LINCS: Towards Building a Trustworthy Litigation Hold Enabled Cloud Storage System

Shams Zawoad, Ragib Hasan, and John Grimes

University of Alabama at Birmingham, Alabama, USA
Outline

- Background
  - Litigation Hold | Spoliation | Model

- Motivation
  - Case Study | Current Solution | Research Gap

- Litigation hold enabled Cloud Storage (LINCS) system
  - Threat Model | Protocol | Security | Results | Tool

- Conclusion & Future Work
Legal notice to a defendant that triggers the preservation of ESI

Preservation obligation comes from common law, statutes, regulations, or a court order

Litigation holds in the cloud

Defendant’s data is now under the direct control of a third party
Deliberate destruction or modification of ESI by a litigant party

Defendant produces proof of preservation of litigation hold

Plaintiff provides the evidence of spoliation

Can cost reputation, fines, penalties, etc.
**Background | Litigation Hold Model**

- **T\(_s\)**: Litigation hold is issued
- **T\(_e\)**: Litigation hold ends
  \[ \Delta L = T_e - T_s \]
- **T\(_c\)**: CSP turns malicious
  \[ \Delta H = T_c - T_s, \ H > 0 \]
  \[ \Delta M = T_e - T_c \]
- **T\(_v\)**: Verification time

Diagrams showing user storage with safe deletion, spoliation, and verification time.
Motivation | Current Solutions

- Legal hold framework in clouds [Schmidt (2012)]

- Provable Data Possession schemes in clouds [Ateniese et al. (2007); Erway et al. (2009)]

Schmidt (2012): Cloud providers are considered as trustworthy

PDP: Metadata are generated by clients, who are considered as trustworthy

No one considered the collusion between CSP, plaintiff and defendant.
Proposing LINCS

- Litigation hold eNabled Cloud Storage (LINCS)
- Defendant or a plaintiff can collude with a malicious CSP
- Secure verifiable proof of file creation and deletion.
LinCs | Threat Model

- Removes files on hold from the cloud storage
- Denies the ownership of files presented by plaintiff

- Bypass the proof of deletion preservation system
- Remove the proof of file
- Present an act of spoliation as a safe deletion operation
- deny hosting a file presented by plaintiff
External attackers extract information from proofs of files or the proofs of file deletions.

- Remove file without defendant’s consent.
- Present a safe deletion operation as an act of spoliation.
- Plant a back-dated fake file to the defendant’s storage.

External attackers extract information from proofs of files or the proofs of file deletions.
**LINCS | File Upload Protocol**

\[ FCM_U^i = < (H(F^i)|F_{ID}^i |CT_U^i), \]
\[ \text{Sig} (H(F^i)|F_{ID}^i |CT_U^i) > \]

\[ FCM_C^i = < (H(F^i)|F_{ID}^i |CT_C^i), \]
\[ \text{Sig} (H(F^i)|F_{ID}^i |CT_C^i) > \]

\[ PF^i = < \text{Mac}_{MKC^i} (CS^i),(CS^i) > \]
\[ CS^i = < FCM_U^i|FCM_C^i > \]

\[ MK_C^i = < \text{MKey}(H(MK_C^{i-1})) >, \]
\[ MK_C^0 = < \text{MKey}(H(S_C)) > \]
LINCS | File Deletion Protocol

FDR\(^i\) = < (H(F\(^i\))|F_{ID}^{i} |DT_{U}),
Sig (H(F\(^i\))|F_{ID}^{i} |DT_{U})>

FDA\(^i\) = < (H(F\(^i\))|F_{ID}^{i} |DT_{C}),
Sig (H(F\(^i\))|F_{ID}^{i} |DT_{C})>

AR = < (Response|F_{ID}^{i} |AT_{U} ), Sig
(Response|F_{ID}^{i} |AT_{U})>

PD\(^i\) = < \text{Mac}_{MK_{Di}^{i}} (DS\(^i\)),(DS\(^i\)) >
DS\(^i\) = <FDR\(^i\)|FDA\(^i\)| AR >

MK_{D}^{i} = < \text{MKey}(H(MK_{D}^{i-1})) >,
MK_{D}^{0} = < \text{MKey}(H(S_{D})) >
LINCS | Verification

No

Continue checking next proof of deletion

Yes

Signature Valid?

Yes

MACa(DS0)

No

Reject

PKc

PKu

PD0

DS0 MAC(DS0)

Yes

Equal?

Yes

Last proof of the day t?

No

Accept

Reject

DSpt MAC(DSpt)

PDpt

Reject

Equal?
**Proposition 1:** Defendant cannot deny the possession of a file $F_i$

- $P_{F_i}$ is attached with the $F_i$ a metadata.
- $P_{F_i}$ includes $FCM_{U_i}$, which contains the signature of the defendant
**Proposition 2:** Defendant cannot deny the proof of deletion $PD_i$ for the $F_i$ file

- $PD_i$ for the $F_i$ contains file deletion request $FDR_i$ and acknowledgement receipt $AR$
- Defendant signed these two components
Proposition 3: If $F^d$ is removed before $T_s$, $PD^d$ cannot be placed after $T_s$

- $PD^{d+1}$ will be appeared after $PD^{d-1}$ in the altered proof of deletion chain.

- At $i=d$, the auditor creates $MAC_A(DS^{d+1})$ from the MAC key $MK_D^d$, but the actual $MAC(DS^{d+1})$ was created using $MK_D^{d+1}$. 
Proposition 4: If $F^d$ is removed after $T_s$, $PD^d$ cannot be placed before $T_s$

- $PD^d$ will be appeared after $PD^j$ in the altered proof of deletion chain.

- $MAC_A(DS^d) \neq MAC(DS^d)$, since the $MAC(DS^d)$ was not calculated using the key $MK^j_{D^j+1}$
**Proposition 5:** Auditor can detect the act of spoliation if a defendant removes a file $F^d$ during $\Delta L$ and the plaintiff presents the file $F^d$ to the court.

- There must be a proof of deletion $PD^d$.
- $PF^d$ can prove the existence of the file $F^d$ in the defendant’s cloud storage.
**Proposition 6:** $F_d$ is removed without the defendant’s consent, the plaintiff cannot prove this deletion as an act of spoliation.

- The plaintiff needs to present the proof of deletion $PD_d$
- $PD_d$ should contain the defendant’s signature with the $FDR_d$ and the AR
Proposition 7: CSP cannot add a fake file $F^f$ to the defendant’s storage without being detected by the auditor.

- The $PF^f$ includes $FCM^f_U$, which is signed by the defendant.
- Presenting the $F^f$ file as a backdated file requires modification in the chain of $PF$. 
**Proposition 8:** An adversary cannot identify the content of the file Fi from the PFi or PDi

- PFi and the proof of deletion PDi are created from the hash of the Fi
- One-way, collusion resistant hash function prevents reverse engineering the proofs
System Configuration

- Ftp server in an AmazonEC2 medium (m1.medium) instance running Ubuntu 12.04.4 LTS
- (LHM) module was running inside the EC2 instance
- RSA (2048 bit) for encryption, SHA-256
- Oracle JDK (version 1.7.051) to implement the modules of LINCS
LINCS | Client Overhead

![Graph showing file size versus log10 of time and overhead. The graph compares Upload With FCM and Regular Upload, with markers for Average Overhead % and Overhead %.]
LINCS | Storage Overhead

%Overhead

File Size (MB)

- File Size (MB): 2, 4, 8, 16, 32
- %Overhead: 0.045, 0.04, 0.035, 0.03, 0.025, 0.02, 0.015, 0.01, 0.005, 0
LINCS | Verification Performance

![Graph 1](Time vs Number of files)

- Time (Minute) on the y-axis
- Number of files on the x-axis (ranging from 0 to 1000)

![Graph 2](Time vs Size)

- Time (Minute) on the y-axis
- Size (GB) on the x-axis (ranging from 0 to 70)

The graphs illustrate the relationship between the number of files and time taken, as well as the relationship between size (in GB) and time taken for verification performance.
LINCS | Verification Tool
Conclusion & Future Work

- Defined a model of trustworthy litigation holds in clouds
- Proposed LINCS that ensures trustworthy management of litigation holds in a cloud storage.

Future Plan:
- Include dynamic behavior of the cloud storage,
- New files creation after Ts,
- Security of the special types of file, such as MBOX or EML.
Thank You

Shams Zawoad

zawoad@cis.uab.edu
LINCS | Proof Creation Performance

![Graph showing Proof Creation Performance](image-url)

- **PrepPFChain**
- **PreparePDChain**
**Lemma 1:** \(PF^i\) is the proof provided by the user and the CSP about the existence of the \(F^i\) file.

**Lemma 2:** \(PD^i\) is the proof provided by the user and the CSP about the deletion of the \(F^i\) file.

**Lemma 3:** The secret keys and the initial secrets \(S_D\) and \(S_C\) cannot be accessed by an adversary.

**Lemma 4:** CSP cannot alter the published proofs or deny the existence of the published proofs.