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

1. Introduction
   - Secure Deduplication
   - Preliminaries
   - Contributions
   - Road-map

2. Construction
   - DD-Across

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

-

How to achieve Secure Deduplication

- **Key Idea**: Derive the key from the message itself.
- Generate a "tag" from the ciphertext.
- Compare the tags of different ciphertexts to see if they are the same.
- We can achieve security only for unpredictable data.

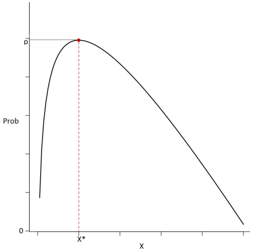
Related Work - Interactive Message Locked Encryption

- 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.

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



Extractors

-

Source

- Formalizing the notion of unpredictability.
- $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow S(1^\lambda, d)$ where $d \in \{0,1\}^*$.
- All components of $\mathbf{m}_0$ and $\mathbf{m}_1$ are unique.
- $|\mathbf{m}_0| = |\mathbf{m}_1| = m(\lambda)$.

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

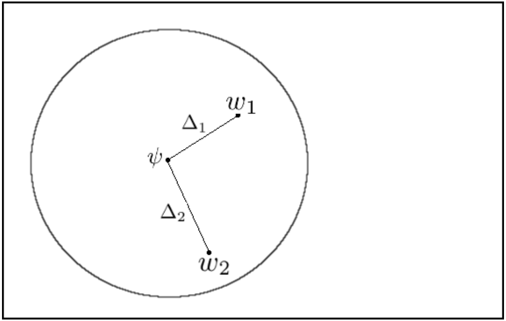
- $\mathcal{H} : \{0,1\}^n \rightarrow \{0,1\}^m$
- Collision resistant if
  - $m < n$ and
  - $\forall$PPT$\mathcal{A}$, $\exists$ a negligible function $\mathsf{negl}(\lambda)$ such that $\forall$ security parameters $\lambda \in \mathbb{N}$,

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

- 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 oracle access to the following:
  - $\textsc{Msg}()$: allows adversary to set up multiple clients and to send arbitrary messages to the server.
  - $\textsc{Init}()$: starts protocol instances on behalf of a legitimate client L, using inputs chosen by A.
  - $\textsc{Step}()$: advances a protocol instance by running the next step algorithm.
  - $\textsc{State}()$: returns the server's state - including stored ciphertexts, public parameters, etc.

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

- $\mathsf{DD-Across}[\mathcal{C}, \mathsf{H}, \mathsf{SE}]$.
- 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

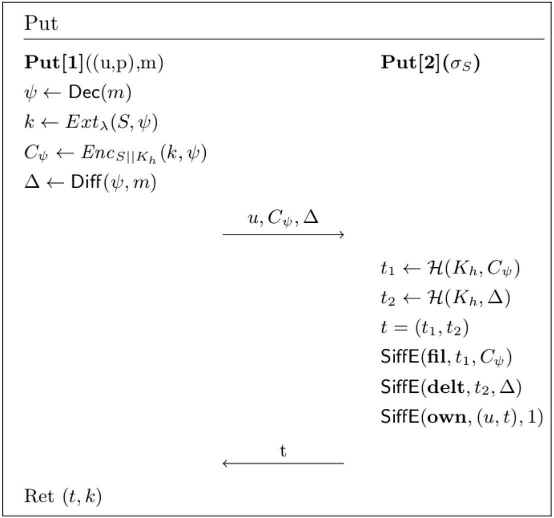
```
Init
$S \leftarrow_{\$} \{0,1\}^{s(\lambda)}$
$K_h \leftarrow_{\$} \mathcal{HK}(1^\lambda)$
$p = (S||K_h)$
$\mathbf{U} \leftarrow \phi$
$\mathbf{fil} \leftarrow \phi; \mathbf{delt} \leftarrow \phi$
$\mathbf{own} \leftarrow \phi$
Ret $\sigma_S = (p, \mathbf{U}, \mathbf{fil}, \mathbf{delt}, \mathbf{own})$
```

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)

$\psi \leftarrow \mathsf{Dec}(m)$

$k \leftarrow Ext_{\lambda}(S, \psi)$

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

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

$\xrightarrow{u, C_{\psi}, \Delta}$

Put[2](σ_S)

$t_1 \leftarrow \mathcal{H}(K_h, C_{\psi})$

$t_2 \leftarrow \mathcal{H}(K_h, \Delta)$

$t = (t_1, t_2)$

$\mathsf{SiffE}(\mathbf{fil}, t_1, C_{\psi})$

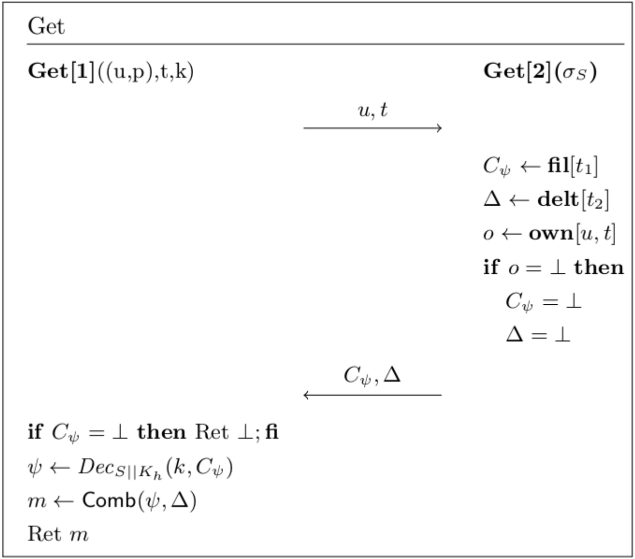
$\mathsf{SiffE}(\mathbf{delt}, t_2, \Delta)$

$\mathsf{SiffE}(\mathbf{own}, (u, t), 1)$

$\xleftarrow{\text{t}}$

Ret (t, k)

DD-Across Construction - Get



Get

Get[1]((u,p),t,k)

Get[2](σ_S)

$$\xrightarrow{u, t}$$

```
$C_\psi \leftarrow \mathbf{fil}[t_1]$
$\Delta \leftarrow \mathbf{delt}[t_2]$
$o \leftarrow \mathbf{own}[u, t]$
**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", $m_{\text{put on server}} \neq m_{\text{retrieved from server}}$
- Immutability of the tables means once put, file cannot be changed.
- Reduces to the security of hash collision.

DD-Across Privacy

Definition

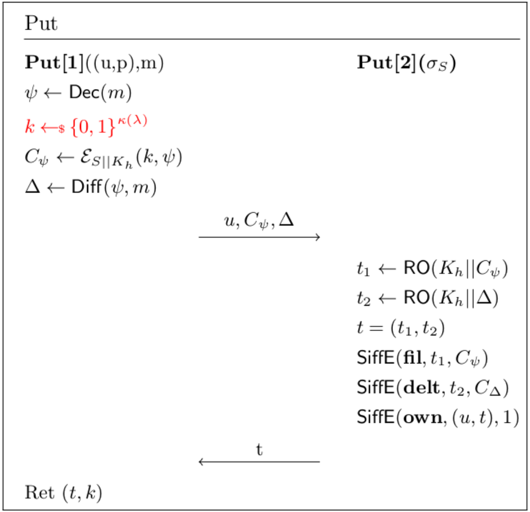
The error-correcting code $C = (\mathcal{M}, K, \tau)$ is said to be compatible with a source S with min-entropy $\mu(\lambda)$ iff $2^{\mu(\lambda)-\tau}$ is negligible.

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

Put[1]((u,p),m)

$\psi \leftarrow \mathsf{Dec}(m)$

$k \leftarrow_{\$} \{0,1\}^{\kappa(\lambda)}$

$C_\psi \leftarrow \mathcal{E}_{S||K_h}(k, \psi)$

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

$\xrightarrow{u, C_\psi, \Delta}$

Put[2](σ_S)

$t_1 \leftarrow \mathsf{RO}(K_h||C_\psi)$

$t_2 \leftarrow \mathsf{RO}(K_h||\Delta)$

$t = (t_1, t_2)$

$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$

$\mathsf{SiffE}(\mathbf{delt}, t_2, C_\Delta)$

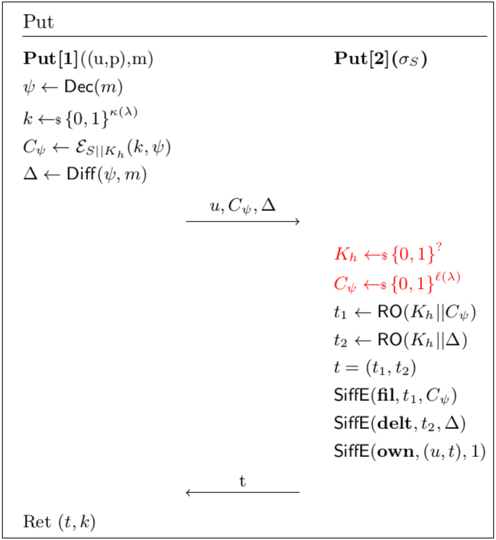
$\mathsf{SiffE}(\mathbf{own}, (u,t), 1)$

$\xleftarrow{\text{t}}$

Ret (t, k)

Figure: The Put protocol in game H_2

DD-Across Privacy Hybrids



Put

Put[**1**]((u,p),m)

$\psi \leftarrow \mathsf{Dec}(m)$

$k \leftarrow_{\$} \{0,1\}^{\kappa(\lambda)}$

$C_\psi \leftarrow \mathcal{E}_{S||K_h}(k,\psi)$

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

$\xrightarrow{u, C_\psi, \Delta}$

Put[**2**](σ_S)

$K_h \leftarrow_{\$} \{0,1\}^{?}$

$C_\psi \leftarrow_{\$} \{0,1\}^{\ell(\lambda)}$

$t_1 \leftarrow \mathsf{RO}(K_h||C_\psi)$

$t_2 \leftarrow \mathsf{RO}(K_h||\Delta)$

$t = (t_1, t_2)$

$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$

$\mathsf{SiffE}(\mathbf{delt}, t_2, \Delta)$

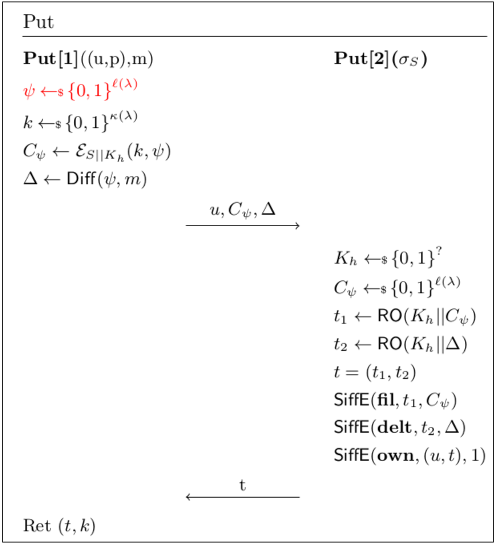
$\mathsf{SiffE}(\mathbf{own}, (u,t), 1)$

$\xleftarrow{\text{t}}$

Ret (t,k)

Figure: The Put protocol in game H_3

DD-Across Privacy Hybrids



Put

Put[1]((u,p),m)

$\psi \leftarrow_{\$} \{0,1\}^{\ell(\lambda)}$

$k \leftarrow_{\$} \{0,1\}^{\kappa(\lambda)}$

$C_\psi \leftarrow \mathcal{E}_{S||K_h}(k, \psi)$

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

$\xrightarrow{u, C_\psi, \Delta}$

Put[2](σ_S)

$K_h \leftarrow_{\$} \{0,1\}^{?}$

$C_\psi \leftarrow_{\$} \{0,1\}^{\ell(\lambda)}$

$t_1 \leftarrow \mathsf{RO}(K_h||C_\psi)$

$t_2 \leftarrow \mathsf{RO}(K_h||\Delta)$

$t = (t_1, t_2)$

$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$

$\mathsf{SiffE}(\mathbf{delt}, t_2, \Delta)$

$\mathsf{SiffE}(\mathbf{own}, (u, t), 1)$

$\xleftarrow{\text{t}}$

Ret (t, k)

Figure: The Put protocol in game H_4

Open Problems and Future Work

- DD – Across allows deduplication across files when the files map to same code-word.
- Connection of Fuzzy Extractors with the existing scheme.
- Implementing the scheme to record real world performance gains.

Thank you

- Questions?