RAGIB HASAN, RADU SION, AND
MARIANNE WINSLETT

secure provenance: protecting the genealogy of bits

THE ABILITY TO TRACK THE ORIGIN OF information is essential in science, medicine, commerce, and government. Applications such as digital rights protection, DNA testing, drug trials, corporate financial accounting, and national intelligence need to guarantee the integrity and authenticity of information as it flows between people and tasks. In this article, we describe what digital provenance means and how to provide strong integrity and confidentiality assurances for data provenance information. We present our provenance-aware system prototype that implements provenance tracking of data writes at the application layer, which makes it extremely easy to deploy. The prototype is efficient: for typical real-life workloads, its runtime overhead does not exceed 13% and is below 3% for most workloads.

Ragib Hasan is a PhD candidate in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His research interests include secure provenance, regulatory compliant storage systems/databases, and other aspects of storage security.

rhasan@cs.uiuc.edu

Radu Sion is an assistant professor in the Department of Computer Science at Stony Brook University. His research interests span information assurance, practical cryptography, and large data and compute-intensive systems.

sion@cs.stonybrook.edu

Marianne Winslett is a research professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. Her research interests lie in information security and in the management of scientific data. She is an ACM Fellow, a former vice-chair of ACM SIGMOD, and recipient of the Presidential Young Investigator Award from NSF.

winslett@cs.uiuc.edu

Provenance

In 2006, the Picasso painting *Dora Maar au Chat* (Dora Maar with Cat) was auctioned at Sotheby's for US $95 million, becoming one of the most expensive paintings in the world. At the same time, someone listed several paintings for sale on eBay, supposedly complete with Picasso's signature. Although eBay quickly removed those listings, many purported Picassos are still on the market.

How do art buyers authenticate paintings? Many factors play a role, but one key element is provenance records that list the ownership history of an item and the actions performed on it. Provenance is widely used in the arts, archaeology, science, genealogy, and data archives, where it has been called the *fundamental principle of archiving.*

Provenance has traditionally been used to authenticate physical objects, but life today has become increasingly dependent on digital information that originated elsewhere, was processed by other people, and was stored in potentially untrustworthy storage. In such situations, it is increasingly important to know where the information comes from and how it has been processed and handled. In other words, to be able to trust a piece of information, we need to know its provenance.

Provenance is a highly overloaded term, but all definitions share the same core concepts: data provenance is a description of the origins, lineage, derivation, and transmission history of a digital object. Until now, scientists have been the primary users of data provenance systems, and provenance research has mainly focused on the tasks of modeling, representation, collection, annotation, and querying.

However, as provenance steps into mainstream computing, new challenges arise. With its increased use in financial, medical, and other non-scientific application areas, provenance information faces a host of security threats, including active attacks from adversaries. In high-stakes business and medical applications, insiders may have significant incentives to alter data records' history. For example, in big finance, regulatory and legal considerations mandate provenance assurances, and the US Sarbanes-Oxley Act sets prison terms for officers of companies that issue incorrect financial statements. As a result, officers have become very interested in tracking and securing the path that a financial report takes during its development, including both input data origins and authors. The US Gramm-Leach-Bliley Act, Securities and Exchange Commission rule 17a, and HIPAA also require documentation and audit trails for financial or medical records.

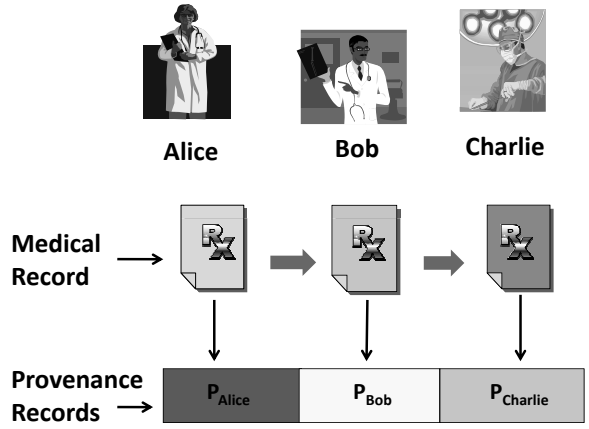


FIGURE 1: EXAMPLE SCENARIO. DANA GOES TO DR. ALICE, WHO REFERS HER TO DR. BOB FOR A TEST, AND THE TEST RESULTS ARE THEN PROCESSED BY DR. CHARLIE. BECAUSE OF A MISDIAGNOSIS BY DR. BOB, DANA SUFFERS HEALTH PROBLEMS AND SUES DR. CHARLIE, WHO WANTS TO USE THE PROVENANCE ($P_{ALICE}|P_{BOB}|P_{CHARLIE}$) OF DANA'S MEDICAL RECORDS TO PROVE HIS INNOCENCE. DR. BOB WANTS TO HIDE HIS ACTIONS BY RETROACTIVELY ALTERING HIS ENTRIES IN DANA'S MEDICAL RECORDS AND TAMPERING WITH THE PROVENANCE RECORD OF HIS DIAGNOSIS (P_{BOB}) TO MATCH.

When information crosses application and organizational boundaries and passes through untrusted environments, its associated provenance information is vulnerable to illicit alteration. For example, in Figure 1, Dr. Bob, wanting to hide evidence of his misdiagnosis, retroactively changes the diagnosis in his patient's record and tampers with the associated provenance record. Access control is insufficient to prevent this tampering, as Dr. Bob may have physical control over a machine where the information resides. Thus, the trustworthiness of the provenance records themselves is in question: we need *provenance of provenance*, i.e., a model for secure provenance.

Making provenance records trustworthy is challenging. Ideally, we need to guarantee *completeness*—all relevant actions pertaining to a piece of information are captured; *integrity*—adversaries cannot forge or alter provenance

records; *availability*—auditors can verify the integrity of provenance information; *confidentiality*—only authorized parties can read provenance records; and *efficiency*—provenance mechanisms should have low overheads.

Here, we take the first step towards preventing forgery of history as stored in provenance records. We present a scheme for providing integrity and confidentiality assurances for provenance records, and we describe a proof-of-concept implementation for file systems that imposes only 1% to 13% overhead for typical real-life workloads.

A Model for Provenance

In what follows we use the term *document* to refer to the data item for which provenance information is collected, such as a file, database tuple, or network packet. At the IT layer, the *provenance* of a document is the record of actions taken on that document over its lifetime. Each access to a document D may generate a *provenance record* P. The types of access that should generate a provenance record depend on the domain, as do the exact contents of the record, but in general P may include the identity of the accessing principal; a log of the access actions (e.g., read, write) and their associated data (e.g., the bytes of the document or its metadata that were read/written); a description of the environment at the time of the action, such as the time of day and the software environment; and confidentiality- and integrity-related components, such as cryptographic signatures, checksums, and keying material. A *provenance chain* for document D is a non-empty time-ordered sequence of provenance records $P_1 \mid \cdots \mid P_n$.

In a given security domain (organization), *users* are principals who read and write documents and their metadata. Each organization has one or more *auditors*, who are principals authorized to access and verify the integrity of provenance records associated with documents. Documents move from one user to another, as email attachments, FTP transfers, or by other means. Provenance chains move with the documents. When a user modifies a document, a new provenance record describing the modifications is appended to the provenance chain and the user permits some subset of the auditors to read the new record. *Adversaries* are principals from inside or outside an organization who have access to a document and its provenance chain and who want to alter them inappropriately, as discussed below.

We cannot track provenance perfectly, because a provenance tracking system implemented at a particular level of the system is oblivious to attacks that take place outside the view of that level. For example, suppose that we implement provenance tracking in the OS kernel. If the kernel is not running on hardware that offers special security guarantees, an intruder can take over the machine, subvert the kernel, and circumvent the provenance system. Even with a trusted pervasive hardware infrastructure and provenance tracking at every level of the system, a malicious user who can read a document can always memorize and replicate portions of it later, minus the appropriate provenance information. Since we cannot fully monitor the information flow channels available to attackers, our power to track the origin of data by monitoring read operations is limited. Given a guarantee that users could never circumvent the provenance mechanisms, we could reliably track all information flows by recording all information that each user reads, in addition to what they write. However, promulgation of provenance for read operations can result in a combinatorial explosion in overhead that can make the system unusable [16]. In this work, we target applications that do not require tracking of reads.

Consider the version history that would result if a document were created and subsequently edited and transferred from user to user, with provenance information correctly and indelibly recorded all along the way. We call this a *plausible history* for the resulting document and its chain. We target applications whose provenance integrity needs are met by the following guarantee: *if a provenance chain does not give a plausible history for its associated document, we will detect this*. Such applications are common. For example, a retail pharmacy will not accept a shipment of drugs unless it can be shown that the drugs have passed through the hands of certain middlemen. If a criminal wants to sell drugs manufactured by an unlicensed company, he will want to forge a provenance chain that gives the drugs a more respectable history, so that he can move them into the supply chain. Our approach detects that the new chain is forged. The criminal will not want to take drugs manufactured and distributed through legitimate channels, strip off their distribution provenance records, and replace them by a record showing that he distributed the drugs himself, as this new plausible history for the drugs makes them worthless. Similarly, there is little danger that someone will remove the provenance chain associated with a box of Prada accessories and try to pass them off as another brand. Instead, the incentive is to pass off non-Prada accessories as Prada, and we detect this attack.

More precisely, suppose that we have a provenance chain ([*A*], [*B*], [*C*], [*D*], [*E*], [*F*]), in which, for simplicity, each record is denoted by the identity of its corresponding principal *A, . . . ,F*. We provide the following integrity and confidentiality assurances with respect to forgery of document history.

> I1: An adversary acting alone cannot selectively remove other principals' records from the beginning or middle of a provenance chain without being detected at the next audit.
> I2: An adversary acting alone cannot add records in the beginning or the middle of the chain without being detected at the next audit.
> I3: Two colluding adversaries who have contributed records to a provenance chain cannot add records of other non-colluding users between theirs without being detected by the next audit.

For example, colluding users *B* and *D* cannot undetectably add records between their own, corresponding to fabricated actions by a non-colluding party *E*.

> I4: Once the chain contains subsequent records by non-colluding parties, two colluding adversaries who have contributed records to a provenance chain cannot remove the record of any non-colluding user between theirs without being detected by the next audit.

For example, colluding users *B* and *D* cannot remove records made by non-colluding user *C*.

A user Gertrude who can read the last record in a chain can recreate the previous version of the document by removing that record and undoing its writes to the document. Thus Gertrude and her colleagues can roll back history, then edit the old version as desired, adding new provenance records that match their actions and thereby constructing a new plausible history. However, constraint I4 prevents Gertrude from attributing any of the new writes to those who are not collaborating with her. For example, suppose Gertrude undoes records *F* and *E*, where *F* is a stamp of approval from the Food and Drug Administration. Her collaborator *E* then alters the old version of the document, generating a new provenance record *E'*. If they then try to reaffix the old FDA stamp of approval *F* without the FDA's help and cooperation, that forgery will be detected by the next audit, as the resulting provenance chain does not correspond to any plausible history.

I5: Users cannot repudiate chain records.
I6: An adversary cannot claim that a valid provenance chain for one document belongs to a document with different contents, without detection by the next audit.
I7: If an adversary alters a document without appending the appropriate provenance record to its chain, this will be detected by the next audit.
C1: Any auditor can verify the integrity of the chain without requiring access to any of its confidential components. Unauthorized access to confidential provenance record fields is prevented.
C2: The set of parties originally authorized to read the contents of a particular provenance record for D can be further restricted by subsequent writers of D.

A Secure Provenance Scheme

We proposed a solution composed of several layered components: encryption for sensitive provenance record fields, a checksum-based approach to ensure provenance record integrity, and an incremental chained signature mechanism for securing the integrity of the chain as a whole. For confidentiality, we deployed a special keying scheme based on broadcast encryption key management to selectively regulate the access for different auditors. To provide fine-grained confidentiality, we used a cryptographic commitment-based construction.

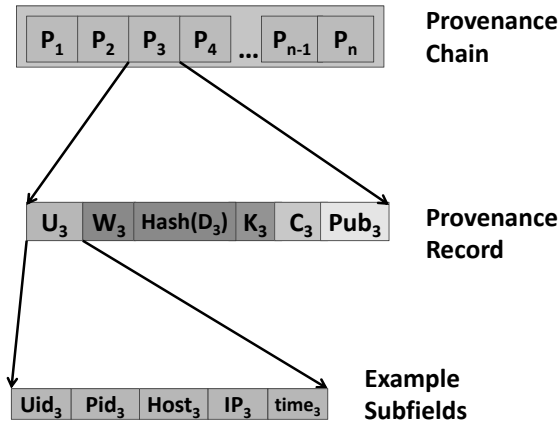


FIGURE 2: STRUCTURE OF A PROVENANCE CHAIN, SHOWING A PROVENANCE RECORD AND THE FIELDS OF ITS USER IDENTITY COMPONENT.

More precisely, each provenance record P_i summarizes a sequence of one or more actions taken by a user:

$$P_i = <U_i, W_i, hash(D), K_i, C_i, [public_i]>$$

where

- U_i is a plaintext identifier for the user;
- W_i is an encrypted or plaintext representation of the sequence of actions (the *modification log*) performed by the user;
- *hash(D)* is a one-way hash of the current content of the document;
- K_i contains keying material that auditors can use to decrypt the encrypted fields, as explained below;
- C_i contains an integrity checksum (defined below) for this provenance record, signed by user U_i;
- $public_i$ is an optional encrypted or plaintext public key certificate for user U_i.

As a practical matter, at the start of an editing session the provenance system should verify that the current contents of D match its hash value stored in the most recent provenance record. We discuss each of the record's fields below.

CONFIDENTIALITY

Certain fields or subfields of a provenance record may be sensitive, such as the identity or individual steps of a proprietary process used to clean experimental data. If all users trusted all auditors, then providing confidentiality for these sensitive fields would be straightforward—we could just encrypt all of them with a single public key, and give the private key to the auditors. If a user trusted only certain auditors, we could make several copies of the sensitive fields, encrypt each copy with the public key of a different trusted auditor, and include all of them in the new provenance record. While secure, this wastes space. Instead, as shown in Figure 3(a), we encrypt the sensitive fields of a record once with a new secret key, make multiple copies of the secret key, and encrypt each copy with the public key of a different trusted auditor. In this scheme, the new provenance record contains the encrypted sensitive fields plus several versions of the encrypted secret key, stored in the K_i field of the record. A trusted auditor can subsequently read the record, decrypt a copy of the secret key using the auditor's private key, and use the secret key to decrypt the sensitive fields. If there are many auditors, the record can be kept small by using a broadcast encryption tree to reduce the number of encrypted copies of the secret key. Other concerns such as separation of duty, or requiring a minimum number of auditors to cooperate when decrypting a secret key, can be addressed by using secret sharing and threshold encryption.

INTEGRITY

An audit must detect whether adversaries have removed or inserted elements from the chain and whether a chain has been switched from one document to another. To achieve this, the checksum C_i of a provenance record P_i is computed as shown in Figure 3(b). First we apply a cryptographic hash function to the tuple containing the user identity U_i, the hash of the document contents $hash(D)$, the modification log W_i, the key-related information K, and (if included in the record) the user's public key $public_i$. Then we concatenate the resulting hash with the checksum C_{i-1} of the previous provenance record P_{i-1}, sign the result with the user's private key, and store the signed result in the provenance record. More formally, the integrity checksum field C_i is:

$$C_i = S_{private_i}\ (hash(U_i, W_i, hash(D), K_i, [public_i])|C_{i-1})$$

where $S_{private_i}$ means that user U_i signs the hash with his or her private key.

To verify chain integrity, the auditor starts from the first record of the chain. The auditor extracts the user identity U_1 from the record and obtains $public_1$ from an external trusted source, or obtains $public_1$ from the record itself and uses an external trusted source to verify that $public_1$ is the public key of user U_1. The auditor uses the W_1 field (whether encrypted or plaintext), plus the U_1, K_1, $hash(D)$, and optional $public_1$ fields to generate a checksum C for the record. The auditor then uses $public_1$ to check that the signed checksum C_1 is in fact what would be produced by U_1 signing C. The auditor then moves on to the next record, remembering to include the signed checksum for the previous record in the computation of the checksum for the current record. Once the integrity of the chain is established, the auditor hashes the docu-

ment *D* and verifies that the resulting hash value was stored in the last provenance record.

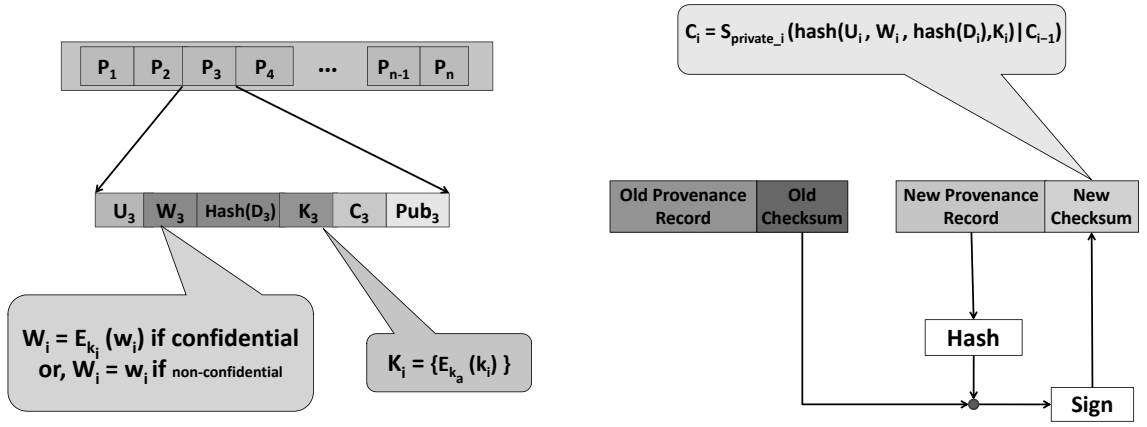


FIGURE 3: (A) CONFIDENTIALITY. W_I IS THE MODIFICATION LOG, K_I IS A SECRET KEY THAT AUTHORIZED AUDITORS CAN RETRIEVE FROM THE FIELD K_I, K_A IS THE KEY OF A TRUSTED AUDITOR. (B) INTEGRITY. NEW CHECKSUM C_I IS A FUNCTION OF THE CURRENT DOCUMENT, NEW PROVENANCE RECORD, AND THE PREVIOUS CHECKSUM C_{I-1}.

Fine-Grained Control Over Confidentiality

Some portions of a provenance record may be quite sensitive. For example, suppose that because of a Freedom of Information Act request, a document containing sensitive information has to be released to the public. Usually this is done by redacting the sensitive content of the document. However, the provenance chain for the redacted document will also contain bits and pieces of the sensitive information here and there. We cannot simply remove every record containing sensitive information, as that will break provenance integrity checks. Encrypting the entire modification log just to hide a small piece of sensitive information is excessive.

To allow selective disclosure in cases like this, we replace each sensitive field (or sub-field) in the record by a *cryptographic commitment* for it, e.g., by appending a random number to the contents of the field and then hashing the result. We encrypt those random numbers with a secret key and leave them in the record for trusted auditors to use. When we compute the checksum, we use the commitments in place of the sensitive fields, and include the encrypted random numbers. The official provenance record includes the sensitive *and* non-sensitive fields, the commitments for the sensitive fields, the encrypted random numbers, and the checksum. When we release the provenance chain we can remove the sensitive fields, and subsequent integrity checks for the chain and document will still work correctly.

SUMMARIZING CHAINS

As the document is modified over time, the provenance chain can eventually become much larger than the document. System administrators may want to summarize a provenance chain by omitting all but the records corresponding to "important" actions. The original chain construction scheme does not allow summarization through removal of records. However, we can augment the chain by including additional independently computed checksums in each record. Each checksum is computed by taking the checksum of a pre-

vious record, combining it with the hash of the current record, and then signing it. For example, one checksum may connect the current record to the previous record, while another may connect it with the record two hops away in the chain. The additional checksums allow us to remove records and still be able to prove the integrity and chronological ordering of the remaining records.

ACTIONS RECORDED IN CHAINS

In our work, we recorded writes of document data and metadata in provenance records, as well as movements of a document across organizational boundaries. We do not record reads. When the document is deleted, the provenance chain may be retained for an application-appropriate period.

Implementation and Empirical Evaluation

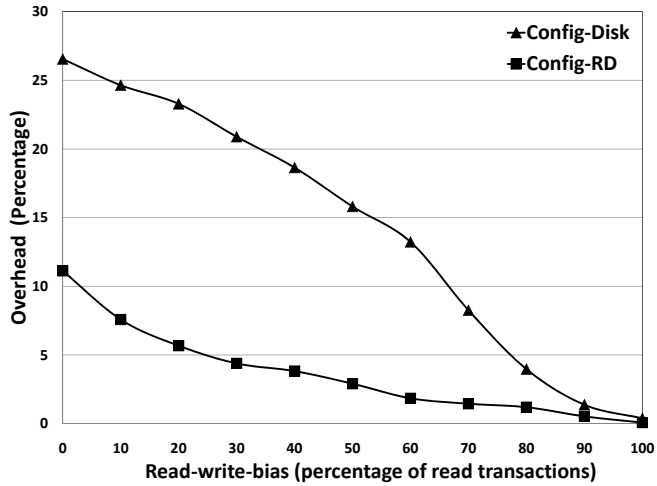


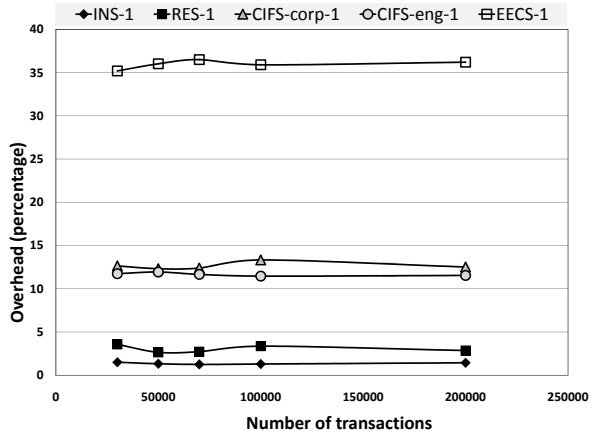
FIGURE 4: OVERHEAD OF SECURE PROVENANCE WITH POSTMARK. THE OVERHEADS ARE SHOWN FROM 0% READ BIAS (100% WRITE TRANSACTIONS) TO 100% READ BIAS (NO WRITE TRANSACTIONS), WITH OR WITHOUT A RAM DISK.

We designed and built a proof-of-concept prototype that provides secure provenance for files. The key design decision was where to place the provenance functionality. A kernel-layer or file-system-layer implementation makes provenance collection transparent to applications, but requires every user platform to run a modified kernel or file system, which hampers portability. A user-level implementation increases portability while still allowing a high degree of transparency. Thus we designed and implemented support for secure provenance through an application-layer C library called SPROV, consisting of wrapper functions for the standard file I/O library *stdio.h*.

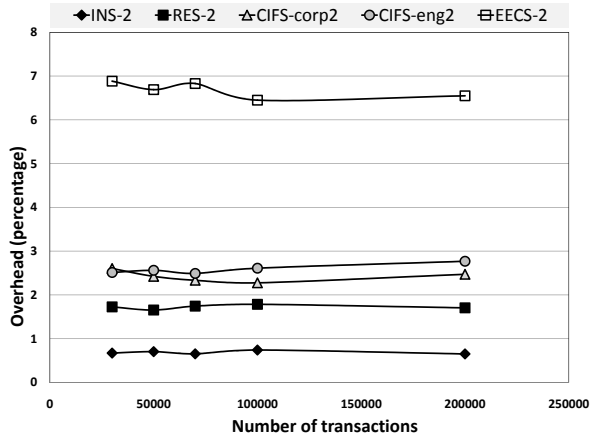
Our experimental testbed for SPROV was a workstation with an Intel Pentium 4 CPU at 3.4GHz, 2GB RAM, running Linux (Suse) at kernel version 2.6.11. In this configuration, each 1024-bit DSA signature took 1.5ms to compute. We used two modes for storing provenance chains—in the *Config-Disk* mode, the chains were stored on the disk, while in the *Config-RD* mode, we buffered the chains in a RAM disk and periodically flushed them to disk using a daemon.

We evaluated the overhead of SPROV using the standard Postmark benchmark for small files and workloads modeled on real-world traces. We ran

Postmark with secure provenance and without any provenance, and measured the runtime overhead. Modifying Postmark to use secure provenance required changing only 8 lines of code. A data set containing 20,000 Postmark-generated binary files with sizes from 8KB to 64KB was subjected to Postmark workloads of 20,000 transactions. Each transaction opened a file, issued a read or a write of size between 8KB and 64KB, then closed the file. We measured the performance overhead under different write loads by varying the percentage of write transactions from 0% to 100%, in 10% increments. As shown in Figure 4, the overheads start at 0.5% for both *Config-Disk* and *Config-RD,* ranging up to 11% for *Config-RD.*



(A)



(B)

FIGURE 5: EFFECT OF DIFFERENT WORKLOADS. (A) THE CONFIG-DISK SETTING. (B) THE CONFIG-RD SETTING.

Next we considered a more realistic scenario involving practical, documented workloads and file system layouts. We constructed a layout in the manner of Douceur and Bolosky [7], who showed that file sizes can be modeled using a log-normal distribution. We used the parameters μ^e=8.46, σ^e=2.4 to generate a distribution of 20,000 files, with a median file size of 4KB and a mean file size of 80KB. To that we added a small number of files with sizes exceeding 1GB, to account for large data objects [7].

We set the percentage of write and read transactions to match five studies of real-world file system workloads [8, 12, 18], where the write percentage ranged from 1.1% to 82.3%. These workloads came from an instructional (INS) and research (RES) setting [18], a campus home directory (EECS) [8],

and CIFS corporate and engineering workloads (CIFS-corp, CIFS-eng) [12]. The RES and INS workloads are read-intensive, with the percentage of write transactions less than 10%. The CIFS workloads have twice as many reads as writes. The EECS workload has the highest write load, with more than 80% write transactions.

As shown in Figures 5(a) and 5(b), the read-intensive workloads have almost no provenance overhead, with less than 5% overhead for both RES and INS on ordinary disk and less than 2% with a RAM disk. Write-intensive workloads on ordinary disk incur higher overheads, but still less than 14% for CIFS and less than 36% for EECS. With a RAM disk, the overheads are less than 3% for CIFS and around 6.5% for EECS.

Related Work

Numerous research efforts have tackled the issues of collecting, storing, representing, annotating, and querying provenance data, but little has been done to secure that information [4, 10]. Researchers have categorized provenance systems for science [20] and explored the question of how to capture provenance information, typically relying on workflow instrumentation [2, 15]. Provenance management systems used by scientists include Chimera for physics and astronomy, myGrid for biology, CMCS for chemistry, and ESSW for earth science [20]. Other efforts propose to collect provenance information within databases [5, 22] social networks [9], and operating and file systems [16]; the latter offers the notable advantage of being hard to circumvent.

Researchers have proposed the use of *entanglement* to preserve distributed systems' history in a non-repudiable, tamper-evident manner [14]. Provenance-related information is supported by source code management systems such as SVN [6], GIT [13], CVS [3], and Monotone [1]; versioning file systems [17]; and *secure audit forensic logs* [19, 21], to provide integrity assurances for subsets of system and data state in a logically centralized authority model. Our work targets provenance information that is highly mobile and may traverse multiple untrusted domains, with no logically centralized repository or authority. Work on audit logs typically secures logs as a whole, but does not allow authentication of individual modifications. Because provenance information is associated with a digital object such as a file, this introduces attacks that are not applicable to secure audit logs. Finally, secure audit log schemes typically assume that at most a handful of parties will process the data and compute checksums, whereas multiple principals' access is required throughout the lifetime of a provenance chain.

Conclusion

Data provenance is growing in importance as more information is shared across organizational boundaries. In this article we introduced a cross-platform, low-overhead architecture for securing provenance information. We implemented our approach for tracking the provenance of *data writes*, in the form of a library that can be linked with any application. Experimental results show that our approach imposes overheads of only 1–13% on typical real-life workloads. Further details are available in our FAST '09 paper [11], and on our Web site (http://tinyurl.com/secprov).

ACKNOWLEDGMENTS

We thank Bill Bolosky and John Douceur for tips about file system distribution. Hasan and Winslett were supported by NSF awards CNS-0716532 and CNS-0803280. Sion was supported by Xerox, IBM, and the NSF through awards CNS-0627554, CNS-0716608, CNS-0708025, and IIS-0803197.

REFERENCES

[1] Monotone Distributed Version Control: http://www.monotone.ca/, accessed on December 22, 2008.

[2] R. Aldeco-Perez and L. Moreau, "Provenance-Based Auditing of Private Data Use," *Proceedings of the BCS International Academic Research Conference, Visions of Computer Science*, 2008.

[3] B. Berliner, "CVS II: Parallelizing Software Development," *Proceedings of the Winter 1990 USENIX Conference,* USENIX Association, 1990, pp. 341–352.

[4] U. Braun, A. Shinnar, and M. Seltzer, "Securing Provenance," *Proceedings of the 3rd USENIX Workshop on Hot Topics in Security (HotSec '08)*, USENIX Association, 2008.

[5] P. Buneman, A. Chapman, and J. Cheney, "Provenance Management in Curated Databases," *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, ACM Press, 2006, pp. 539–550.

[6] B. Collins-Sussman, "The Subversion Project: Building a Better CVS," *Linux Journal* 2002(94): 3.

[7] J.R. Douceur and W.J. Bolosky, "A Large-Scale Study of File-System Contents," *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS),* ACM Press, 1999, pp. 59–70.

[8] D. Ellard, J. Ledlie, P. Malkani, and M. Seltzer, "Passive NFS Tracing of Email and Research Workloads," *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, USENIX Association, 2003, pp. 203–216.

[9] J. Golbeck, "Combining Provenance with Trust in Social Networks for Semantic Web Content Filtering," *International Provenance and Annotation Workshop*, in volume 4145 of *Lecture Notes in Computer Science*, L. Moreau and I.T. Foster, eds., Springer, 2006, pp. 101–108.

[10] R. Hasan, R. Sion, and M. Winslett, "Introducing Secure Provenance: Problems and Challenges," *Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS)*, ACM Press, 2007, pp. 13–18.

[11] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST '09)*, USENIX Association, 2009.

[12] A.W. Leung, S. Pasupathy, G. Goodson, and E.L. Miller, "Measurement and Analysis of Large-Scale Network File System Workloads," *Proceedings of the 2008 USENIX Annual Technical Conference*, USENIX Association, 2008, pp. 213–226.

[13] J. Loeliger, "Collaborating Using GIT," *Linux Magazine*, June 2006.

[14] P. Maniatis and M. Baker, "Secure History Preservation through Timeline Entanglement," *Proceedings of the 11th USENIX Security Symposium*, USENIX Association, 2002, pp. 297–312.

[15] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga, "The Provenance of Electronic Data," *Communications of the ACM*, 51(4): 52–58 (2008).

[16] K.-K. Muniswamy-Reddy, D.A. Holland, U. Braun, and M.I. Seltzer. "Provenance-Aware Storage Systems," *Proceedings of the 2006 USENIX Annual Technical Conference*, USENIX Association, 2006, pp. 43–56.

[17] Z.N.J. Peterson, R. Burns, G. Ateniese, and S. Bono, "Design and Implementation of Verifiable Audit Trails for a Versioning File System," *Proceedings of the 5th USENIX conference on File and Storage Technologies (FAST '07)*, USENIX Association, 2007, pp. 93–106.

[18] D. Roselli, J.R. Lorch, and T.E. Anderson, "A Comparison of File System Workloads," *Proceedings of the 2000 USENIX Annual Technical Conference*, USENIX Association, 2000.

[19] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics," *ACM Transactions on Information and System Security*, 2(2): 159–176 (1999).

[20] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in E-Science," *ACM SIGMOD Record*, 34(3): 31–36 (Sept. 2005).

[21] R. Snodgrass, S. Yao, and C. Collberg, "Tamper Detection in Audit Logs," *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB),* VLDB Endowment, 2004, pp. 504–515.

[22] J. Widom, "Trio: A System for Integrated Management of Data, Accuracy, and Lineage," *Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR)*, January 2005.