ACE: a Novel Software Platform to Ensure the Long Term Integrity of Digital Archives

Sangchul Song, Joseph JaJa; Institute for Advanced Computer Studies, Department of Electrical and Computer Engineering, University of Maryland, College Park, Maryland, USA

Abstract

We develop a new methodology to address the integrity of long term archives using rigorous cryptographic techniques. A prototype system called ACE (Auditing Control Environment) was designed and developed based on this methodology. ACE creates a small-size integrity token for each digital object and some cryptographic summary information based on all the objects handled within a dynamic time period. ACE continuously audits the contents of the various objects according to the policy set by the archive, and provides mechanisms for an independent third-party auditor to certify the integrity of any object. In fact, our approach will allow an independent auditor to verify the integrity of every version of an archived digital object as well as link the current version to the original form of the object when it was ingested into the archive. We show that ACE is very cost effective and scalable while making no assumptions about the archive architecture. We include in this paper some preliminary results on the validation and performance of ACE on a large image collection

Introduction

One of the most challenging problems facing digital archives is how to ensure the authenticity of their holdings over the long term (tens or hundreds of years). Unless the authenticity of an archive can be assured, it may not be possible to use the archive's holdings to support any significant endeavor. Digital information is in general quite fragile, especially over time. Errors can be introduced because of hardware and media degradation, hardware and software malfunction, operational errors, security breaches, and malicious alterations, to name a few of the obvious ones. Other potential sources of errors, which are particularly relevant for long term archives, include major hardware and software systems changes due to technology evolution, and the possibility of major natural hazards and disasters such as fires, floods, and hurricanes. Two additional factors complicate this problem further. First, an object may be subjected to a number of transformations during its lifetime, including those migrative transformations due to format obsolescence. These transformations may alter the object in unintended ways. Second, most current integrity checking mechanisms are based on some type of cryptographic techniques, most of which are likely to become less immune to potential attacks over time and hence they will need to be replaced by stronger techniques. Therefore any approach to ensure the authenticity of a long term archive has to also be able to address these two problems as well.

Several technical approaches have been proposed to address the long term integrity of digital archives, including those that appeared in [1], [3], [4], [6], [7], [8], and [9], but none seems to offer a solid approach that is applicable to the different emerging architectures for digital archives (including centralized and distributed archives) and that is capable to continually monitor and verify the integrity of the data in a cost effective way.

In this paper, we introduce a general software environment called ACE (Auditing Control Environment), which is based on a rigorous cryptographic approach and yet quite efficient and can interoperate with any archiving architecture. Using the new framework, we introduce procedures to continually verify the integrity of the archive. Our approach will allow an independent auditor to verify the integrity of every version of an archived digital object as well as link the current version to the original form of the object when it was ingested into the archive.

Specifically, ACE is based on creating a small-size integrity token for each digital object upon its deposit into the archive (or upon registration of the object of an existing archive), to be stored either with the object itself or in a registry at the archive as authenticity metadata. Cryptographic summary information that depends on all the objects registered during a dynamic time period is stored and managed separately. The summary information is very compact and is of size independent of the number or sizes of the objects ingested. Regular audits will be continuously conducted, which will make use of the integrity tokens and the summary integrity information to ensure the integrity of both the objects and the integrity information. In our prototype, audits can also be triggered by an archive manager or by a user upon data access. However we are assuming that the auditing services are not allowed to change the content of the archive even if errors are detected. The responsibility for correcting errors is left to the archive administrator after being alerted by the auditing service.

Overview of the ACE Approach

ACE adopts a two-tier approach. The first tier deals with creating a small size Integrity Token (IT) (Figure 1) for each digital object upon its deposit into the archive (or upon registration of the object of an existing archive), to be stored either with the object itself or in a registry at the archive as authenticity metadata. Cryptographic Summary Information (CSI) depending on all the objects registered during a dynamically adjustable time interval is stored and managed independently of and separately of the archive. The ITs and CSIs are used to continually verify the authenticity of the corresponding digital object. The second tier involves the generation of very compact witness values that cryptographically depend on all the objects ingested during a relatively long time period (such as a week).

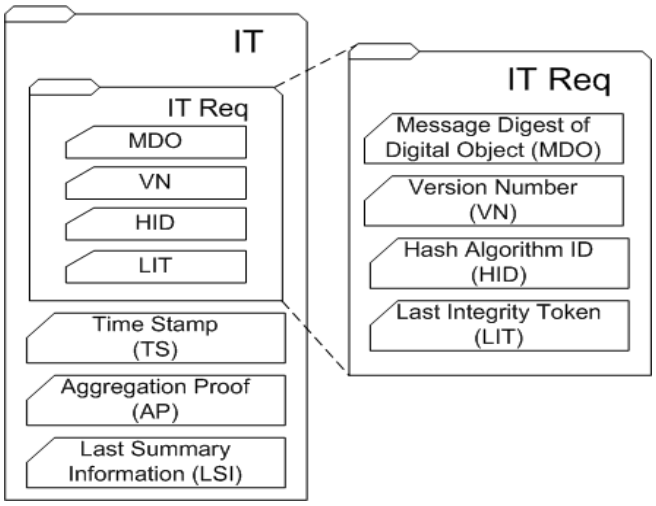

Figure 1. *Integrity Token*

ACE: First Tier

The first tier integrity information types (IT and CSI) are generated in two steps; aggregative registration and hash-linking. The aggregative registration of the objects is typically invoked during ingestion, and composed of aggregation rounds. The interval of each round is determined dynamically based on the number of registration requests and time passed. This dynamic aggregation period allows us to control both the maximum size of ITs and maximum wait-time for registration. During an aggregation round, the hashes of all the objects submitted for registration as well as random hashes as necessary are aggregated using an authentication tree such as the Merkle's tree [10]. Note that, in practice, the hash of the object is submitted as a part of the registration request (IT Req in Figure 1). The internal node in the authentication tree has the hash value of the concatenated hashes at the children. We insert random hash values into each round to ensure that the tree will always have a certain minimal number of leaves. Figure 2 shows an authentication tree for a round involving eight objects with hash values $h_0, h_1, h_2, ..., h_7$.

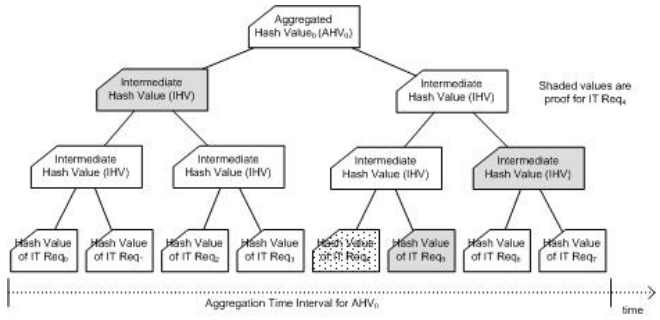

Figure 2*. Authentication Tree (IT Req$_i$ contains h_i)*

Note that the value at the root is a hash value that depends in a cryptographic sense on all the objects processed during a round. For each object, we assemble a short list of hashes from the tree, called an aggregation proof, to enable the derivation of the root value from the hash of the object. We time stamp all the objects participating in each round with the same time stamp.

The second step consists of linking the hash value generated at each round with the hash values generated at the previous rounds using a structure that depends on the linking scheme used. In our prototype, we use a simple binary linking scheme that computes the hash value of the previous Cryptographic Summary Information (CSI) concatenated with the hash value of the current round as illustrated in Figure 3. This is the same scheme as suggested in [5] and [24]. In this binary linking scheme, the only two data necessary to construct CSI is the previous CSI and the root value of the authentication tree. The former is included in IT (LSI in Figure 1), whereas the latter can be re-computed using the aggregation proof. In the other words, IT has all the information to re-compute the corresponding CSI at any time.

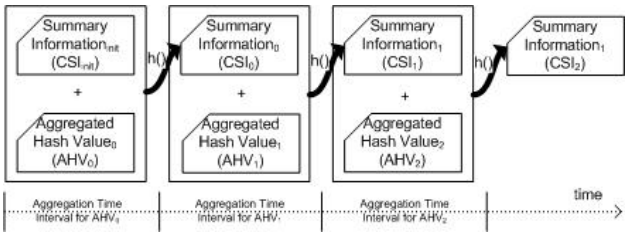

Figure 3. *CSI Chain*

ACE: Second Tier

As mentioned before, the second tier deals with generating witness values that will ensure the integrity of CSIs which are generated from the first tier operations. A witness value is constructed by aggregating the CSIs that have been created over each week, using an authentication tree whose root value becomes the witness value of the week. These witness values are published over the Internet at well-known public sites offering storage, library, or publication services. Since these witness values are small in size (1KB~2KB a year), we also store them on a CD ROM (in fact, on multiple CD-ROMs that are refreshed on a regular basis). ACE currently uses the Internet newsgroups at Google, Yahoo, and MSN as publication services.

ACE Procedures

Registration

Objects are typically registered with ACE during the ingestion process. A registration request involves creating a registration token (IT Req in Figure 1), and submitting the token to the ACE server. The ACE server generates the IT and the CSI, and returns IT to the requester. The returned IT is locally stored with proximity to the object itself, whereas CSI is managed internally within the ACE server.

Verification and Auditing

ACE provides two types of integrity auditing, the first involves a process running on a moderately secure server external to the archive, which verifies the integrity of the archive's content in a periodic, regular fashion; the second involves an auditing process triggered by an archivist or by a user upon data access. A simple way to verify the integrity of an object involves the following steps

Step 1. Compute the hash of the given object and compare it to the hash stored in the object's token. Proceed to the next step if there is an agreement; otherwise, either the object or its token has been modified. In the latter case, we can distinguish between the two possibilities by proceeding to the next step as well

Step 2. Use the computed hash value in combination with the proof in the token to determine the CSI of the round during which the object was injected into the system. An agreement indicates that the object is intact (correct the hash value stored in the token if necessary).

The decision on the integrity of the given object is made only if there is an agreement in Step 2. In case of a disagreement, we report that the object may be corrupt, and leave it up to the archive manager to decide whether or not to verify the integrity of the corresponding CSI value or compare the corresponding object with another copy in the archive.

Considering that it is more likely to have a corrupted object than a corrupted integrity token (integrity tokens do not have to be made publicly available), in a practical implementation, one can choose to have two separate processes – one that performs only Step 1, and the other performs Step 2. That is, the object integrity and the token integrity are verified separately and independently according to the policies set by the archive and the integrity management system. The two can be linked together whenever necessary using the above procedure.

Witness Publication

The ACE server constantly runs a process that constructs and publishes witnesses on a weekly basis. The fact that these witnesses are cryptographically dependent on CSIs maintained within the server, along with the assumption that the published witnesses are almost impossible to tamper with, allows us to ensure the integrity of CSIs through the witness validation procedure discussed below.

Witness Validation

In this internal procedure, the integrity of each CSI is checked using witnesses previously published. In particular, for each witness value W and for each CSI value computed within that week, we check to see if the proof attached to the CSI value yields W. In the affirmative, the CSI value is correct. Otherwise, it is not.

In the case when the object or the CSI is determined to be incorrect, our verification service notifies the archive manager about the faulty object, and it is left up to the archive to take the appropriate action. We believe that an integrity verification service should not be allowed to modify anything in the archive. Its main function is to continually monitor and verify the authenticity of the data. In our experimental setting, we use the distributed persistent archive pilot system between the National Archives, the University of Maryland, and the San Diego Supercomputer Center, which provides at least three replicas for each digital object based on the federated SRB (Storage Resource Broker) grid technology [2], and hence a copy can be corrected using a voting scheme over the distributed archive. As for correcting erroneous CSI values, our integrity checking prototype makes use of a three-way mirrored registry of the CSI values, each of which is audited independently. Hence the faulty CSI can be corrected using a correct replica from the other registries. Note that the size of a registry grows in the order of a few gigabytes per year (independent of the size of the archive), and that the registry is not publicly accessible. Therefore maintaining the integrity of the CSI registry can be done in a cost effective way.

Updating Integrity Information

Updating IT becomes necessary either when a new stronger hash function replaces the current hash function, or an object is updated to a new version due to, for example, a format transformation. In order to tackle the former case, there is a well-known solution to deal with renewing the integrity information by re-registering each related object with the old integrity token attached to it (see for example [4]). Such a solution will ensure our ability to verify the integrity of the object since its ingestion into the archive as articulated in earlier work. This process increases the size of the integrity token, but has no impact on the sizes of the other integrity components. In the case when an object is subjected to a transformation, we include the version number in addition to the hash of the object and re-register the object (VN in Figure 1). Different versions can be linked through the global ID of the object using a dark archive, and hence it is possible to verify the integrity of all the versions of each object starting with the current one and ending with the first version ingested into the archive. Note that the integrity of an object should be verified before it is transformed into a new format to ensure its authenticity at this time of its history.

ACE Prototype

The ACE prototype includes two major components: ACE Integrity Management System (ACE-IMS) and ACE Audit Manager (ACE-AM). The ACE-IMS is a server that issues integrity tokens, preserves the CSI values, and computes and publishes the witness values. The ACE-AM is a bridging component between the archive and the ACE-IMS, which is local to each archiving node. In a distributed setting, the audit managers work asynchronously independent of each other, and hence copies of the same object will be audited independently of each other.

The ACE-IMS, operating separately from the archive, provides two important services: integrity token issuing and CSI verification. The former service generates an integrity token upon a request from the archive. Using the digital object and the integrity token, the archive can at anytime construct the cryptographic summary corresponding to the round in which the digital object was registered. The CSI values will be maintained separately and independently by the ACE-IMS.

In a typical archiving environment, the integrity tokens can be stored either with the object itself or in a separate registry dedicated to authenticity metadata. In our prototype, we use a separate database to hold the integrity tokens.

The ACE Audit Manager (ACE-AM) is local to an archiving node whose main function is to pass information between the archiving node and the ACE-IMS. In particular, the ACE-AM selects a digital object to be audited, either based on the local periodic auditing policy of the archiving node or upon request from an archive manager or a user. It then retrieves the digital object's integrity token, computes the hash of the object, and sends this information to the ACE-IMS.

Figure 4 shows the overall ACE architecture assuming a distributed archiving infrastructure. A centralized archiving infrastructure will reduce to a single archiving node. The upper section represents the archive, the middle section contains the ACE-AM that is local to each archiving node, and the lower section represents the ACE-IMS, which is completely outside the archive.

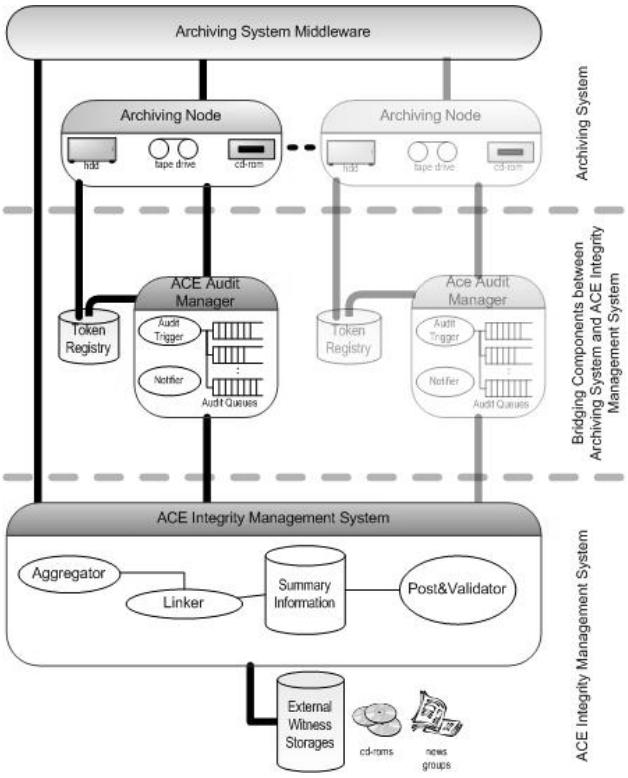

Figure 4. *ACE System Architecture*

ACE Preliminary Performance Test

We have tested and evaluated ACE on a number of collections, the largest of which is the NARA EAP (Electronic Access Project) Image Collection consisting of over 1.1TB of 126,548 files. For the latter collection, we were able to fully audit the 126,548 objects in about 15 hours while storing the data remotely on a separate server. Most of the time was spent in moving the data between the separate machines. We expect performance to be much better in a production environment since all the data movement will be carried out locally between the audit manager and the local storage.

Conclusion

In this work, we have designed and implemented a platform-independent system, called ACE, which addresses the integrity of long-term archives. Although our approach is based on rigorous cryptographic techniques, the computational requirements are minimal. ACE is policy-driven and is able to continuously audit the data and its integrity information. Moreover, ACE provides rigorous methods for an independent auditor to certify the integrity of any particular object of the archive.

Acknowledgments

This work was supported in part by the National Science Foundation and the Library of Congress, contract IIS-0455995, under the DIGARCH program.

References

[1] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. DOS-resistant authentication with client puzzles. In Bruce Christianson, Bruno Crispo, and Mike Roe, editors, Proceedings of the 8th International Workshop on Security Protocols, to appear in the Lecture Notes in Computer Science series, Cambridge, UK, April 2000. Springe

[2] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC Storage Resource Broker. In Procs. of CASCON'98, Toronto, Canada, 1998.

[3] O. P. Damani, P. Y. Chung, Y. Huang, C. Kintala, and Y. M. Wang, "ONE-IP: Techniques for hosting a service on a cluster of machines," in Proc. the Sixth Int. World Wide Web Conference, April 1997.

[4] Stuart Haber and Pandurang Kamat. "Content Integrity Service for Long-Term Digital Archives." In Proceedings of Archiving 2006, May 2006, pp 159-164.

[5] Stuart Haber and W. Scott Stornella, "How to time-stamp a digital document," Journal of Cryptology, 1991.

[6] Ronald Jantz and Michael J. Giarlo. "Digital Preservation – Architecture and Technology for Trusted Digital Repositories Reich." D-Lib Magazine, 7(6), June 2005. <http://www.dlib.org/dlib/june05/jantz/06jantz.html>

[7] Lisa Kelly. "British Library secures integrity of digital archive." Computing. 25 Apr 25 2006. <http://www.computing.co.uk/computing/news/2154704/british-li>

[8] T. T. Kwan, R. E. McGrath, and D. A. Reed, "NCSA's World Wide Web Server: Design and Performance", IEEE Computer, pp. 68-74, Nov. 1995

[9] Petros Maniatis, TJ Guili, David S. H. Rosenthal and Mary Baker. "THE LOCKSS Peer-to-peer Digital Preservation System." ACM TOCS, 23(1), February 2005.

[10] Ralph Merkle. "Protocols for public key cryptosystems," In Proceedings of the 1980 Symposium on Security and Privacy, IEEE Computer Society Press, 1980, pp 122–133.

Author Biography

Sangchul Song is a doctoral student in Electrical and Computer Engineering at University of Maryland, College Park, MD. Before joining Maryland, he worked as a security software engineer for several years in San Jose, CA. He received BE and MS degree at Korea University, Seoul, Korea. At Maryland, he has been actively involved in the long term digital preservation group led by Prof. Joseph JaJa.

Joseph JaJa currently holds the position of Professor of Electrical and Computer Engineering with a joint appointment at the Institute for Advanced Computer Studies at the University of Maryland, College Park. Dr. JaJa received his Ph.D. degree in Applied Mathematics from Harvard University and has since published extensively in a number of areas including parallel and distributed computing, combinatorial optimization, algebraic complexity, VLSI architectures, and data-intensive computing. His current research interests are in parallel algorithms, digital preservation, and scientific visualization of large scale data. Dr. JaJa has received numerous awards including the IEEE Fellow Award in 1996, the 1997 R&D Award for the development software for tuning parallel programs, and the ACM Fellow Award in 2000. He served on several editorial boards, and is currently serving as a subject area editor for the Journal of Parallel and Distributed Computing and as an editor for the International Journal of Foundations of Computer Science.