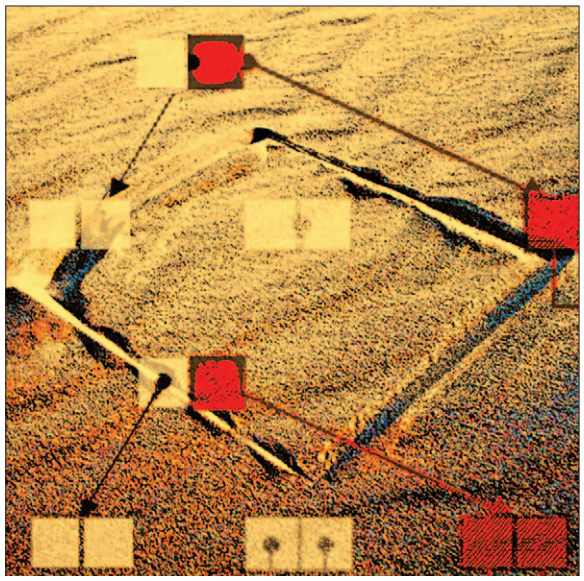
Technical Forum



WORM STORAGE IS NOT ENOUGH

The fundamental purpose of record keeping is to preserve the details associated with certain transactions or events and, furthermore, to preserve irrefutable evidence of the occurrence of such events. Trustworthy record keeping is vital to an organization in the current regulatory and business environment. It enables the smooth operation of the organization, and it helps reduce the exposure of the organization to business risks and liabilities. The problems of trustworthy record keeping are exacerbated nowadays when most records are stored in electronic form, which is susceptible to undetected destruction or modification. The requirements for proper record keeping are further underscored by current intense regulatory scrutiny, which is in part the result of recent incidents of corporate misconduct and ensuing attempts to destroy incriminating records. Consequently, several write-once-read-many (WORM) (e.g., References 1, 2, and 3) storage devices have been introduced in recent years to facilitate the effective preservation of records.

In this paper, we contend that (1) not all WORM storage devices are effective for trustworthy record keeping, (2) simply storing records in WORM storage is not sufficient to ensure that they are trustworthy, and (3) an end-to-end approach to record-keeping systems is necessary to achieve trustworthy records. Because of the high stakes that could be involved in tampering with records, the likelihood of malicious attacks, possibly by insiders, is real. The record-keeping system must, thus, be secure against such attacks. At the same time, the record-keeping system must enable the timely retrieval of all of the records that are relevant to an inquiry. Furthermore, the records must be protected from any alteration after they are retrieved from the system and before they are delivered to the inquiry agent. We use the term *Fossilization** to refer to such an end-to-end approach to designing systems for trustworthy record keeping.

TRUSTWORTHY RECORD KEEPING

When events such as financial transactions occur, records that document these events are created, and

0018-8670/07/$5.00

these records can be used as evidence that such events have occurred. A trustworthy record-keeping system must not only store the records, but also protect them from any modification. In addition, the system must be capable of efficiently locating the records associated with an inquiry and of delivering the records intact to the inquiry agent. In other words, trustworthiness must be established on an end-to-end basis.

In the normal course of business, the record creation process is an ongoing activity subjected to periodic audits in order to ensure its proper execution. Although record keeping is an important aspect of business operations, it is not meant to and cannot protect against all wrongdoing. The basic objective of record keeping is not to prevent the writing of history, but to prevent the changing of history; that is, changing the records after the fact. From this perspective, the record creation process is assumed to be trusted.

In an inquiry, all the records of interest have to be promptly located and delivered unchanged to the inquiry agent. This includes preventing any physical modification of records, including selective destruction, during their storage or their delivery to the agent. To satisfy the requirement for prompt location of relevant records when dealing with large volumes of records and quick response times for users, some form of direct access mechanism, such as an index, must be used. Protecting the records from change must, therefore, also include preventing any logical modification of the records by manipulating the access mechanism, including replacing them with new versions, performing logical deletions, and employing other forms of record hiding.

Modification and loss of records could result from hardware failures, software bugs, natural disasters, and human errors (such as issuing the wrong command or replacing the wrong disk). Given our increasing reliance on computer records, the potential gain from manipulating the records is huge. It is thus important that records be protected from attacks, possibly by company insiders. An adversary could have the highest level of access, privilege, and knowledge. Although operational procedures, such as separation of duties, can help reduce the exposure to such attacks, a well-placed adversary can still cause much damage.

In a typical scenario, the adversary might be the system administrator who is tasked by a company executive to initiate the attack when there is a threat of an audit, a legal or regulatory discovery process, or an internal investigation that could uncover damaging or embarrassing information. Although the adversary may have physical access to the records, destroying them in a blatant fashion may not be an option because it would result in the presumption of guilt. Moreover, actual destruction of records can be effectively prevented through physical security measures and by replicating the records at one or more remote sites, as is often the practice. The adversary's task, therefore, is to secretly hide or modify specific records that might contain incriminating or embarrassing information. Note that the adversary cannot clone a WORM storage device, omitting specific records in the process, without triggering suspicion because the devices tend to be large and the records are time-stamped internally.

The following threat model results: Alice, a legitimate user, creates a record (document) *R* and, through the use of an application that manipulates and stores documents, commits *R* to WORM storage. We assume the application performs as expected, and *R* reaches WORM storage. After *R* has been committed, user Mala has reasons to want *R* deleted or its contents changed; we say that Mala "begins to regret the existence of *R*." Mala will attempt to prevent user Bob, possibly representing a regulatory authority, from obtaining *R* as the answer to one of his future queries. We assume that Mala can take on the identity of any legitimate user or superuser in the system, performing any action that person can perform. For example, Mala can write any data to WORM storage as long as the write does not overwrite existing data, and she can read any data from WORM storage. This means that we cannot rely on conventional file or storage system access-control mechanisms to ensure that documents and indexes are only modifiable by legitimate applications. We assume that physical access to WORM storage is restricted or monitored so that Mala cannot steal or destroy WORM storage devices. We also assume that Mala cannot inconspicuously clone a WORM storage device and omit specific records from the clone.

FOSSILIZATION

Fossilization is the end-to-end approach for trustworthy record keeping. Fossilization consists of

three parts—storage, discovery, and delivery. The Fossilization of *storage* ensures that all records are reliably stored and securely protected from modification. The efficient identification and retrieval of all records relevant to an inquiry is known as Fossilization of *discovery*. The Fossilization of *delivery* ensures that exactly the retrieved records are delivered to the agent and that none of them are compromised along the delivery path.

Fossilization of storage

The basic requirement for preserving records is to prevent their physical deletion or modification. For electronic records, this means protecting from change the bits and bytes that make up the records. One approach is to digitally sign the contents of each record using one-way hash functions combined with cryptographic techniques such as used in notary services (e.g., Reference 4). Such an approach, however, only detects whether a record has been tampered with; it does not protect the record from being tampered with, as is required by regulatory bodies such as the Securities and Exchange Commission (SEC).

Currently, the requirement that records be immutable is satisfied by storing the records in WORM storage. Note that although immutability is often stipulated as a requirement for records, what is needed in practice is that the records be "term-immutable", that is, immutable for a specified retention period (e.g., three years). Consequently, most of the recently introduced WORM storage devices are actually "term-WORM", that is, not rewritable for a specified duration. In practice, the WORM capability can be achieved in various ways. Given the need to support term immutability, and the performance and cost advantage that rewritable (magnetic) disks currently enjoy over optical media, a popular approach is to use rewritable disks as the storage media and implement overwrite checks or content-based addressing[5] in software or firmware. In some systems, overwriting is replaced by maintaining multiple versions of a record through mechanisms such as copy-on-write (e.g., References 6 and 7). These systems can be viewed as providing WORM storage with an audit trail.

The degree of overwrite protection afforded depends on how the system is implemented. Self-securing storage[8] leverages the fact that storage servers run separate software on separate hardware to ensure that security mechanisms embedded in the file server cannot be disabled by software running on client systems. Recall, however, that Mala can take on the identity of any legitimate user or superuser in the system and can perform any action that person can perform. Thus, we need a secure mechanism for enforcing the WORM property that can not only protect against accidental alteration but also withstand malicious attacks, including attacks from the inside.

It follows that, based on the computer security principle of minimizing the trusted computing base, the component for enforcing the WORM property should be as small as possible, both to reduce the probability that something could go wrong or be compromised, and to increase our ability to verify the correctness of the component. The component should also have a simple and well-defined interface to robustly restrict traffic into the component to only legitimate requests. In addition, we must make sure that the component mediates all requests; in other words, that the overwrite protection cannot be circumvented by, for example, directly accessing the rewritable disk. One such system is the Content Immutable Storage described in Reference 9.

A record is typically composed of data objects such as a collection of documents or a sequence of data blocks. As illustrated in ***Figure 1***, a record involves not only the data objects but also additional objects that are used to identify and manage the record and that are known as metadata. Storing the data objects in secure WORM storage is analogous to "casting the data objects in stone" (once written the data objects cannot be modified). Keeping the metadata in rewritable storage is akin to "writing them in sand" (they can be changed); alteration of records is possible as shown in Figure 1. Specifically, a record with data objects 1, 2, 3, and 4 can be made to appear as consisting of data objects 1, 2, 3′, and 4. In general, a record must be preserved in WORM storage together with all of its associated evidentiary metadata, including information that describes the structure and attributes of the record, such as its creation date. The Fossilization of storage ensures that no part of a record can be altered, replaced, or removed and that the record is preserved completely with all of the information necessary to ensure its long-term usefulness.

Fossilization of discovery

Formerly, Fossilization of storage, that is protecting the record from physical modification, was sufficient

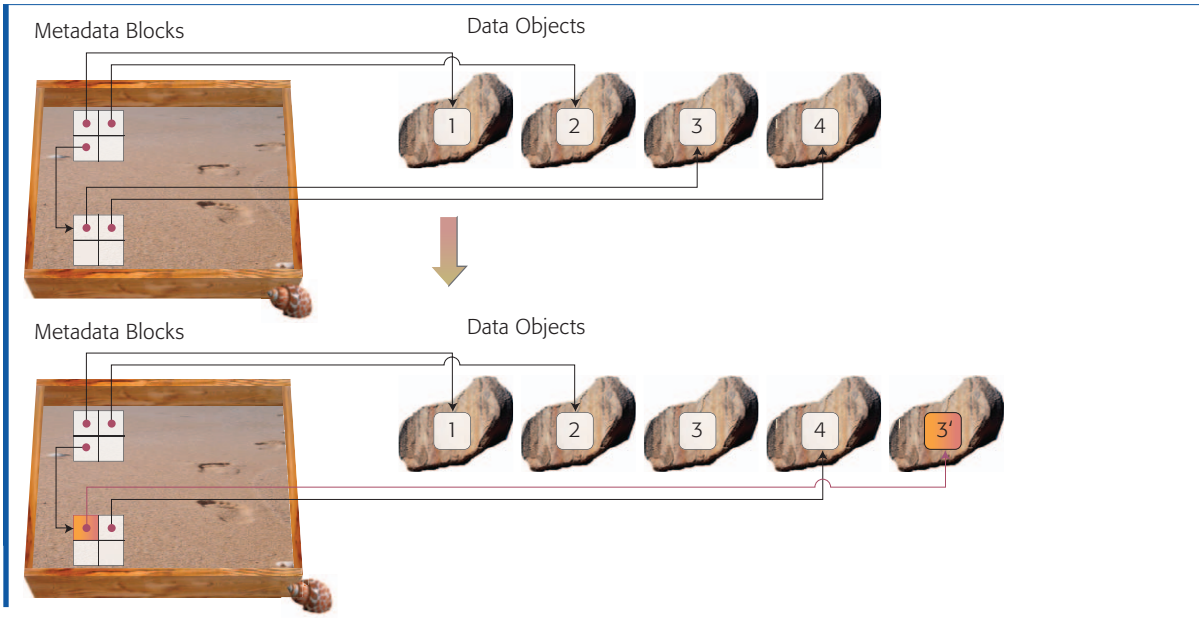


Figure 1
Possible alteration of records when metadata are kept in rewritable storage

to ensure that the records were effectively immutable because even if a record were to be logically modified, all versions of the record would be preserved so that one could theoretically scan all the stored data to discover the original record. For example, if we store a file in an optical disc, we can later write a newer version of the file and have it logically replace the first file. If we scan through the entire contents of the optical disc, we can find the first file; but scanning all of the data to piece together possible versions of a record has become far less practical, given huge volumes of records, increasingly stringent query response times, and the growing complexity of record-keeping systems. For the same reasons, creating a direct access mechanism such as an index at query time is not a viable option. Moreover, the trustworthiness of an index created at query time is suspect because, according to our threat model, an adversary who is aware of the investigation could prevent the index from including the record *R*.

Consequently, some form of index must be maintained for accessing the records. But unless the index is properly realized, the records stored in WORM storage can in effect be hidden or altered. *Figure 2* illustrates the way a record stored in WORM storage can be logically modified if the index through which it is accessed is susceptible to improper manipulation by an adversary. Figure 2A shows how the manipulation of the index could cause a logical deletion of record B. Figure 2B shows how the index could be manipulated to modify record B to record B′. To prevent such logical modification of the records, the index must have the property that once a record is preserved in WORM storage, it is accessible through the index. For example, both the index entry for that record and the path through the index to that entry could be made immutable.

In our threat model, an adversary, Mala, may try to keep the record *R* out of the index that agent Bob uses in his search. She can do this by preventing *R* from ever getting into the index or by ensuring that *R* is not in the index that Bob uses. We must, therefore, make sure that (1) *R* is entered in the index before Mala regrets *R*'s existence, and (2) any record that ever enters the index stays accessible through it forever (or at least for a mandated retention period). To stop Mala from preventing *R* from entering the index, one approach is to insert *R* and construct the index entry for *R* as a single action, because we trust the document insertion code to get *R* into WORM storage initially. In general, inserting a document (e.g., e-mail) into a

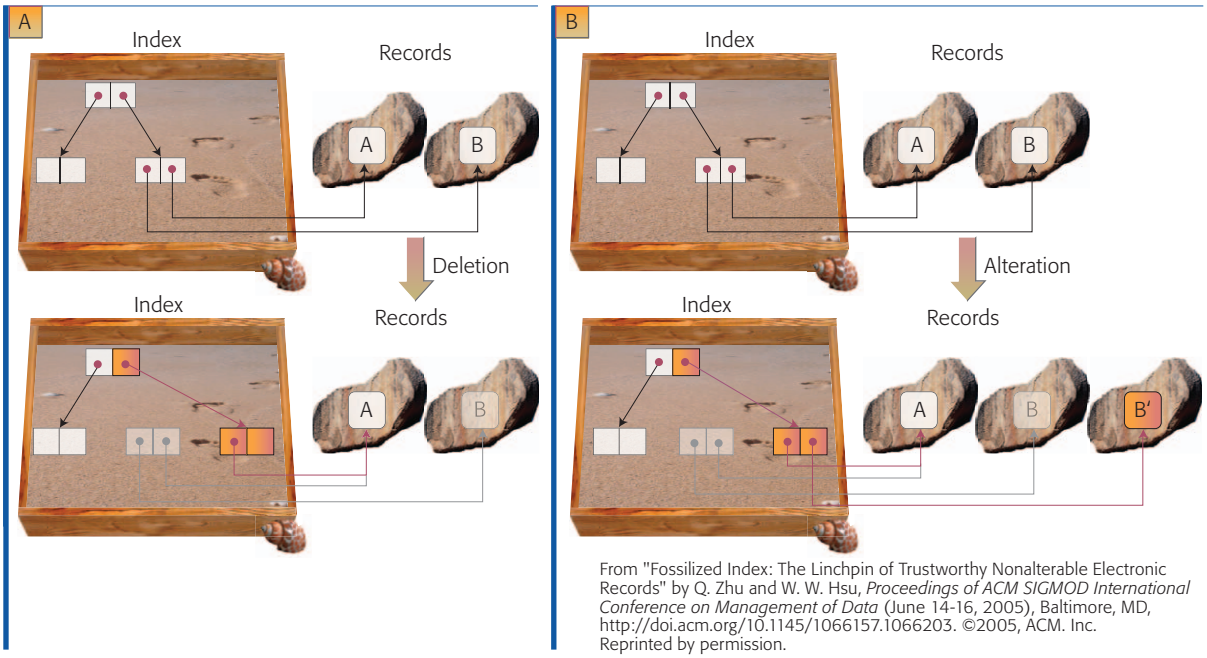


Figure 2
Possible logical modification of records if index is stored in rewritable storage: (A) deletion and (B) alteration

full-text index could involve many random I/Os[10] and would be prohibitively expensive to perform incrementally in an online fashion. However, recent techniques, such as that introduced in Reference 11, enable online update of an inverted index while maintaining good query performance.

Mala may also attempt to logically modify *R* by manipulating the index. Research work into indexing techniques has been extensive, but simply storing conventional indexes such as the B-tree, extensible hashing, and their derivatives in WORM storage (e.g., Reference 12) does not achieve Fossilization of discovery.[11,13] In general, index trees that grow from the leaves up to the root are vulnerable to compromise because an adversary could modify records at will by exploiting the provision for creating new versions of the tree nodes. Any approach that requires the rebalancing of a tree is similarly exposed because it inevitably allows an adversary to create new paths to records. Methods that permit index entries to be relocated are also not trustworthy because they open the door for an adversary to create new versions of any entry. In short, new indexing techniques are necessary to ensure that the index cannot be suitably manipulated to hide the index entry of a committed record. Recent work has introduced such indexing techniques and shown that they do not incur much additional cost.[11,13]

Note that Fossilization of discovery must be applied to any trusted means of finding and accessing a record. Examples include the file-system directory that allows records to be located by the file name, the database index that enables records to be retrieved based on the value of some specified field or combination of fields, and the full-text index that allows records containing some particular words or phrases to be found.

Fossilization of delivery

The Fossilization of storage and discovery ensures that all records are securely preserved and that any record can be quickly located, but, a weak link in the system remains—the records may be susceptible to alteration during their *delivery* to the agent conducting the inquiry. For example, adversary Mala could fabricate a set up with a version of the search engine or the system stack that has been doctored to compromise the results to agent Bob's queries.

To illustrate this, ***Figure 3*** shows a typical record-keeping system in which the records retrieved from WORM storage are handled by an elaborate stack of

software components before they are delivered to the agent. The system stack is designed in modular fashion in order to allow the addition of new components or the replacement of existing components. In fact, the components often have to be updated for function or to fix bugs and security exposures. The dynamic nature of the system stack makes the system vulnerable to security attacks. For example, as illustrated in Figure 3, an additional component or software patch labeled Filter could be installed by Mala and used to modify or drop selected records in transit. Such a system is clearly not trustworthy.

Therefore, the inquiry agent should not assume that record delivery is trustworthy. Agent Bob should, for example, be able to verify that he is running a certified version (e.g., signed by a trusted party) of the search engine and operating system. In some cases, it may be necessary for Bob to use his own trusted software stack or version of the software stack to directly access the records. Alternatively, the layout of the records and indexes in the WORM storage could be disclosed so that Bob could use his own code or some third-party software, possibly open-source software, to interpret the layout and retrieve the records.

In practice, Bob may not personally or directly conduct the record discovery. Instead, the search results are typically delivered to Bob through some physical mechanism (depicted in the figure as a squiggle arrow), such as a CD (compact disc). In such cases, Mala (depicted in the figure as a bandit) could target the physical delivery mechanism. For example, she could create a near duplicate copy of the search results CD that omits the record *R* and send that to Bob. We must prevent Mala from compromising the integrity of any record sent to Bob, and even more important, we must protect the completeness of the search results; that is, we must ensure that Mala cannot omit any record from the result set without Bob's knowledge.

Techniques for ensuring data integrity, such as digital signatures, have been well studied. Ensuring the completeness of search queries is a more challenging problem and has been addressed in part for the data publishing model (e.g., References 14 and 15). In this model, the completeness of query results can be certified by having the data owner attach signatures to the data/indexes, which can later be verified by the inquiry agent. With our current threat model, however, because adversary Mala can take on the data owner's identity, she can modify the data/indexes and re-sign them. More research is needed, therefore, toward the design of a trustworthy delivery mechanism. One approach is to use the WORM storage as a trusted entity to certify that the result set received by Bob is proper.[16]

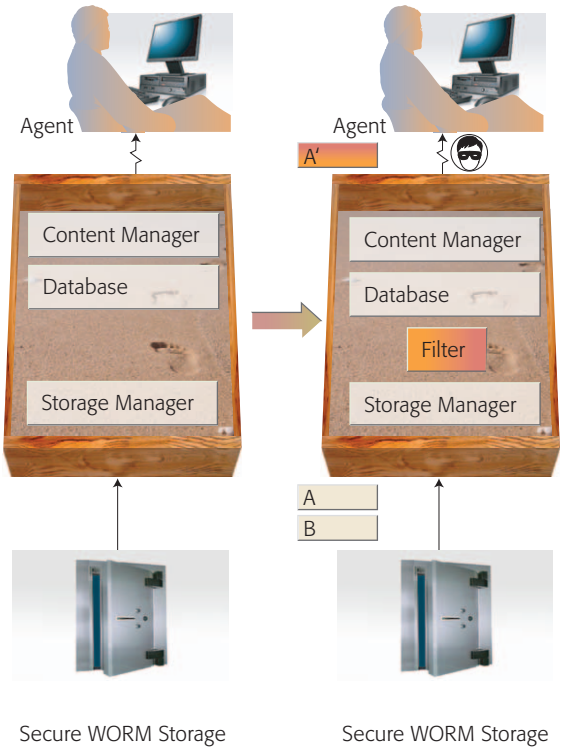


Figure 3
Possible alteration of records on delivery path from WORM storage to agent

Summary

The fundamental purpose of record keeping is to preserve accurate details of events and establish solid proof that the events have occurred. Trustworthy records are, therefore, those that can be relied upon to provide irrefutable evidence of all the events that have been logged. Such records must be managed from an end-to-end perspective, beginning with their preservation, and including their subsequent discovery and delivery to an agent seeking proof or details of an event.

The current limited focus on storing electronic records in WORM storage is increasingly inadequate

to ensure that such records are trustworthy. In particular, given the high stakes that could be involved in tampering with the records, the WORM storage must be secure against attacks, even those from the inside. Furthermore, the WORM storage must be used in such a way as to allow all records that are relevant to an inquiry to be quickly found without being vulnerable to logical modification. In addition, these records must be protected against any alteration after they are retrieved from the WORM storage and before they are delivered to the inquiry agent.

We use the term *Fossilization* to describe a holistic approach to storing and managing records that ensures the records are trustworthy. Fossilization is composed of three parts. The first, Fossilization of storage, ensures that all the records and their associated metadata are securely protected from any loss or modification. The second, Fossilization of discovery, ensures that every preserved record which is pertinent to an inquiry can be readily located and retrieved in a timely fashion, and the third, Fossilization of delivery, warrants that exactly the records retrieved from storage are delivered to the agent and that the records are delivered unaltered.

CITED REFERENCES

1. EMC Corp., "EMC Centera Governance Edition: Content Addressed Storage System" (2003), http://www.emc.com/products/systems/centera_ce.jsp.
2. Network Appliance, Inc., "SnapLock™ Compliance and SnapLock Enterprise Software" (2003), http://www.netapp.com/products/enterprise-software/data-retention-software/archive-compliance/snaplock.html.
3. IBM Corp., "IBM System Storage DR550" (2004), http://www-1.ibm.com/servers/storage/disk/dr.
4. J. M. Peha, "Electronic Commerce with Verifiable Audit Trails," *Proceedings of the Internet Society Conference INET '99* (June 1999).
5. S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage," *Proceedings of the FAST '02 Conference on File and Storage Technologies* (January 2002), pp. 89–101.
6. D. S. Santry, M. J. Fecley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, and J. Ofir, "Deciding When to Forget in the Elephant File System," *Proceedings of the 17th ACM Symposium on Operating System Principles* (December 1999), pp. 110–123.
7. Z. N. Peterson and R. C. Burns, "Ext3cow: A Time-Shifting File System for Regulatory Compliance," *ACM Transactions on Storage* **1**, pp. 190–212 (May 2005).
8. J. D. Strunk, G. R. Goodson, M. L. Scheinholtz, C. A. N. Soules, and G. R. Ganger, "Self-Securing Storage: Protecting Data in Compromised Systems," *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI 2000)* (October 2000), pp. 165–180.
9. L. Huang, W. W. Hsu, and F. Zheng, "Content Immutable Storage for Trustworthy Electronic Record Keeping," *Proceedings of the Conference on Mass Storage Systems and Technologies* (May 2006), pp. 101–112.
10. A. Tomasic, H. García-Molina, and K. Shoens, "Incremental Updates of Inverted Lists for Text Document Retrieval," *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data* (September 1994), pp. 289–300.
11. S. Mitra, W. W. Hsu, and M. Winslett, "Trustworthy Keyword Search for Regulatory-Compliant Records Retention," *Proceedings of the 32nd International Conference on Very Large Data Bases* (September 2006), pp. 1001–1012.
12. M. C. Easton, "Key-Sequence Data Sets on Indelible Storage," *IBM Journal of Research and Development* **30**, No. 3, 230–241 (May 1986).
13. Q. Zhu and W. W. Hsu, "Fossilized Index: The Linchpin of Trustworthy Non-alterable Electronic Records," *Proceedings of the ACM SIGMOD International Conference on Management of Data* (June 2005), pp. 395–406.
14. M. Goodrich, R. Tamassia, and A. Schwerin, "Implementation of an Authenticated Dictionary with Skip Lists and Commutative Hashing," *Proceedings of the DARPA Information Survivability Conference and Exposition* (June 2001), pp. 1068–1082.
15. R. Sion, "Query Execution Assurance for Outsourced Databases," *Proceedings of the 31st International Conference on Very Large Data Bases*, (September 2005), pp. 601–612.
16. S. Mitra, M. Winslett, X. Ma, and W. W. Hsu, "Trustworthy Migration and Retrieval of Regulatory-Compliant Records." Submitted for publication.

Accepted for publication December 1, 2006.
Published online April 11, 2007.

Windsor W. Hsu
IBM Almaden Research Center,
Computer Science Storage Systems Department,
San Jose, California

Shauchi Ong
IBM Almaden Research Center,
Computer Science Storage Systems Department,
San Jose, California