Secure Deduplication Across Files

Nithin V Nath

Advisor: Dr. Bhavana Kanukurthi

Department of Computer Science and Automation
Indian Institute of Science

23-Jun-2016

Outline

Deduplication

- Large amount of data stored in cloud storage.
- Multiple users store the same file.
- Service providers need to employ space saving techniques to keep cost down.

Definition

Technique that enables storage providers to store a single copy of the data.

Deduplication in Action

- Alice uploads a file M to the server S.
- Bob requests to upload his copy of the same file M to S.
- The server identifies that M is already stored and simply updates the metadata associated with M to show that the file is owned by both Alice and Bob.
- Make this an image

Secure Deduplication

- Deduplication along with privacy is a conflicting idea
- Users would like their data to be encrypted
- Storage providers would like to identify the file uploaded by user to enable deduplication.

Motivation

- Photos taken one after the other are often *almost* identical to each other.
- These multiple files are not supported in traditional file level deduplication.
- **Challenge**: Identify that plaintexts underneath these ciphertexts are close to each other and store only the difference.

Problem Statement

-

Related Work

- Convergent Encryption (year?)
- Message Locked Encryption
- Interactive Message Locked Encryption

Convergent Encryption

- Deterministic cryptosystem that produces identical ciphertext files from identical plaintext files.
- $K = H(M)$
- $C = E(K, M)$
- $M = D(K, C)$

Message Locked Encryption

- A cryptographic primitive. MLE $= (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$.
- $\mathcal{K}_P(M)$ derives the key from the message.
- $\mathcal{T}$ is the tag generation algorithm.
- Semantic security cannot be achieved using MLE.
- [Include an image]

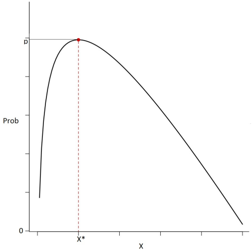
Interactive Message Locked Encryption

- Extension of *MLE*.
- Uses interaction.
- Defined using one algorithm and three protocols
  1. Init(1^λ) - The initialization algorithm.
  2. Reg - Register a client with the server.
  3. Put(M, σ_C) - Puts a plaintext M and returns f, an identifier
  4. Get(f, σ_C) - Fetches the file f.

Entropy

- Entropy is a measure of randomness
- Min-entropy of X is the negative log of maximum predictability.

$$H_\infty(X) = -log(\max_x \Pr[X = x])$$



Statistical Distance

-

Extractors

-

Deterministic Encryption

- $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$
- $c \leftarrow \mathsf{E}(1^\lambda, k, m)$
- $m \leftarrow \mathsf{D}(1^\lambda, k, c)$
- Why is this meaningful in this setting?

Error Correcting Codes

- $(\mathcal{M}, K, \tau)$-code C.
- C is a subset $\{w_0, w_1, \ldots, w_K\}$ of $\mathcal{M}$.
- $\tau > 0$ is the largest number such that there is at most one valid code word $c \in C$ for a message w such that $\text{dis}(w, c) \leq \tau$.
- Enc - The map from i to w_i.
- Dec - The map that finds, given w, the $c \in C$ such that $\text{dis}(w, c) \leq \tau$

Collision Resistant Hash Functions

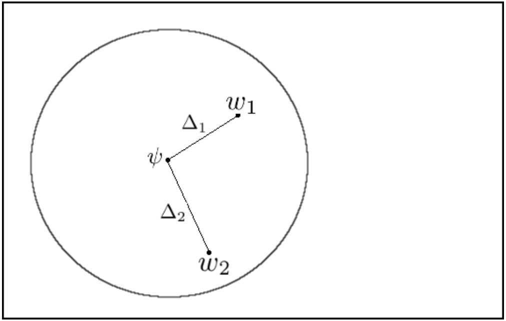
- $\mathcal{H} : \{0,1\}^n \rightarrow \{0,1\}^m$
- Collision resistant if
  - $m < n$ and
  - for all PPT$\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all security parameters $\lambda \in \mathbb{N}$,

$$\Pr[(x_0, x_1) \leftarrow \mathcal{A}(1^\lambda, \mathcal{H}) : x_0 \neq x_1 \wedge \mathcal{H}(x_0) = \mathcal{H}(x_1)] \leq \mathsf{negl}(\lambda)$$

- Family of hash functions: $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$

Our Work

- DD – Across (deduplication across files) which enables deduplication even for files that are close to each other.



Road-map

- Setting
- Adversarial Model
- Privacy Games
- DD – Across construction
- DD – Across proof

Setting

- An honest-but-curious server.
- A set of clients.
- $\mathcal{A}$ can control a subset of these clients.
- Formally modelled using a game G.
- G sets up and controls an instance of a server.

Adversarial Model

- Adversary $\mathcal{A}$ is invoked with access to a set of procedures.
- MSG procedure allows adversary to set up multiple clients and to send arbitrary messages to the server.
- INIT procedure starts protocol instances on behalf of a legitimate client L, using inputs chosen by A.
- STEP procedure advances a protocol instance by running the next step algorithm.
- STATE procedure returns the server's state - including stored ciphertexts, public parameters, etc. Only read only access is gained using this.

Games

-

DD-Across Ingredients

- A metric space $(\mathcal{M}, \text{dis})$ with hamming distance as the distance metric.
- An (l, m, κ, ϵ)-strong extractor.
- An error-correcting code $C = (\mathcal{M}, K, \tau)$.
- A collision resistant hash function family $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$.
- $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ denotes a symmetric encryption scheme.

DD-Across Construction

- $\mathrm{DD-Across}[C, \mathrm{H}, \mathrm{SE}]$.
- Server maintains 3 tables
  - **fil**: which contains the encryptions of the files uploaded by the clients.
  - **delt**: which stores the Δ.
  - **own**: which stores the ownership information.

DD-Across Construction - Init

Init

$S \leftarrow_{\$} \{0,1\}^{s(\lambda)}$

$K_h \leftarrow_{\$} \mathcal{HK}(1^{\lambda})$

$p = (S||K_h)$

$\mathbf{U} \leftarrow \phi$

$\mathbf{fil} \leftarrow \phi$; $\mathbf{delt} \leftarrow \phi$

$\mathbf{own} \leftarrow \phi$

Ret $\sigma_S = (p, \mathbf{U}, \mathbf{fil}, \mathbf{delt}, \mathbf{own})$

DD-Across Construction - Reg

Reg		
Reg[1](ϵ)		**Reg[2](σ_S)**
	$\xrightarrow{\epsilon}$	$u \leftarrow_{\$} \{0,1\}^{\lambda} \setminus \mathbf{U}$
		$\mathbf{U} \leftarrow \mathbf{U} \cup \{u\}$
	$\xleftarrow{(u, p)}$	
Ret $\sigma_c = (u, p)$		

DD-Across Construction - Put

Put

Put[1]((u,p),m)		**Put[2]**(σ_S)
$\psi \leftarrow \mathsf{Dec}(m)$		
$k \leftarrow Ext_\lambda(S, \psi)$		
$C_\psi \leftarrow Enc_{S\|\|K_h}(k, \psi)$		
$\Delta \leftarrow \mathsf{Diff}(\psi, m)$		
	$\xrightarrow{u, C_\psi, \Delta}$	
		$t_1 \leftarrow \mathcal{H}(K_h, C_\psi)$
		$t_2 \leftarrow \mathcal{H}(K_h, \Delta)$
		$t = (t_1, t_2)$
		$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$
		$\mathsf{SiffE}(\mathbf{delt}, t_2, \Delta)$
		$\mathsf{SiffE}(\mathbf{own}, (u, t), 1)$
	$\xleftarrow{\text{t}}$	
Ret (t, k)		

DD-Across Construction - Get

Get

Get[1]((u,p),t,k) | **Get[2](σ_S)**

$$\xrightarrow{u, t}$$

$C_\psi \leftarrow \textbf{fil}[t_1]$

$\Delta \leftarrow \textbf{delt}[t_2]$

$o \leftarrow \textbf{own}[u, t]$

if $o = \perp$ **then**

$C_\psi = \perp$

$\Delta = \perp$

$$\xleftarrow{C_\psi, \Delta}$$

if $C_\psi = \perp$ **then** Ret $\perp$; **fi**

$\psi \leftarrow Dec_{S||K_h}(k, C_\psi)$

$m \leftarrow \mathsf{Comb}(\psi, \Delta)$

Ret m

DD-Across Recovery

- Recovery is guaranteed.
- For $\mathcal{A}$ to win, a mismatch in the plaintext m put on the server and the plaintext m' recovered using Get.
- Immutability of the tables means once put, it cannot be changed.
- Only possibility is hash collision.

DD-Across Privacy

Definition

The error-correcting code $C = (\mathcal{M}, K, \tau)$ is said to be compatible with a source S with min-entropy $\mu(\lambda)$ iff $2^{\mu(\lambda)-\tau}$ is negligible.

Theorem

If $\mathcal{E}$ is CPA-secure and the code $C = (\mathcal{M}, K, \tau)$ is compatible with the source S, then $\mathrm{DD} - \mathrm{Across}_{RO}[\mathcal{E}, C]$ [a] *is* PRIV*-secure.*

[a] $\mathrm{DD} - \mathrm{Across}_{RO}$ is the ROM analogue of $\mathrm{DD} - \mathrm{Across}$ which models H as a random oracle

DD-Across Privacy Hybrids

Put

$\mathbf{Put[1]}$((u,p),m)		$\mathbf{Put[2]}(\sigma_S)$
$\psi \leftarrow \mathsf{Dec}(m)$		
$\underline{k \leftarrow_\$ \{0,1\}^{\kappa(\lambda)}}$		
$C_\psi \leftarrow \mathcal{E}_{S\|\|K_h}(k,\psi)$		
$\Delta \leftarrow \mathsf{Diff}(\psi, m)$		
	$\xrightarrow{u, C_\psi, \Delta}$	
		$t_1 \leftarrow \mathsf{RO}(K_h\|\|C_\psi)$
		$t_2 \leftarrow \mathsf{RO}(K_h\|\|\Delta)$
		$t = (t_1, t_2)$
		$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$
		$\mathsf{SiffE}(\mathbf{delt}, t_2, C_\Delta)$
		$\mathsf{SiffE}(\mathbf{own}, (u,t), 1)$
	$\xleftarrow{\text{t}}$	
Ret (t, k)		

DD-Across Privacy Hybrids

Put		
$\mathbf{Put[1]}((\mathrm{u,p}),\mathrm{m})$		$\mathbf{Put[2]}(\sigma_S)$
$\psi \leftarrow \mathsf{Dec}(m)$		
$k \leftarrow_{\$} \{0,1\}^{\kappa(\lambda)}$		
$C_\psi \leftarrow \mathcal{E}_{S\|\|K_h}(k,\psi)$		
$\Delta \leftarrow \mathsf{Diff}(\psi, m)$		
	$\xrightarrow{u, C_\psi, \Delta}$	
		$\underline{K_h \leftarrow_{\$} \{0,1\}^?}$
		$\underline{C_\psi \leftarrow_{\$} \{0,1\}^{\ell(\lambda)}}$
		$t_1 \leftarrow \mathsf{RO}(K_h\|\|C_\psi)$
		$t_2 \leftarrow \mathsf{RO}(K_h\|\|\Delta)$
		$t = (t_1, t_2)$
		$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$
		$\mathsf{SiffE}(\mathbf{delt}, t_2, \Delta)$
		$\mathsf{SiffE}(\mathbf{own}, (u,t), 1)$
	$\xleftarrow{\mathrm{t}}$	
Ret (t,k)		

Figure: The Put protocol in game H_3

DD-Across Privacy Hybrids

Put		
Put[1]((u,p),m)		**Put[2]**(σ_S)
$\underline{\psi \leftarrow_{\$} \{0,1\}^{\ell(\lambda)}}$		
$k \leftarrow_{\$} \{0,1\}^{\kappa(\lambda)}$		
$C_\psi \leftarrow \mathcal{E}_{S\|\|K_h}(k,\psi)$		
$\Delta \leftarrow \mathsf{Diff}(\psi, m)$		
	$\xrightarrow{u, C_\psi, \Delta}$	
		$K_h \leftarrow_{\$} \{0,1\}^?$
		$C_\psi \leftarrow_{\$} \{0,1\}^{\ell(\lambda)}$
		$t_1 \leftarrow \mathsf{RO}(K_h\|\|C_\psi)$
		$t_2 \leftarrow \mathsf{RO}(K_h\|\|\Delta)$
		$t = (t_1, t_2)$
		$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$
		$\mathsf{SiffE}(\mathbf{delt}, t_2, \Delta)$
		$\mathsf{SiffE}(\mathbf{own}, (u,t), 1)$
	$\xleftarrow{\text{t}}$	
Ret (t, k)		

Figure: The Put protocol in game H_4

Summary

- The first main message of your talk in one or two lines.
- The second main message of your talk in one or two lines.
- Perhaps a third message, but not more than that.

Open Problems and Future Work

-

Blocks

Block Title

You can also highlight sections of your presentation in a block, with it's own title

Theorem

There are separate environments for theorems, examples, definitions and proofs.

Example

Here is an example of an example block.

For Further Reading I

A. Author.
Handbook of Everything.
Some Press, 1990.

S. Someone.
On this and that.
Journal of This and That, 2(1):50–100, 2000.