The Case of the Fake Picasso!
Preventing History Forgery with Secure Provenance

Ragib Hasan*, Radu Sion+, Marianne Winslett*
Dept. of Computer Science
* University of Illinois at Urbana-Champaign
+ Stony Brook University

USENIX FAST 2009
February 25, 2009
Let’s play a game

Real, worth $101.8 million

Fake, listed at eBay, worth nothing

Can you spot the fake Picasso?
So, how do art buyers authenticate art?

Among other things, they look at **provenance records**

**Provenance**: from Latin *provenire* ‘come from’, defined as

“(i) the fact of coming from some particular source or quarter; origin, derivation.

(ii) the history or pedigree of a work of art, manuscript, rare book, etc.; a record of the ultimate derivation and passage of an item through its various owners” (Oxford English Dictionary)

In other words, **who owned it, what was done to it, how was it transferred** ...

Widely used in arts, archives, and archeology, called the Fundamental Principle of Archival.

L'artiste et son modèle (1928), at Museum of Modern Art

Let’s consider the digital world

Data is generated, processed, and transmitted between different systems and principals, stored in databases or storage.

Unlike data processing in the past, digital data can be rapidly copied, modified, and erased.

To trust data we receive from others or retrieve from storage, we need to look into the integrity of both the present state and the past history of data.

Am I getting back untampered data?

Was this data created and processed by persons I trust?

Our life today has become increasingly dependent on digital data. Our most valuable asset is data.
What exactly is data provenance?

Definition*

– Description of the **origins** of data and the **process** by which it arrived at the database. [Buneman et al.]

– Information describing materials and **transformations** applied to derive the data. [Lanter]

– Information that helps determine the **derivation history** of a data product, starting from its original sources. [Simmhan et al.]

### Example provenance systems

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Processing Framework</td>
<td>Command Processing</td>
<td>Service Oriented</td>
<td>Service Oriented</td>
<td>Service Oriented</td>
<td>Service Oriented</td>
<td>Script Based</td>
<td>Relational Database</td>
<td>Relational/Semi Structured Database</td>
<td>Relational Database</td>
</tr>
<tr>
<td>Application of Provenance</td>
<td>Informational; update state, regenerate &amp; compare data</td>
<td>Informational, Audit, Data Regeneration; Planning</td>
<td>Contextual Information; Re-enactment</td>
<td>Informational, Data Update</td>
<td>Informational, Re-enactment</td>
<td>Informational</td>
<td>Track errors</td>
<td>Annotation propagation; View Update</td>
<td>Information; update propagation</td>
</tr>
<tr>
<td>Data/Process Oriented</td>
<td>Data</td>
<td>Process</td>
<td>Process</td>
<td>Data</td>
<td>Process</td>
<td>Both</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Granularity</td>
<td>Spatial layers</td>
<td>Abstract datasets (Currently files)</td>
<td>Abstract resources having LSID</td>
<td>Files</td>
<td>Abstract parameters to Workflow</td>
<td>Files</td>
<td>Attributes in Database</td>
<td>Attributes &amp; Tuples in Databases</td>
<td>Tuples in Database</td>
</tr>
<tr>
<td>Representation Scheme</td>
<td>Commands &amp; Frames as Annotations</td>
<td>Virtual Data Language Annotations</td>
<td>XML/RDF Annotations</td>
<td>Dublin Core XML Annotations</td>
<td>Annotations</td>
<td>XML/RDF Annotations</td>
<td>Inverse Functions</td>
<td>Inverse Queries</td>
<td>Inverse queries</td>
</tr>
<tr>
<td>Semantic Information</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Limited</td>
<td>No</td>
<td>No, Proposed</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Storage Repository/Backend</td>
<td>MetaDatabase</td>
<td>Virtual Data Catalog/Relational DB</td>
<td>mIR repository/Relational DB</td>
<td>SAM over WebDAV/Relational DB</td>
<td>PReDev/Relational DB, File System</td>
<td>Lineage Server/Relational DB</td>
<td>Relational DB</td>
<td>N/A</td>
<td>Relational DB</td>
</tr>
<tr>
<td>Provenance Collection Overhead</td>
<td>Store User commands, solicit metadata</td>
<td>User defines derivations; automated WF trace</td>
<td>User defines service semantics; Automated WF Trace</td>
<td>Manual; Apps use DAV APIs, Users use portal</td>
<td>Manual; Actors use PReP API</td>
<td>Libraries assist user to generate, store provenance</td>
<td>User registers inverse functions</td>
<td>N/A</td>
<td>Inverse queries automatically generated</td>
</tr>
<tr>
<td>Addressed Scalability</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No (Proposed)</td>
<td>No (Proposed)</td>
<td>Yes</td>
<td>N/A</td>
<td>No</td>
</tr>
<tr>
<td>Provenance Dissemination</td>
<td>Queries</td>
<td>Queries</td>
<td>Semantic browser; Lineage graph</td>
<td>Browser; Queries; GXL/RDF</td>
<td>Queries</td>
<td>Browser</td>
<td>Queries; box-and-arrows visualization</td>
<td>N/A</td>
<td>SQL/TriQL Queries</td>
</tr>
</tbody>
</table>

Simmhan et al., 2005
What was the common theme of all those systems?

• They were all scientific computing systems
• And scientists trust people (more or less)

• Previous research covers provenance collection, annotation, querying, and workflow, but security issues are not handled

• For provenance in untrusted environments, we need integrity, confidentiality and privacy guarantees

So, we need provenance of provenance, i.e. a model for Secure Provenance
Secure provenance means preventing “undetectable history rewriting”

- Adversaries cannot insert fake events, remove genuine events from a document’s provenance
- No one can deny history of own actions

- **Allow** fine grained preservation of privacy and confidentiality of actions
  - Users can choose which auditors can see details of their work
  - Attributes can be selectively disclosed or hidden without harming integrity check
Usage and threat model

- **Users**: Edit **documents** on their machines
- **Auditors**: semi-trusted principals
  - All auditors can verify chain integrity
  - Only certain auditors can read each entry

**Adversaries**: insiders or outsiders
- Add or remove history entries
- Collude with others to add/remove entries
- Claim a chain belongs to another document
- Repudiate an entry

Previous work on integrity assurances

• (Logically) centralized repository (CVS, Subversion, GIT)
  – Changes to files recorded
  – Not applicable to mobile documents
• File systems with integrity assurances (SUNDR, PASIS, TCFS)
  – Provide local integrity checking
  – Do not apply to data that traverses systems
• System state entanglement (Baker 02)
  – Entangle one system’s state with another, so others can serve as
    witness to a system’s state
  – Not applicable to mobile data
• Secure audit logs / trails (Schneier and Kelsey 99), LogCrypt
  (Holt 2004), (Peterson et al. 2006)
  – Trusted notary certifies logs, or trusted third party given hash
    chain seed
Our solution: Overview

\[ U_i = \text{identity of the principal (lineage)} \]

\[ W_i = \text{Encrypted modification log} \]

\[ K_i = \text{confidentiality locks for } W_i \]

\[ C_i = \text{integrity checksum(s)} \]
Our solution: Confidentiality

A single auditor

Modification log → Encrypted Modification log

Multiple

Issues

• Each user trusts a subset of the auditors

• Only the auditor(s) trusted by the user can see the user’s actions on the document

Modification log → Encrypted Modification log

Optimization: Use broadcast encryption tree to reduce number of required keys
Our solution: Confidentiality

- \( W_i = E_{k_i}(w_i) \mid \text{hash}(D) \)

- \( K_i = \{ E_{k_a}(k_i) \} \)

- \( k_i \) is a secret key that authorized auditors can retrieve from the field \( K_i \)

- \( w_i \) is either the diff or the set of actions taken on the file

- \( k_a \) is the key of a trusted auditor
Our solution: Integrity

\[ C_i = S_{private_i}(hash(U_i, W_i, K_i) | C_{i-1}) \]
Fine grained control over confidentiality

Classified Document

Declassify / release

Redacted (unclassified) Document

- Provenance chain has sensitive info
- Deleting sensitive information will break integrity checks

Original attributes

Nonsensitive Information  Sensitive Information

- Commit(sensitive info)

Nonsensitive Info  Sensitive info  Commit(sensitive info)

Checksum calculation

Nonsensitive Info  Commit(sensitive info)

Disclosable provenance entry

Blinded entry disclosed to third party
We can summarize provenance chains to save space, make audits fast

Each entry has 1 checksum, calculated from 1 previous checksum

Each entry has n checksums, each of them calculated from 1 previous checksum

We can systematically remove entries from the chain while still being able to prove integrity of chain
Our Sprov application-level library requires almost no application changes

- Sprov provides the file system APIs from stdio.h
- To add secure provenance, simply 
  **relink** applications with Sprov library instead of stdio.h
Experimental settings

Crypto settings
- 1024 bit DSA signatures
- 128 bit AES encryption
- SHA-1 for hashes

Experiment platform
- Linux 2.6.11 with ext3
- Pentium 3.4 GHz, 2GB RAM,
- Disks: Seagate Barracuda 7200 rpm, WD Caviar SE16 7200 rpm

Modes
- **Config-Disk**: Provenance chains stored on Disk
- **Config-RD**: Provenance chains stored on RAM Disk buffer, and periodically saved to disk
Postmark small file benchmark:
Overhead < 5% for realistic workloads

- **20,000** small files (8KB-64KB) subjected to 100% to 0% write load with the Postmark benchmark

- At 100% write load, execution time overhead of using secure provenance over the no-provenance case is approx. 27% (12% with RD)

- At 50% write load, overheads go down to 16% (3% with RD)

- Overheads are less than 5% with 20% or less write load

100% writes, 0% reads

0% writes, 100% reads
Hybrid workloads: Simulating real file systems

File system distribution:
- File size distribution in real file systems follows the log normal distribution [Bolosky and Douceur 99]
- Median file size = 4KB, mean file size = 80KB
- We created a file system with 20,000 files, using the lognormal parameters \( \mu = 8.46 \), \( \sigma = 2.4 \)
- In addition, we included a few large (1GB+) files

Workload
- **INS**: Instructional lab (1.1\% writes) [Roselli 00]
- **RES**: A research lab (2.9\% writes) [Roselli 00]
- **CIFS-Corp**: (15\% writes) [Leung 08]
- **CIFS-Eng**: (17\% writes) [Leung 08]
- **EECS**: (82\% writes) [Ellard 03]
Typical real life workloads: 1 - 13% overhead

- **INS** and **RES** are read-intensive (80%+ reads), so overheads are very low in both cases.
- **CIFS-corp** and **CIFS-eng** have 2:1 ratio of reads and writes, overheads are still low (range from 12% to 2.5%)
- **EECS** has very high write load (82%+), so the overhead is higher, but still less than 35% for Config-Disk, and less than 7% for Config-RD
Summary: Secure provenance possible at low cost

Yes, We CAN achieve secure provenance with integrity and confidentiality assurances with reasonable overheads

– For most real-life workloads, overheads are between 1% and 15% only

More info at http://tinyurl.com/secprov