, including thorough developer-oriented and user-oriented documentation.

- *Failure handling*: handling of failures from the cloud storage backend, either by retrying or by raising an informative exception (in addition to the logging mentioned above).

- *Statistics and logging*: the storage server exports operational statistics about performance of the cloud storage backend and a record of exceptions or failures from the cloud storage backend.

The quantitative measure is how many cloud storage backends meet this standard of completeness and quality.

Statement of Work

The goal is to implement *Redundant Array of Independent Clouds*, including streaming upload and download, scalable modify, and lease-tracking, for four major cloud storage services.

The work for this project is broken into six phases, one for design, one for each of the four cloud storage backends, and one for the lease tracking.

All phases and tasks will be conducted by the PIs or other representatives of Least Authority Enterprises.

Note that the work to satisfy the Quality Requirements — *Unit tests*, *Documentation and open source publication*, *Failure handling*, and *Statistics and logging* — will not be performed in a separate task but will be a part of every task, since we have a policy of implementing those quality measures at the same time as writing the initial code.

Task 1—Design mapping between LAFS shares and cloud files

Task 1 is to design the mapping between LAFS files and each cloud storage backend.

The deliverable of Task 1 is a document describing how RAIC will map from LAFS mutable and immutable shares to the cloud storage backend, for each cloud storage backend, in terms of the specific API calls offered by that backend. (See notes: catalog of features offered by different cloud storage backends below for those APIs.)

For each mapping, this document will analyze the costs for each of the functional requirements. The costs include the following:

- network usage—bandwidth and number-of-round-trips
- disk usage—bandwidth and estimated number-of-seeks
- storage—including not-yet-collected garbage
- API usage—cloud storage backends typically charge a small fee per API call

This design document will also answer the following questions:

Are mutable and immutable implemented the same or differently?

Each LAFS Cloud Storage Server has to map each LAFS share—which it is responsible for storing—to the server's cloud storage backend. The requirement for efficient modification of mutable files imposes strenuous constraints on how the LAFS Cloud Storage Server does this—the LAFS server is required to mutate part of the contents of a mutable file without rewriting the entire file.

The requirement for streaming upload, of both mutable and immutable files, also imposes a less restrictive constraint. The LAFS server is required to write out the initial part of the file to the cloud storage backend before the LAFS server has received later parts of the file.

It may turn out in the performance of *Task 1* that a technique which satisfies the more difficult *efficient modify* requirement also satisfies the *streaming upload* requirement, in which case it is a more efficient use of developer resources to implement one solution that satisfies both uses, instead of separate solutions for mutable and immutable files.

Or it may turn out that using a different technique for immutable files has some engineering or efficiency advantage over using the same technique for both. This decision will also interact with Are different cloud storage backends implemented the same or differently?, below.

Are different cloud storage backends implemented the same or differently?

In addition, the LAFS Cloud Storage Server will have to either take advantage of extended functionality offered by some but not all cloud storage backends (such as mutable files, multipart files, and resumable uploads), or else implement its functionality based on only the minimal functionality—common to all cloud storage backends—of limited-size, immutable files.

It may turn out in the execution of *Task 1* that implementing in terms of only limited-size, immutable files turns out to be necessary for some cloud storage backends, and that therefore it is a more efficient use of developer resources to implement a *generic* LAFS Cloud Storage Server which satisfies all of the functional requirements using only limited-size, immutable files. That generic LAFS Cloud Storage Server can then be targeted to each specific cloud storage backend with a simple mapping to that storage backend's immutable file support. We will also take into account the possible advantage that relying on a limited set of features will help if in the future someone extends the *RAIC* idea to support other cloud storage services.

Alternately, it may turn out that mapping the different kinds of LAFS shares to features offered by the different cloud storage backends offers engineering or efficiency advantages.

Notes: catalog of features offered by different cloud storage backends

- *Amazon S3*

  Amazon S3 offers support for scalable immutable files and streaming upload by dint of a *multipart upload* feature (S3 multipart upload, developer guide [21], S3 multipart upload, API reference [22]). It also offers S3 server side copying [23], which *might* be able, combined with the multipart upload feature, to optimize out the network usage costs (but not the disk usage costs) of simulating mutable files by copying. It does not offer any first-class mutable storage.

- *OpenStack Object Storage*

  Very like Amazon S3, OpenStack Object Storage offers support for scalable immutable files and streaming upload by dint of a *multipart upload* feature (OpenStack Large Object Creation [24]). It also offers OpenStack server side copying [25] which could certainly, combined with the multipart upload feature, optimize out the network usage costs (but not the disk usage costs) of simulating mutable files by copying. It does not offer any first-class mutable storage.

See also OpenStack Large Object Administration [26].

- *Google Cloud Storage*

  Unlike the first two, Google Cloud Storage doesn't offer multipart upload, but does offer Google Cloud Storage resumable uploads [27], which can support scalable immutable files and streaming upload, but can't be used to avoid network costs while simulating mutable files by copying. It does not offer any first-class mutable storage.

  See also Google Cloud Storage copy [28], which is probably not flexible enough to support simulation of mutation by copying.

- *Windows Azure Storage*

  Windows Azure Storage is different from the others. It offers two kinds of files, termed Azure block blobs and page blobs [29]. Both are mutable. Block blobs are limited to 200 GB and page blobs are limited to 1 TB. It appears that either kind would support *streaming upload*, and *efficient modify*. The file size limits—at least those of block blobs—may be a problem for some users.

Notes: Build on top of the prototype

We have already developed a prototype of this layer, which works only for the Amazon S3 backend, does not satisfy the *scalable shares*, *streaming upload*, or *efficient modify* requirements, and which implements handling of mutable and immutable shares separately.

The prototype is, however, functional, reliable, and well-made—satisfying the quality requirements of *unit tests* and *documentation and open source publication*. Developing this working prototype has proven the concept and has resulted in an abstract interface which we believe matches the needs of the full system described here.

Task 2—Create plugin for Amazon Simple Storage Service (S3)

- *Task 2a—Upload and download immutable shares:* implement scalable, streaming upload and download of shares mapped to S3 files, using the mapping strategy from *Task 1*.
- *Task 2b—Upload, download, and modify mutable shares:* implement streaming upload, download, and efficient mutation using the Amazon S3 API, using the mapping strategy from *Task 1*.

The deliverable for *Task 2* is the source code of a plugin for the Amazon S3 service.

Task 3—Create generic lease tracker

- *Task 3—Create generic lease tracker*: implement a lease-tracker service which operates generically for any cloud storage backend, testing it with the S3 backend. It needs direct-attached storage, but not highly reliable storage. It needs to be designed so that loss or corruption of its database "fails safe" by failing to collect garbage in a timely way rather than by failing to preserve non-garbage data.

The deliverable for *Task 3* is the source code of a generic lease tracker service that runs in the Tahoe-LAFS Storage Server and works with any backend.

Task 4—Create plugin for OpenStack Object Storage/Rackspace Cloud Files

- *Task 4a—Upload and download immutable shares:* implement scalable, streaming upload and download of shares mapped to OpenStack files, using the mapping strategy from *Task 1*.
- *Task 4b—Upload, download, and modify mutable shares:* implement streaming upload, download, and efficient mutation using the OpenStack Storage API, using the mapping strategy from *Task 1*.

The deliverable for *Task 4* is the source code of a plugin for the OpenStack Object Storage service.

Task 5—Create plugin for Google Cloud Storage

- *Task 5a—Upload and download immutable shares:* implement scalable, streaming upload and download of shares mapped to Google Cloud Storage files, using the mapping strategy from *Task 1*.
- *Task 5b—Upload, download, and modify mutable shares:* implement streaming upload, download, and efficient mutation using the Google Cloud Storage API, using the mapping strategy from *Task 1*.

The deliverable for *Task 5* is the source code of a plugin for the Google Cloud Storage service.

Task 6—Create plugin for Windows Azure Storage

- *Task 6a—Upload and download immutable shares:* implement scalable, streaming upload and download of shares mapped to Windows Azure Storage blobs, using the mapping strategy from *Task 1*.
- *Task 6b—Upload, download, and modify mutable shares:* implement streaming upload, download, and efficient mutation using the Windows Azure Storage blobs, using the mapping strategy from *Task 1*.

The deliverable for *Task 6* is the source code of a plugin for the Windows Azure Storage service.

Appendix A

Team Member Identification

- Zooko Wilcox-O'Hearn, US citizen; CEO of Least Authority Enterprises, LLC. a Colorado corporation.
- Daira Hopwood, British citizen; Engineer at Least Authority Enterprises, LLC. a Colorado corporation.

Government or FFRDC Team Member

None

Organizational Conflict of Interest Affirmations and Disclosure

None

Intellectual Property

Copyright on the works produced in this research will be owned by Least Authority Enterprises. Least Authority Enterprises is required by the terms of the Tahoe-LAFS software licence to open-source a work derived from Tahoe-LAFS, such as this, within twelve months of redistributing it to others or operating it as a service for others. Least Authority Enterprises currently intends to open-source the source code and documentation immediately (instead of waiting for the twelve month deadline) in order to facilitate inclusion of the results in the Tahoe-LAFS open source project.

Human Use

None

References

references

[1] "Amazon S3" Amazon (2012) https://aws.amazon.com/s3/

[2] "OpenStack Object Storage" openstack.org (2012) http://openstack.org/projects/storage/

[3] "Rackspace Cloud Files" Rackspace (2012) https://www.rackspace.com/cloud/cloud_hosting_products/files/

[4] "Google Cloud Storage" Google (2012) https://developers.google.com/storage/

[5] "Windows Azure Storage" Microsoft (2012) https://www.windowsazure.com/en-us/develop/net/fundamentals/cloud-storage/

[6] Parrish, K "Dropbox Accidently Turned Off Password for 4 Hrs" Tom's Guide (2011) http://www.tomsguide.com/us/dropbox-Arash-Ferdowski-cloud-storage-code-update-login,news-11576.html

[7] Kundra, K "Federal Cloud Computing Strategy" The White House (2011) http://www.cio.gov/documents/Federal-Cloud-Computing-Strategy.pdf

[8] Anderson, H "TRICARE Hit With $4.9 Billion Lawsuit" GovInfoSecurity (2011) http://govinfosecurity.com/articles.php?art_id=4158

[9] "National Defense Authorization Act for Fiscal Year 2012" U.S. Government Printing Office (2011) http://www.gpo.gov/fdsys/pkg/BILLS-112hr1540enr/pdf/BILLS-112hr1540enr.pdf

[10] Wilcox-O'Hearn, Z. & Warner, B. *"Tahoe – The Least-Authority Filesystem."* Proceedings of the 4th ACM international workshop on Storage security and survivability 21–26 (2008). http://www.laser.dist.unige.it/Repository/IPI-1011/FileSystems/TahoeDFS.pdf

[11] "Packages of Tahoe-LAFS" tahoe-lafs.org (2011) https://tahoe-lafs.org/trac/tahoe-lafs/wiki/OSPackages

[12] "search results" scholar.google.com (2011) https://scholar.google.com/scholar?q=%22tahoe%2Bthe%2Bleast-authority%2Bfilesystem%22%2BOR%2B%22tahoe-lafs%22

[13] "National Cyber Leap Year Summit 2009—Co-Chairs' Report" Department of Homeland Security (2009) http://www.cyber.st.dhs.gov/docs/National_Cyber_Leap_Year_Summit_2009_Co-Chairs_Report.pdf

[14] "Strange Loop: Emerging Languages Camp 2013" (2013) https://thestrangeloop.com/preconf

[15] "Applied Cryptography and Network Security 2013" (2013) http://acns2013.cpsc.ucalgary.ca/

[16] "Vail Computing Elements Workshop 2013" USENIX (2013) http://vcew.org/

[17] "Crypto For 2020" ECRYPT II (2013) https://www.cosic.esat.kuleuven.be/ecrypt/cryptofor2020/

[18] "7th International Digital Curation Conference (IDC11) Domain names and persistence workshop 2011" W3C (2011) http://www.w3.org/2001/tag/doc/idcc_workshop_programme.html

[19] "CONFidence 2010" confidence.org.pl (2010) http://2010.confidence.org.pl/prelegenci/zooko-wilcox-ohearn

[20] "USENIX Conference on File And Storage Technologies (FAST) 2009" USENIX (2009) http://www.usenix.org/events/fast09/tech/

[21] "Cloud Files"Amazon Simple Storage Service (Amazon S3) Developer Guide" Amazon (2012) http://docs.amazonwebservices.com/AmazonS3/latest/dev/uploadobjusingmpu.html

[22] "Amazon Simple Storage Service (Amazon S3) API Reference" Amazon (2012) http://docs.amazonwebservices.com/AmazonS3/latest/API/mpUploadInitiate.html

[23] "Amazon Simple Storage Service (Amazon S3) API Reference" Amazon (2012) http://docs.amazonwebservices.com/AmazonS3/latest/API/RESTObjectCOPY.

[24] "Cloud Files™ Developer Guide" Rackspace http://docs.rackspace.com/files/api/v1/cf-devguide/content/Large_Object_Creation-d1e2019.html

[25] "Cloud Files™ Developer Guide" Rackspace http://docs.rackspace.com/files/api/v1/cf-devguide/content/Copy_Object-d1e2241.html

[26] "OpenStack Object Storage Administrator Manual" OpenStack LLC http://docs.openstack.org/diablo/openstack-object-storage/admin/content/managing-large-objects.html

[27] "Google Cloud Storage Developer Guide" Google https://developers.google.com/storage/docs/developer-guide#resumable

[28] "Google Cloud Storage Reference Guide" Google https://developers.google.com/storage/docs/reference-headers#xgoogcopy

[29] "Windows Azure Storage Services REST API Reference" Microsoft http://msdn.microsoft.com/en-us/library/windowsazure/dd179355.aspx