Auditing to Keep Online Storage Services Honest

Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, Ram Swaminathan (HP Labs, Palo Alto)
Mehul.Shah@hp.com, Mary.Baker@hp.com, Jeff.Mogul@hp.com, Ram.Swaminathan@hp.com

Abstract

A growing number of online service providers offer to store customers' photos, email, file system backups, and other digital assets. Currently, customers cannot make informed decisions about the risk of losing data stored with any particular service provider, reducing their incentive to rely on these services. We argue that third-party *auditing* is important in creating an online service-oriented economy, because it allows customers to evaluate risks, and it increases the efficiency of insurance-based risk mitigation. We describe approaches and system hooks that support both *internal* and *external* auditing of online storage services, describe motivations for service providers and auditors to adopt these approaches, and list challenges that need to be resolved for such auditing to become a reality.

1 Introduction

The year is 2027. Alice and Bob are preparing a family photo album for the wedding of their daughter Carly, and want to include her baby photos. They took these photos with a digital camera in 2002, and uploaded them to the LensElephant photo-sharing service. For 25 years, they have paid LensElephant a small fee to store their photos. But today, after they click on Carly's baby-photo thumbnails to view the full files, they see nothing but an error message "Sorry, those files are corrupted."

This scenario is not just a dark fantasy. Many online services allow customers – both end users and businesses – to store data for as long as they want, and some charge for the service. Yet news reports [1, 5] reveal that even the most popular sites can lose data, and customers have no rational basis on which to evaluate the risk of data loss or to choose between storage services. Although examples in this paper focus on online storage services, the problem generalizes to the nascent "online service-oriented economy" (OSOE) in which businesses and end users purchase IT services from a variety of online service providers (OSPs). For this nascent economy to become established, customers will need ways to assess risk and gain trust in OSPs.

Third-party auditing is an accepted method for establishing trust between two parties with potentially different incentives. Auditors assess and expose risk, enabling customers to choose rationally between competing services. Over time, a system that includes auditors reduces risk for customers: when combined with a penalty or incentive mechanism, auditing gives incentives to providers to improve their services. Penalty and incentive mechanisms become supportable when risks are well understood. We believe auditing is necessary not only for traditional businesses, but also for online services to create a viable OSOE.

Auditing of OSPs is not feasible yet. First, customers are not yet sophisticated enough to demand risk assessment. Second, OSPs do not yet provide support for third-party audits.

In this paper, we describe the issues involved in making auditing online services a reality. We speculate on an insurance-based incentive system to encourage audits of OSPs. For concreteness, we focus on audit-enabling interfaces that online storage services should support.

2 What We Mean by Auditing

OSPs must convince customers that their platforms are reliable. Because a customer knows that a provider's primary incentive is to make a profit, not simply to serve the customer's needs, this is a variant of what economists call the "principal-agent problem" [14]: how does the customer know whether to trust the provider?

One way is to rely on a trusted third-party auditor, who has sufficient access to the provider's environment. An auditor understands the service level agreement (SLA) between a customer and a provider and quantifies the extent to which a provider might not meet the SLA. The auditor has expertise and capabilities that the customer does not. Auditors understand the threats posed, know best practices, and have the resources to check for process adherence and service quality. They perform these checks through well-defined interfaces to the service. With proper safeguards, auditors can investigate providers who serve multiple customers without fear of information leakage.[1]

Inspired by methods for auditing "bricks and mortar" businesses, we identify two approaches: *external auditing* and *internal auditing*. We also suggest some principles that should apply to auditing of online services.

2.1 Internal vs. external auditing

We illustrate the two types of audits using an analogy to the restaurant business. A restaurant's "SLA" is to provide its diners with enjoyable and safe meals at reasonable expense. We can audit restaurants simply by trying out the service, or (using third parties) by relying on recommendations from friends or food critics. This corre-

[1] This paragraph is adapted from [9].

sponds to *external* auditing of a service. External audits evaluate the quality of service through externally available interfaces, and usually assume that one can predict future success ("reputation") from limited samples. With this approach, we can determine whether the restaurant is enjoyable, but it might not warn us in advance if the food is sometimes contaminated.

So we also rely on health inspectors to go into the kitchen and identify procedures that might result in sick customers. This corresponds to *internal* auditing, which determines the extent to which a service follows best practices. Internal audits evaluate the structure and processes within a service to ensure that the service can continue to meet its objectives (SLAs). To do so, these audits need specialized interfaces that reveal and test the inner workings of a service. Other examples of internal audits include fire marshal inspections of office buildings to check electrical codes, and accounting firms checking whether corporations meet FASB standards. As with health inspections, these internal audits warn customers of practices that could lead to impending disasters. Customers can use this information to choose between services or complement one service with another.

We need both internal and external audits of OSPs. External audits can only confirm past behavior, so without internal audits, we could not predict upcoming problems or assess risk exposure. On the other hand, internal audits might not be exhaustive and might be based on incomplete failure models; we can use external audit results to assess whether internal audits are really working.

2.2 Auditable metrics

Auditing requires metrics. Typically, best practices determine the relevant metrics for internal audits, and the SLAs determine the relevant metrics for external audits. Metrics need not be quantitative; they may simply indicate whether the service follows a certain practice. For example, a financial audit might (among other metrics) check whether accounting data conforms with appropriate standards; a health inspection could verify that raw food is always stored below 45°F. Sections 4 and 5 describe metrics for internal and external audits of storage services, respectively.

2.3 Desirable properties

In this section, we list desirable properties for both internal and external audits. These are ideal goals, and might not all be fully achievable.

Establish standards for comparison. One purpose of auditing is to allow customers to make a rational choice. To do so, they need standards to compare the results of two different audits performed at different times, of different providers, and by different auditors. Without standards we are not even sure what is being audited.

Minimize auditing cost. Auditing imposes costs on OSPs, which should not outweigh its benefits.

Introduce no new vulnerabilities. Auditing interfaces inherently provide new ways to access an online service. These interfaces, and their implementations, should not introduce ways of attacking or corrupting the system.

Protect customer data privacy. Customers often do not want their data revealed or leaked. For OSPs that include customer privacy in their SLA, auditing approaches that reveal customer data to auditors are a vulnerability and should be avoided.

Protect proprietary provider information. OSPs seldom want to reveal internal information (including key technologies) to competitors or hackers.

Audit results must be trustworthy. Audits should not be biased either toward the OSP or the customer.

Avoid prescribing a technology. Technologies used to implement and support online services continually change. Audit practices that prescribe or prefer a particular technology could inhibit transitions to better technologies. Thus, external audits should rely on implementation-independent interfaces, if possible. Although internal audits must be technology-aware to spot flaws in technical approaches, they should adapt quickly to avoid constraining service implementors.

3 Motivating audit processes

Providers will not offer auditing interfaces unless there is motivation to do so. Mechanisms to provide such motivation are more likely to be social than technical, but we should keep them in mind when trying to design the system interfaces that support auditing.

Generally, these behavior-changing mechanisms either use penalties or incentives, or a combination. For example, regulations (and the associated fines), laws (and the threat of incarceration), or loss of reputation (which can put a provider out of business) are penalty-based mechanisms. Market forces (i.e., the ability to charge a premium for better service), or the need to obtain cost-effective insurance, can create incentives.

3.1 The role of insurance providers

Users of traditional services (e.g., home remodeling contractors) understand the value of selecting insured providers. Similarly, we believe that in many cases customers are likely to prefer insured OSPs, since it gives them both protection against SLA violations and a clear signal about which providers are trustworthy enough to be insurable.

Insurance companies will not insure OSPs unless they can quantify their risk, which gives both insurers and OSPs an incentive to use auditors. Since insurance premiums will reflect both the known risks and the uncertainty in risks, OSPs will have an incentive (reduced premiums) to improve their risk-management practices and to increase their transparency to auditors. Insurers will pass audit costs to the OSPs, giving them an incentive to improve auditing interfaces to reduce audit costs.

Others have projected this benefit for a more specific area, security-risk auditing:

> Beginning in January 2007, "financial service providers [in South Korea] will be required to insure customers' accounts to cover financial damage caused by hackers and financial accidents." ... [this] will, in effect, turn insurance providers into [security] compliance auditors. Those financial services providers who follow sound security procedures will be a better risk, and therefore see more favorable insurance rates. [7]

In theory, individual consumers could do their own external audits, but they are unlikely to be able to afford the costs and skills associated with professional risk audits. Insurance-based mechanisms are scalable; they allow an OSP to efficiently convince many customers that their SLAs will be met.

Insurance-based mechanisms are especially suitable for online storage services. We usually protect valuable physical assets (e.g. homes, jewelry) with insurance, so it is natural to extend this mechanism to digital assets. For this mechanism to work, however, we need ways of assigning value to data, which remains an open problem. For certain data, such as business assets, we may be able assign a value based on the cost incurred as a result of loss. However, this assessment is more difficult for sentimental data such as family photos.

Although insurance-based incentives may not extend to all online services, they are applicable to many OSPs, such as financial or utility computing services.

3.2 Chicken-or-egg Problem

We realize that existing OSPs will face costs to add support for auditing, both internal and external. These include additional hardware, new software, additional network bandwidth, new procedures, additional employees, and training. There is also a chicken-or-egg problem: OSPs will be reluctant to incur these costs before auditing is the norm, but auditing cannot become ubiquitous until the OSPs support it.

We see a plausible evolutionary path, however. OSPs will initially perform self-auditing to comply with regulations such as Sarbanes-Oxley (SOX); most large companies already do significant IT auditing for SOX compliance, although they probably do not yet check long-term storage processes. Once these self-audit mechanisms are in place, they can then be used as selling points, to differentiate well-run providers against their competition. Insurance companies will then also have a basis for writing data-loss policies.

3.3 Quid custodiet ipsos custodes?

Auditing depends on the competence and honesty of the auditors, properties that themselves need to be enforced. Financial auditors are already subject to regulatory and legal sanctions (for example, Arthur Anderson was forced to close its audit business after repeated instances of malfeasance, including their work for Enron). Insurers will also tend to avoid auditors who do bad jobs, as this exposes the insurers to unprofitable risks.

Likewise, insurance-based incentives depend on the competence and honesty of the insurers. We assume that OSOE insurers will be subject to existing insurance-industry regulations, and will not disappear when it comes time to pay up.

Having multiple competing auditors can help ensure trustworthy results. This implies the need for standard auditing interfaces, to avoid "auditor lock-in."

4 Internal Auditing for Storage Services

Internal auditing checks the inside behavior and processes of a provider to assess the likelihood the provider will fail to meet its SLAs. In this section, we describe the difficulty in internally auditing storage services and outline auditing interfaces online storage services can support to address the difficulty.

The SLAs for a storage service can include data integrity (customer bits are preserved), data exit (customers can get their original data), security, privacy, and more. Although auditing can apply to all of these, in this paper we mainly focus on data integrity.

4.1 Internal audits are hard

To assess the risk of violating data integrity, an auditor must understand the threats and the current best practices to prevent data loss. Unfortunately, this information is hard to obtain for storage services:

- Only a few studies describe failures and their causes in deployed large-scale storage systems [10, 12]. Baker *et al.* [2] provide a list of known threats to data preservation, e.g. natural disasters and insider attack.
- Individual organizations can learn to recognize certain bad practices that have resulted in data loss, but determining "best practices" requires comparison and experimentation. This in turn requires sharing failure information across different environments and vendors. Storage services have been reluctant to share this data.

The current situation is counter-productive for both customers and providers. Customers cannot assess service quality, and storage services can only make improvements based on internal data. Once a system with proper incentives is established, storage services will allow internal audits. Trusted third-party auditors will then be in a position to piece together this information gradually and integrate it into the audit process.

4.2 Threats and hooks for internal audits

We list three classes of threats to data integrity that audits must address. (There are others!) For each class, we touch on potential best practices and outline the hooks storage services could provide to check for adherence to these practices.

Latent faults: Many potential sources of data corruption are not immediately visible (e.g., damage from successful but undetected attacks; undetected human errors; bit rot in the storage medium). These "latent" faults require periodic checking of the data [2]. Auditors need hooks to know whether the storage service checks for appropriate sources of latent faults, whether the checks happen often enough, and whether there are enough independent replicas of the data to support successful repair when the checks uncover damage. To support auditing, the storage service could provide interfaces that access log files to verify the frequency and success of checks. Another possible hook would allow an auditor to implant triggers on random data to notify the auditor when the data is accessed for particular checks.

Correlated faults: Correlated failures increase the risk of data loss. Example sources of correlated failures include lack of diversity in software and hardware platforms, in geographic locations of storage, and in number of independently administered domains. Storage services should provide hooks that expose the level of diversity along these and other dimensions, to gauge the potential for correlated failures.

Recovery faults: Data is often more vulnerable to corruption and loss during recovery procedures, since these might be executed only rarely and are often less well debugged. Recovery also tends to take effect after things have gone wrong, when the system being recovered may be in an unanticipated state. Thus, it is important for storage services to practice regular "fire drills," (in which they deliberately disable disks, nodes or sites or deliberately corrupt or destroy data), and then measure the success of the resulting recovery mechanisms. System hooks should allow auditors to see logs of these faults and the resulting recovery activity. (Faulty recovery may corrupt unrelated data, which might only be detected through further audits.)

More importantly, storage services should provide hooks that allow auditors to test recovery safely. They should allow auditors to inject data corruption into "decoy" files, or to take disks or systems offline temporarily.

There are many open issues related to internal auditing, including:

- How much overhead does internal auditing add?
- Could internal audits cause unwanted damage?
- Beyond providing a checklist of preventive steps taken, we do not have ways to translate internal audits into a quantitative risk of *future* data loss.
- We will need ways to evolve the internal-auditing process and hooks without disrupting storage services.

5 External Auditing for Storage Services

An external audit provides an end-to-end measurement of service quality in terms of its SLA. The primary SLA for storage services is to maintain data integrity, which we quantify by the *past* rate of data loss. We obtain this rate by measuring the fraction of data lost at appropriate intervals. There are a few direct methods for measuring this fraction, such as sampling the service's stored contents or simply downloading all content. External sampling cannot detect small but important losses because it only checks a fraction of the data. Exhaustive checks are intrusive and infeasible. In this section, we detail the problems with these approaches and propose a solution to overcome these problems.

5.1 Hurdles for detecting data loss

A simple method for auditing data integrity is to sample stored data through the public read and write interfaces of a storage service. This approach requires no modification to the service. The auditor simply creates some "fake" user accounts, uploads content, and periodically extracts all of the uploaded content and ensures the result matches the original. For some services (e.g., Amazon's S3), we can access the read and write interfaces programmatically. For other services (e.g., email or photo sharing sites), we might need additional effort (e.g. screen-scraping) to crawl and download uploaded content. We cannot externally audit services that actively prevent users from downloading their content. If our premises about the advantage of auditing are correct, customers will eventually avoid such services.

This simple approach meets our guidelines but has drawbacks. First, devious services would have an incentive to identify auditors' accounts and devote more resources to those than others. Second, the approach suffers from limited coverage. Even if the service is not biased, the auditor can only introduce a limited number of fake accounts and a limited amount of data without excessive overhead to both service and the auditor. A small sample can only detect loss from failures that affect a large fraction of the data (e.g. natural disasters or major operator error).

Another option is for storage services to provide dedicated, auditor-only read interfaces that allow access to all internal customer data. The auditor can then compute cryptographic hashes (e.g., using SHA-1) of each customer data object, and then replicate these hashes (perhaps at other storage services), later using these to check the original content. Unfortunately, to detect small amounts of loss, an audit must check nearly all the data. For example, imagine an SLA that requires a storage service to maintain 1PB, and the service has lost ten 1KB blocks. If no additional damage occurs, we would need to randomly sample 40% of the 1PB to have better than a 98% chance of detecting at least 1 lost block. Since auditors will want to know the loss rate with high precision, this approach is too expensive. It requires transferring nearly all customer data, which would consume too much network bandwidth. This approach also violates our customer-privacy principle. Straightforward encryption is not a viable privacy solution, since it relies on keeping secret keys for the long term. Customers are prone to losing keys, and the auditor has no way to check if the key remains intact.

5.2 A privacy-preserving approach

We propose a solution that allows auditors to detect any change to stored data, malicious or otherwise, while adhering to our auditing principles. In our scheme, we encrypt data to hide it from the auditor, and we store *both* the encrypted data and key with the storage service. To check data integrity, our method uses a challenge-response protocol in which the service can respond correctly only if both the encrypted data and key are intact. The auditor can glean nothing about the data or key from these checks, and these checks incur little network overhead since the messages are small. Our solution also allows the auditor to assist in returning the key and encrypted data to the customer, while maintaining customer privacy. With our method, the customer need not maintain any long-term secrets or state, and the auditor needs only a little long-term state.

Our scheme is based on the following assumptions. Since many sites, such as email hosting or photo sharing, offer value-added services beyond storage, customers are willing to reveal their data to a service. Services have other legal and social incentives not to leak this data. A customer may claim data loss either innocently or fraudulently, and services have an incentive to hide data loss. The auditor does not collude with either the service or customer, because its task is to assess service quality accurately. The auditor can, however, accidentally or purposefully gain or leak private customer information.

Our solution has three phases: *initialization*, *verification*, and *extraction*. In initialization, the storage service commits to storing a document on behalf of the customer, and the auditor initializes long-term state. During verification, the auditor repeatedly checks the stored contents. Finally, in extraction, the auditor assists in returning the data to the customer in case of doubt or dispute. We sketch each of these next.

During initialization, the storage service commits to storing the key, K, and encrypted data, $E_K(D)$, after receiving these items from the customer. The service commits by publishing two values: a *data-commitment* that is a hash of the encrypted data, $H(E_K(D))$, using a secure digest such as SHA-1, and a *key-commitment* that hides the key using modular exponentiation, $g^K \bmod p$ (g is a generator of Z_p^*, and p is a large prime). Because these functions are one-way and collision-resistant, these values bind the service to the encrypted data and the key without revealing either. The service publishes these values to one or more auditors. (We also have methods to transfer the values from one auditor to the next, or make them publicly available so anyone can audit.) An auditor must remember these values, typically much smaller than the data, for all subsequent checks.

The auditor also precomputes a list of challenges and corresponding responses for checking the encrypted data. Each challenge-response pair is a random number, R_i, and a hash (implemented with keyed-hash message authentication codes — HMACs) that can be computed only by knowing R_i and the encrypted data in entirety. This list is much smaller than the encrypted data, and its contents are unknown *a priori* to the service.

During verification, the auditor must check that (a) the encrypted data is unchanged and (b) the encryption key is unchanged. For (a), the auditor randomly selects a pair from its precomputed list, sends the challenge to the service, and awaits the correct response. The auditor must discard the pair after use, otherwise the service may cheat by using previously cached results. When the auditor exhausts this list, it can generate more pairs by getting the encrypted data from the service, which is potentially expensive, and verifying the encrypted data's hash.

Our main innovation is how to repeatedly check (b) without revealing the key to the auditor. We rely on the hardness of the discrete-log and the commutativity of modular exponentiation to generate challenge-response pairs from the key-commitment. The key-verification protocol between the auditor, A, and service, S is:

1. A chooses a random β s.t. g^β is a generator of Z_p^*.
2. A sends g^β to S.
3. S computes and sends $(g^\beta)^K$ to A.
4. A checks $(g^K)^\beta = (g^\beta)^K$ else declares S **lost** key.

In steps 1-2, the auditor generates a random challenge while keeping β secret from the service. Because of the hardness of the discrete-log, the service, in step 3, must compute the response anew using the key. Since modular exponentiation commutes, the auditor uses the key-commitment, in step 4, to check the response.

For extraction, the auditor assists in returning the encrypted data simply by forwarding it to the customer after verifying its hash. The auditor also forwards the key, but before forwarding, we must first establish a shared random secret, X, between the service and customer, C, which is used to hide the key. The auditor also needs g^X to check the key. We skip the protocol for pre-arranging these values and instead focus on key forwarding:

1. S sends $K + X$ to A.
2. A checks $g^K g^X = g^{K+X}$ else declares S **lost** key.
3. A sends $K + X$ to C.

In step 1, the shared secret, X, is a "blinding" value that hides the key. In step 2, g^X hides the blind from the auditor, while allowing it to verify the key. The customer, after step 3, extracts the key by subtracting X.

Our protocols detect data loss and are not vulnerable to a cheating storage service. Given an honest auditor, the protocols either leave no doubt that the key and data are intact or reveal which party is at fault. We can prove that the auditor, honest or otherwise, learns nothing about the data contents when using these protocols. Thus, the auditor can reveal transcripts of the interaction to additional external parties, thereby providing an audit trail of the audit itself, without violating our privacy principle.

To support our protocols, storage services must export hooks for challenge-response queries and compute expensive functions for responses. Since our protocols mostly send small hashes, the main overhead comes from computing HMACs rather than network traffic. If we limit the number of checks, however, this overhead can be tolerable for a service. We measured the performance of a SHA-1 HMAC over files stored on five 500GB SATA disks with a 2-core 2GHz Intel Xeon 5130 at 362 MB/s. At this peak rate, 50 CPUs in parallel can check 1PB in 16 hours. Spread over 30 days, this work imposes less than 2% overhead. Since many large-scale storage services handle an archival workload in which most data is rarely touched, checking monthly is reasonable.

6 Related Work

Storage-related auditing: Baker *et al.* [2] describe threats to long-term storage and a reliability model, which includes the effect of latent faults and periodic internal checks of data integrity. Miller and Schwarz [13] describe an efficient method for checking online storage, but it cannot provide full coverage and privacy simultaneously. OceanStore [4] periodically checks stored data for integrity, but relies on users to keep secret keys for privacy and does not offer an interface for external auditing. LOCKSS [8] is a P2P archival system for library periodicals. Although a library site in LOCKSS periodically performs audits of its content, there is no need for complete privacy since the contents are published. Lillibridge *et al.* [6] present a P2P backup scheme in which peers request random blocks from their backup peers. This scheme can detect data loss from free-riding peers, but does not ensure *all* data is unchanged.

Auditing in general: One of us (Mogul) previously suggested that auditing support would become necessary for IT outsourcing in general [9]. Satyanarayanan proposed dealing with a variety of "Internet risks" by an audit-like mechanism ("inspection-enforced safety") based on periodic signed, encrypted snapshots of entire virtual machine states, which would then be inspected when necessary (e.g., during legal proceedings) [11]. Others have suggested building accountable systems that provide an non-repudiable history of their state and actions [15]. These audit trails are useful for detecting and pinpointing problems after the fact, and could support internal and external audits.

Click fraud scares businesses that advertise through search engines [3]. To preserve advertiser confidence (on pain of lost business), search engines internally self-audit click streams; also, advertisers can externally audit search engines by monitoring the "click-throughs."

7 Conclusion

In this paper, we motivate the need for auditing to support an online service-oriented economy. We highlight issues around both internal and external auditing and detail ways of auditing online storage services.

References

[1] M. Arrington. Gmail Disaster: Reports of Mass Email Deletions. TechCrunch, `http://www.techcrunch.com/2006/12/28/gmail-disaster-reports-of-mass-email-deletions/`, Dec. 2006.

[2] M. Baker, M. Shah, D. Rosenthal, M. Roussopoulos, P. Maniatis, T. Giuli, and P. Bungale. A Fresh Look at the Reliability of Long-term Digital Storage. In *Proc. EuroSys*, Leuven, Belgium, Apr. 2006.

[3] B. Grow and B. Elgin. Click Fraud: The Dark Side of Online Advertising. BusinessWeek Online, `http://www.businessweek.com/magazine/content/06_40/b4003001.htm`, Oct. 2006.

[4] J. Kubiatowicz et al. OceanStore: An Architecture for Global-Scale Persistent Storage. *ASPLOS*, Nov. 2000.

[5] D. Lazarus. Precious Photos Disappear. San Francisco Chronicle, `http://www.sfgate.com/cgi-bin/article.cgi?file=/chronicle/archive/2005/02/02/BUG7QB3U0S1.DTL`, Feb. 2005.

[6] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard. A Cooperative Internet Backup Scheme. *Proc. USENIX Annual Conf.*, pages 29–41, June 2003.

[7] T. Liston. Korean Financial Service Providers Required to Insure Accounts. SANS NewsBites, `http://www.sans.org/newsletters/newsbites/newsbites.php?vol=8&issue=97`, Dec. 2005.

[8] P. Maniatis, D. S. H. Rosenthal, M. Roussopoulos, M. Baker, T. Giuli, and Y. Muliadi. Preserving Peer Replicas by Rate-Limited Sampled Voting. In *Proc. SOSP*, pages 44–59, Oct. 2003.

[9] J. C. Mogul. Operating Systems Should Support Business Change. In *Proc. HotOS X*, June 2005.

[10] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure Trends in a Large Disk Drive Population. In *Proc. FAST*, Feb. 2007.

[11] M. Satyanarayanan. Inspection-enforced Internet Safety. In *NSF Workshop on Grand Challenges in Distributed Systems*, M.I.T., Sep. 2005. `http://pdos.csail.mit.edu/~kaashoek/nsf/`.

[12] B. Schroeder and G. Gibson. Disk Failures in the Real World: What does an MTTF of 1,000,000 hours mean to you? In *Proc. FAST*, Feb. 2007.

[13] T. Schwarz and E. Miller. Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage. *ICDCS*, July 2006.

[14] J. K. Shim and J. G. Siegel. *Dictionary of Economics*. Wiley, 1995.

[15] A. Yumerefendi and J. Chase. The Role of Accountability in Dependable Distributed Systems. *HotDep*, 2005.