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

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

Knowledge that will change your world

SECuRE and **T**rustworthy computing **Lab**
Department of CIS, UAB
http://secret.cis.uab.edu

DFRWS USA 2015

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

Background | Litigation Hold

- 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

Background | Spoliation

- 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

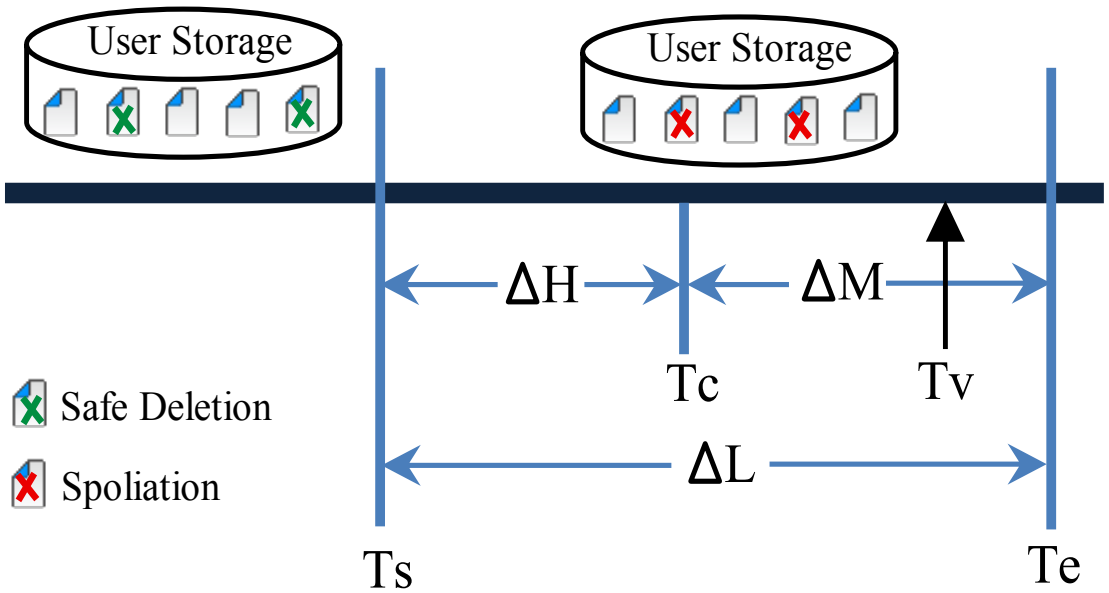
$\Delta L = T_e - T_s$

T_c: CSP turns malicious

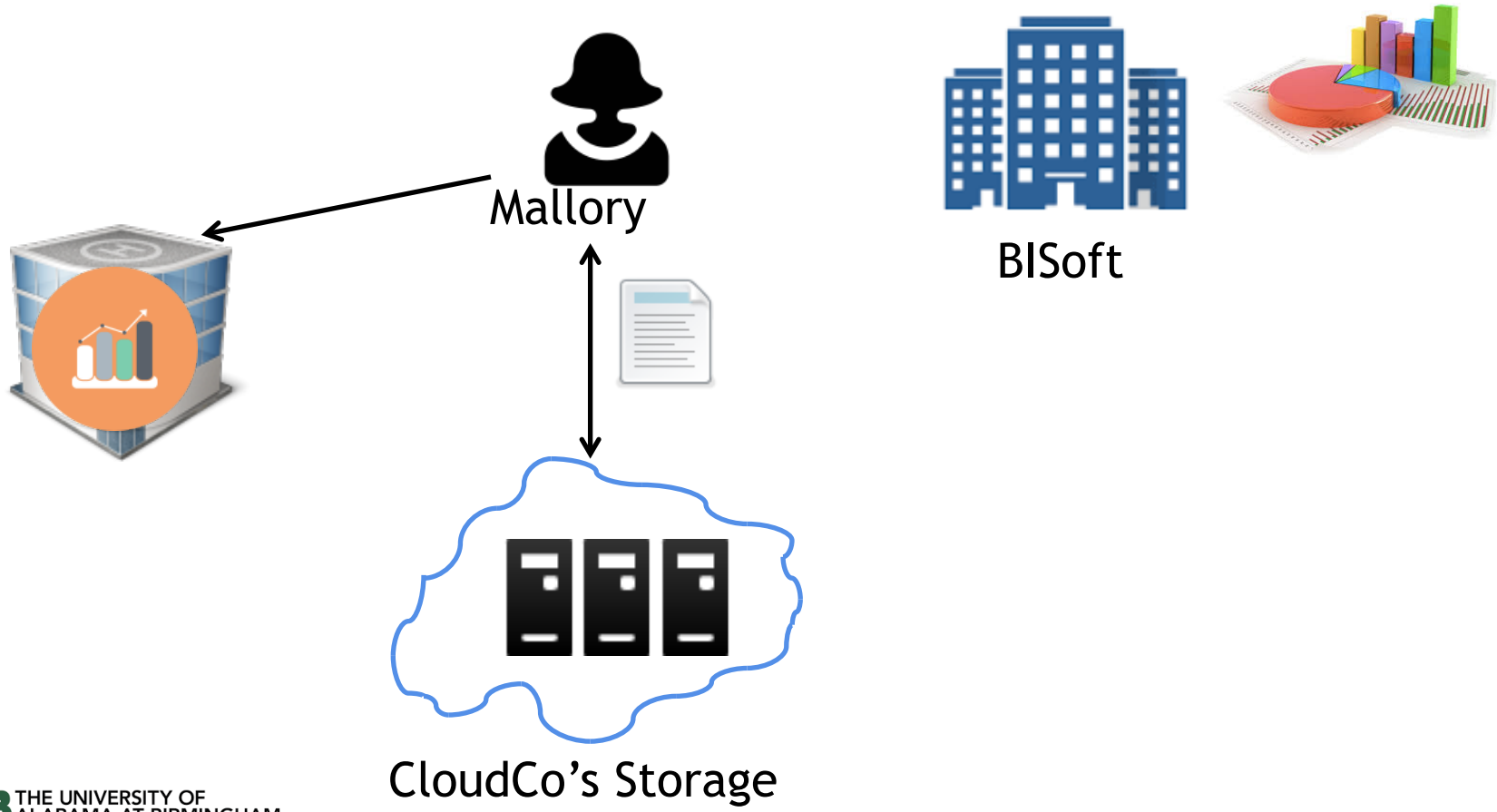
$\Delta H = T_c - T_s$, $H > 0$

$\Delta M = T_e - T_c$

T_v: Verification time



Motivation | Case Study



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, O., 2012. Managing a legal hold on cloud documents. US Patent App.13/543,254.
-Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D., 2007. Provable data possession at untrusted stores. In: 14th ACM conference on Computer and communications security. ACM, pp. 598–609.
-Erway, C., Kupcu, A., Papamanthou, C., Tamassia, R., 2009. Dynamic provable data possession. In: 16th ACM conference on Computer and communications security. ACM, pp. 213–222.

Motivation | Research Gap

- 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

- **LI**itigation hold e**N**abled **C**loud **S**torage (*LINCS*)
- Defendant or a plaintiff can collude with a malicious CSP
- Secure verifiable proof of file creation and deletion.

LINCS | Threat Model

C P D

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

C P D

- 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

LINCS | Threat Model

C P D

- 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

C P D

- External attackers extract information from proofs of files or the proofs of file deletions

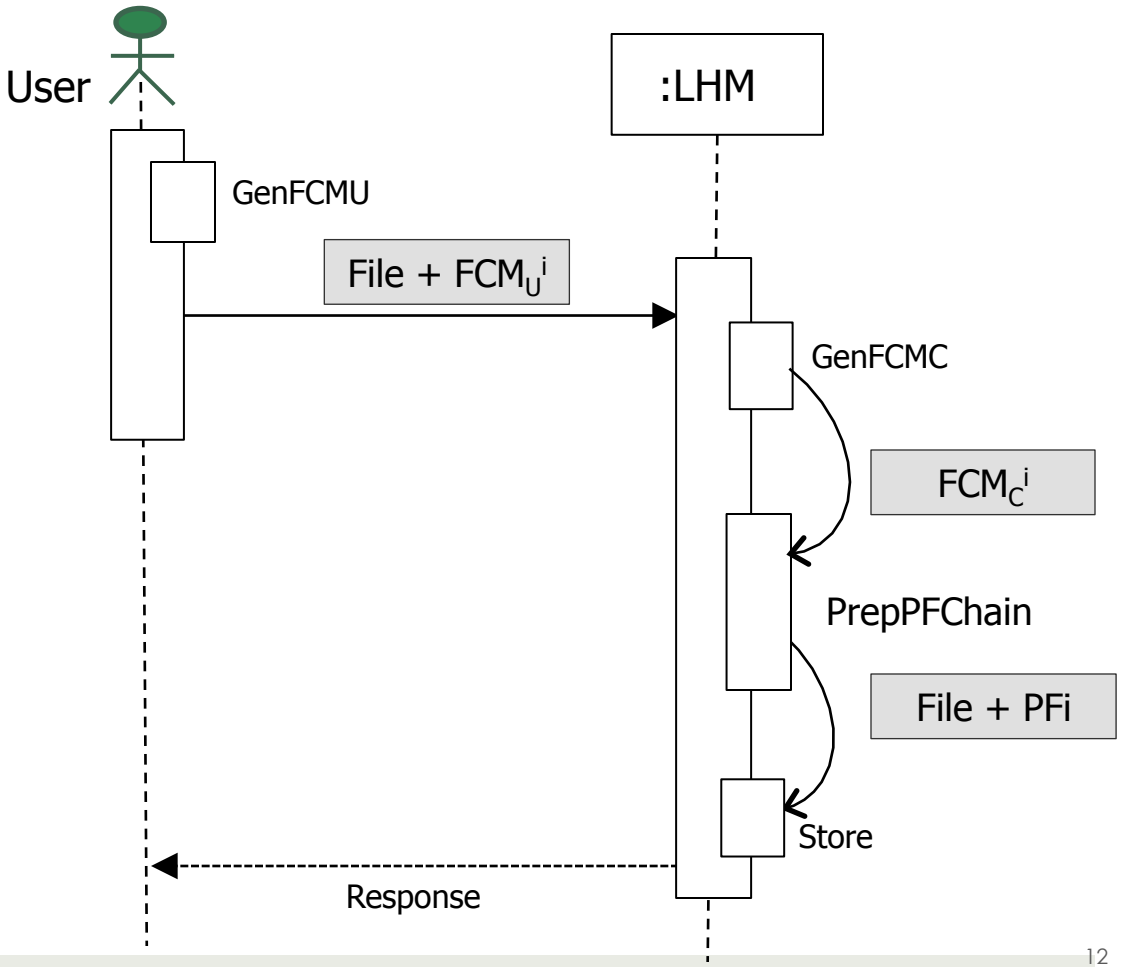
LINCS | File Upload Protocol

$FCM_U^i =< (H(F^i)|F^i_{ID} |CT_U),$
$Sig\ (H(F^i)|F^i_{ID} |CT_U)>$

$FCM_C^i =< (H(F^i)|F^i_{ID} |CT_C),$
$Sig\ (H(F^i)|F^i_{ID} |CT_C)>$

$PF^i =< Mac_{MKCi}\ (CS^i),(CS^i) >$
$CS^i = <FCM_U^i|FCM_C^i >$

$MK_C^i =< MKey(H(MK_C^{i-1})) >,$
$MK_C^0 =< MKey(H(S_C)) >$



LINCS | File Deletion Protocol

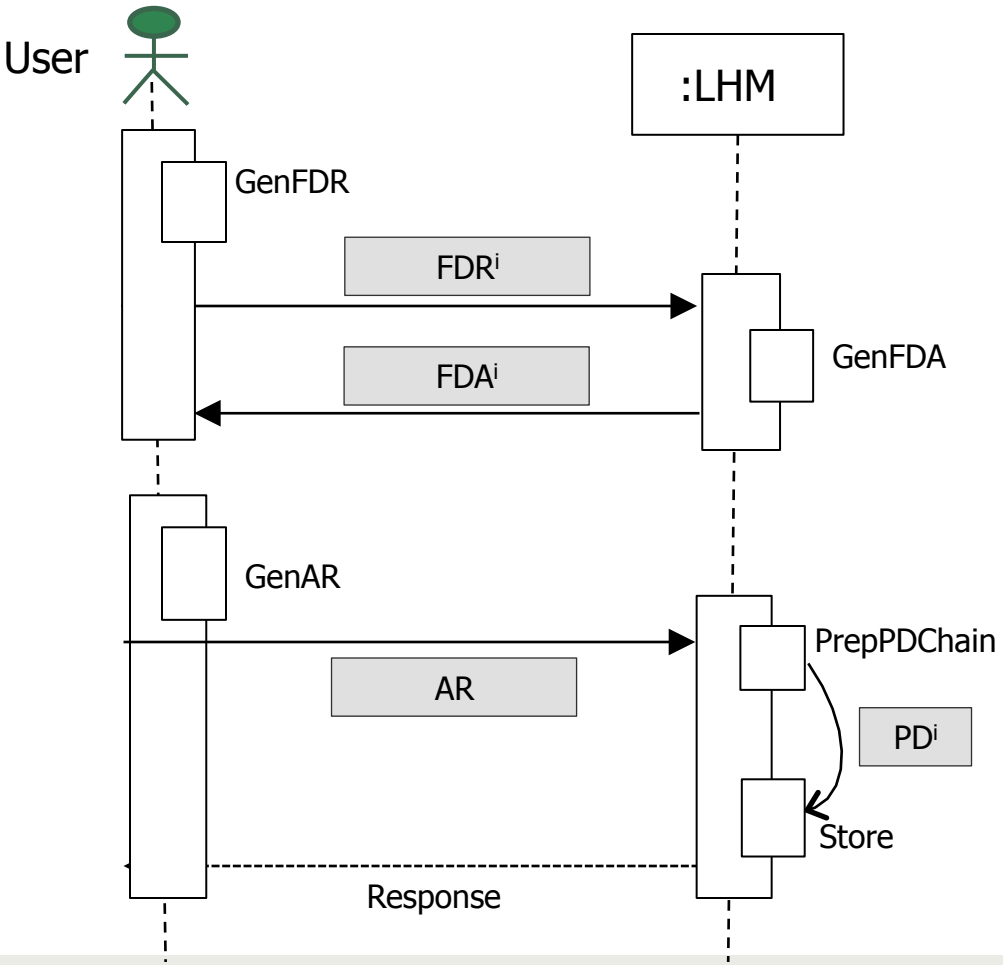
$FDR^i =< (H(F^i)|F^i_{ID}|DT_U),$
$Sig\ (H(F^i)|F^i_{ID}|DT_U)>$

$FDA^i =< (H(F^i)|F^i_{ID}|DT_C),$
$Sig\ (H(F^i)|F^i_{ID}|DT_C)>$

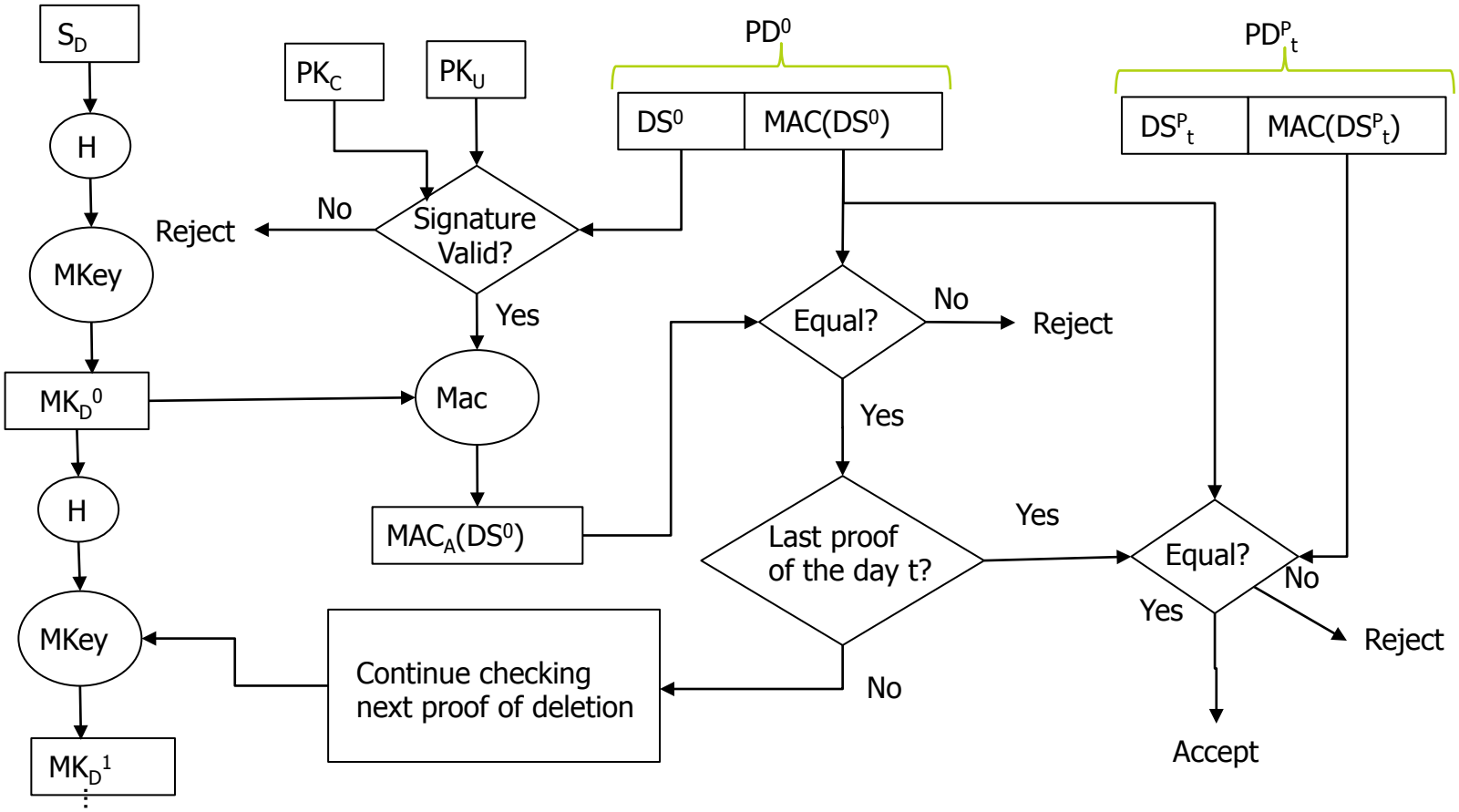
$AR =< (Response|F^i_{ID}|AT_U), Sig$
$(Response|F^i_{ID}|AT_U)>$

$PD^i =< Mac_{MKDi}(DS^i),(DS^i) >$
$DS^i = <FDR^i|FDA^i|\ AR >$

$MK_D^i =< MKey(H(MK_D^{i-1})) >,$
$MK_D^0 =< MKey(H(S_D)) >$



LINCS | Verification



LINCS | Security Propositions

Proposition 1: Defendant cannot deny the possession of a file F^i

- PF^i is attached with the F^i a metadata.
- PF^i includes $FCM_U{}^i$, which contains the signature of the defendant

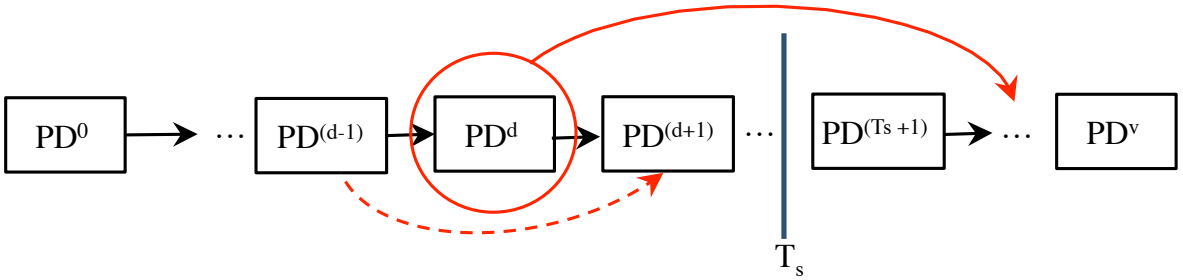
LINCS | Security Propositions

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

LINCS | Security Propositions

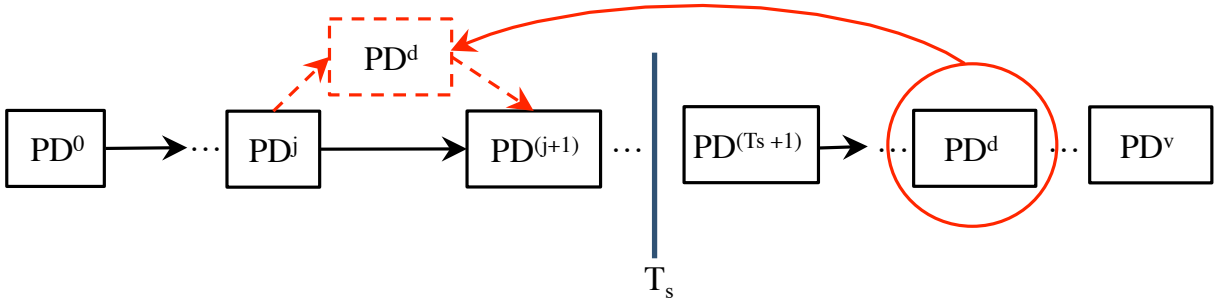
Proposition 3: If F^d *is removed before* Ts, PD^d *cannot be placed after* Ts



- 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}

LINCS | Security Propositions

Proposition 4: If F^d is removed after Ts , PD^d cannot be placed before Ts



- PD^d will be appeared after PD^j in the altered proof of deletion chain.
- $MAC_A(DS^d)$!= $MAC(DS^d)$, since the $MAC(DS^d)$ was not calculated using the key MK_D^{j+1}

LINCS | Security Propositions

Proposition 5: Auditor can detect the act of spoliation if a defendant removes a file F^d during ΔL 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

LINCS | Security Propositions

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

LINCS | Security Propositions

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_U^f , which is signed by the defendant.
- Presenting the F^f file as a backdated file requires modification in the chain of PF

LINCS | Security Propositions

Proposition 8: An adversary cannot identify the content of the file Fi from the PF^i or PD^i

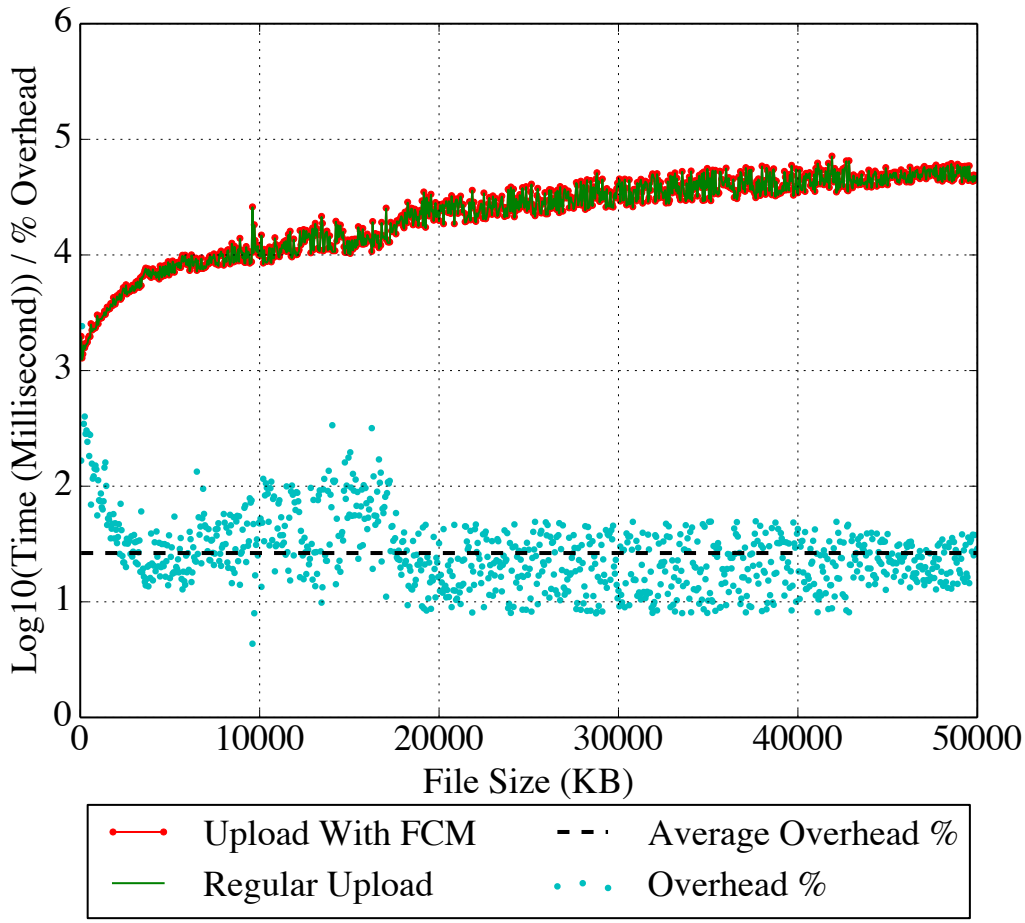
- PF^i and the proof of deletion PD^i are created from the hash of the F^i
- One- way, collusion resistant hash function prevents reverse engineering the proofs

LINCS | Results

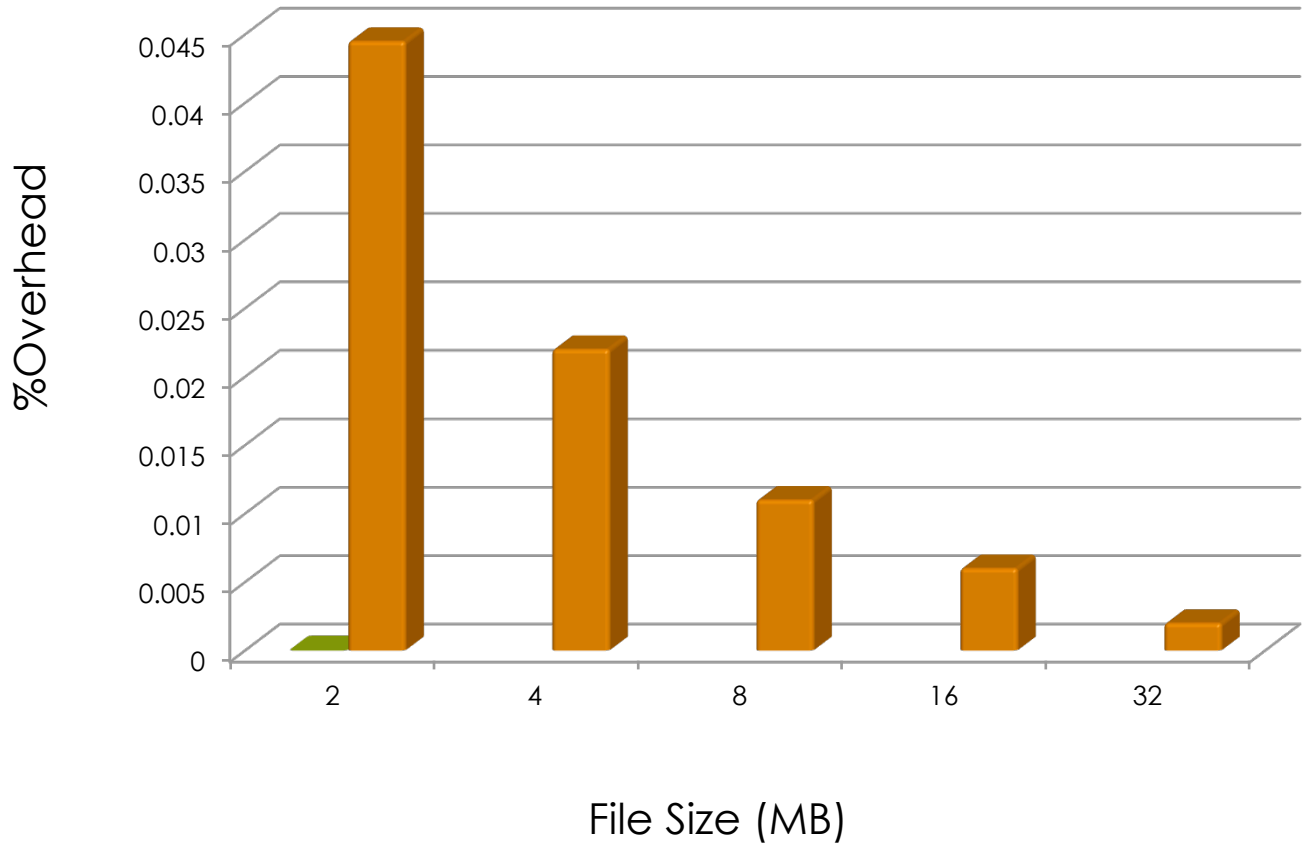
System Configuration

- 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 (version1.7.051) to implement the modules of *LINCS*

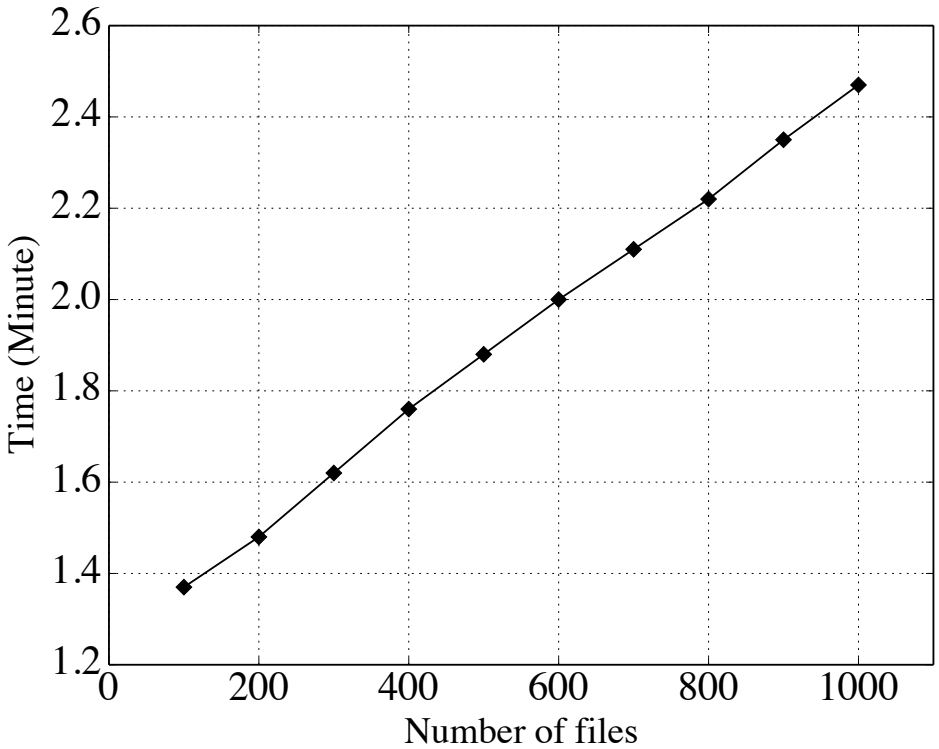
LINCS | Client Overhead

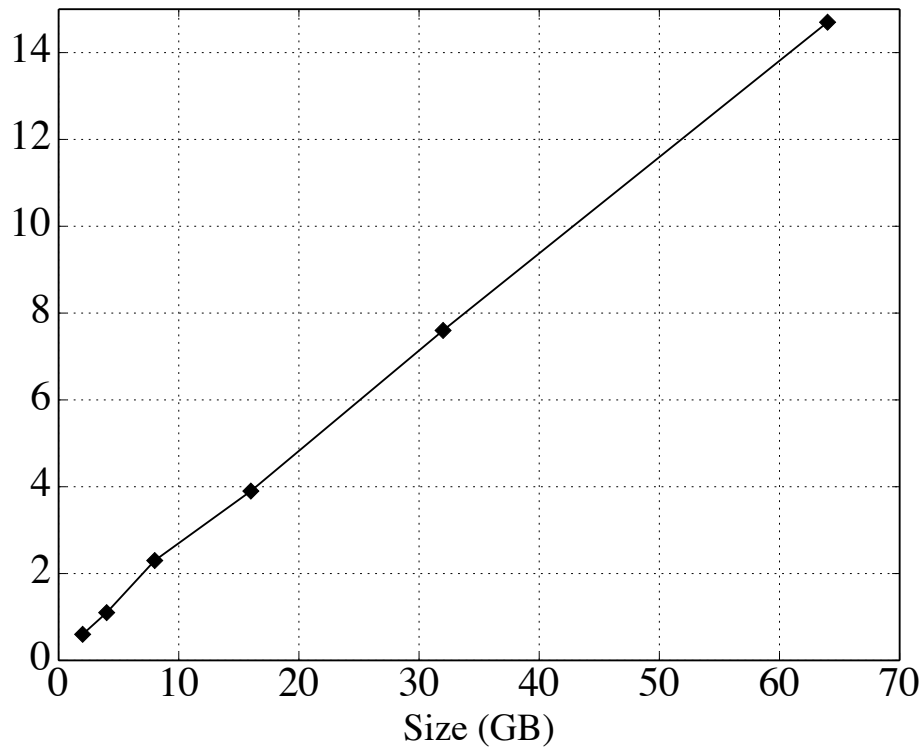


LINCS | Storage Overhead

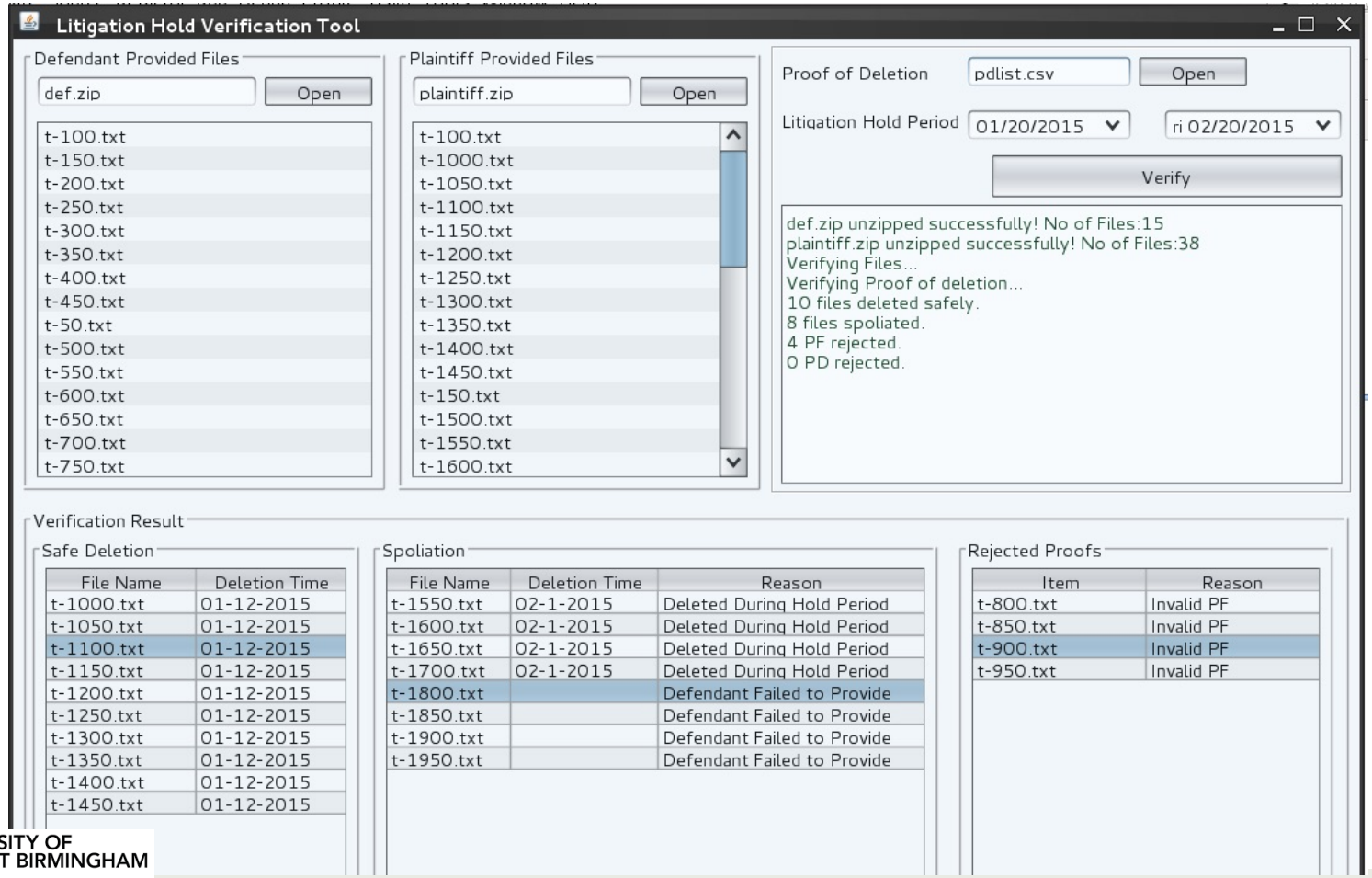


LINCS | Verification Performance





LINCS | Verification Tool



Conclusion & Future Work

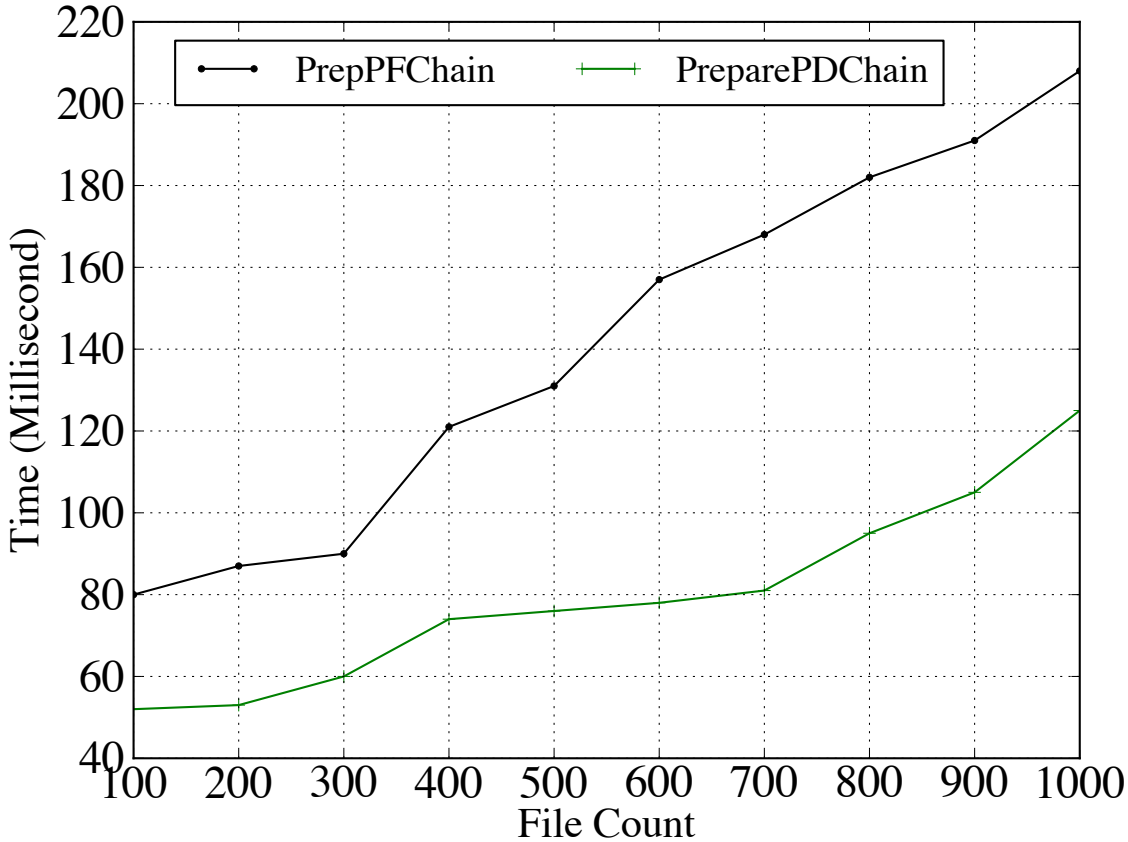
- Defined a model of trustworthy litigation holds in clouds
- Proposed *LINCS* that ensures trustworthy management of litigation holds in a cloud storage.
- Future Plan:
  - Include dynamic behavior of the cloud storage,
  - New files creation after Ts,
  - Security of the special types of file, such as MBOX or EML .

Thank You

Shams Zawoad

- zawoad@cis.uab.edu

LINCS | Proof Creation Performance



LINCS | Security | Lemma

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