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

Outline

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 I

- Photos taken one after the other are often *almost* identical to each other.

Motivation II

- 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:** Achieve deduplication across files, which are close to each other, in a privacy preserving way.

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. $\text{Init}(1^\lambda)$ - The initialization algorithm.
  2. Reg - Register a client with the server.
  3. $\text{Put}(M, \sigma_C)$ - Puts a plaintext M and returns f, an identifier
  4. $\text{Get}(f, \sigma_C)$ - Fetches the file f.

Outline

Entropy

- Entropy is a measure of randomness

- Min-entropy of X is the negative log of maximum predictability.

$$H_{\infty}(X) = -log(\max_{x} \Pr[X = x])$$
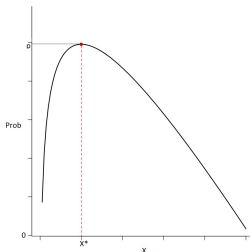
- Guessing Probability of X is $GP(X) = 2^{-H_{\infty}(X)}$

Extractors

- A family of extractors $\mathsf{Ext} = \{\mathsf{Ext}_\lambda\}$

- $\mathsf{Ext}_\lambda : \{0,1\}^{s(\lambda)} \times \{0,1\}^{l(\lambda)} \to \{0,1\}^{\kappa(\lambda)}$
  - s - seed length
  - l - input length
  - κ - output length

- (l, m, κ, ϵ)-strong extractor $\Rightarrow \forall$ min-entropy m distributions W on $\{0,1\}^l$

$$\mathsf{SD}((\mathsf{Ext}(W; X), X), (U_\kappa, X)) \leq \epsilon$$

where X is uniform on $\{0,1\}^s$

Source

- The source S is an algorithm: $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow S(1^\lambda, d)$ where $d \in \{0, 1\}^*$.

- All components of the tuples $\mathbf{m}_0$ and $\mathbf{m}_1$ are unique.

- $|\mathbf{m}_0| = |\mathbf{m}_1| = m(\lambda)$.

- Guessing probability of source:

$$\mathbf{GP}_S(\lambda) = max_{i,b,d}(\mathbf{GP}(\mathbf{m}_b[i]))$$

- We require $\mathbf{GP}_S(\lambda)$ to be negligible.

Deterministic Encryption

- $SE = (E, D)$

- $c \leftarrow E(1^\lambda, k, m)$

- $m \leftarrow D(1^\lambda, k, c)$

- **Correctness:** $D(1^\lambda, k, E(1^\lambda, k, m)) = m$ $\forall$ plaintexts $m \in \{0,1\}^*$ $\forall$ keys $k \in \{0,1\}^{\kappa(\lambda)}$ $\forall \lambda \in \mathbb{N}$.

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 $\text{dis}(w, c) \leq \tau$
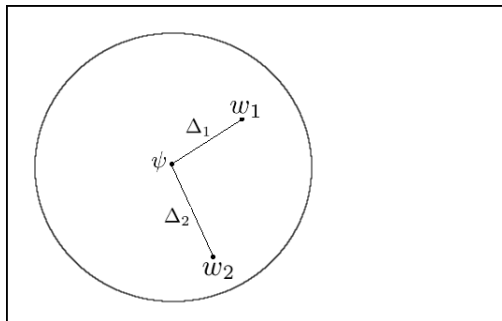
Collision Resistant Hash Functions

- $\mathcal{H} : \{0,1\}^n \to \{0,1\}^m$

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

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

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

Outline

Our Work

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

Outline

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.

The recovery game REC

Challenger

$win \leftarrow \mathit{False}$
$\sigma_S \leftarrow\!\!\$\ \mathsf{Init}(1^\lambda)$

Adversary

$\mathrm{REG}()//\mathsf{Set\ up\ a\ legitimate\ client}$
$\mathrm{INIT}()$
$\mathrm{STEP}()$
$\mathrm{MSG}()$
$\mathrm{STATE}()$
$win \leftarrow \mathrm{WINCHECK}()$

$\downarrow$

win

The privacy game PRIV

Challenger

$b \leftarrow_\$ \{0, 1\}$
$\sigma_S \leftarrow_\$ \mathsf{Init}(1^\lambda)$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow S(1^\lambda, \epsilon)$
Ret $b = b'$

Adversary

$\mathrm{REG}()$
$\mathrm{PUT}(i)$
$\mathrm{STEP}()$
$\mathrm{MSG}()$
$\mathrm{STATE}()$

$\downarrow$

b'

Outline

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 $\text{H} = (\mathcal{HK}, \mathcal{H})$.

- $\text{SE} = (\text{E}, \text{D})$ denotes a symmetric encryption scheme.

DD-Across Construction

- DD − Across[C, H, 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.

Init

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

Reg

Reg[1](ϵ)	**Reg[2](σ_S)**

$$\xrightarrow{\quad \epsilon \quad}$$

$u \leftarrow_\$ \{0,1\}^\lambda \setminus \mathbf{U}$

$\mathbf{U} \leftarrow \mathbf{U} \cup \{u\}$

$$\xleftarrow{\quad (u,p) \quad}$$

Ret $\sigma_c = (u, p)$
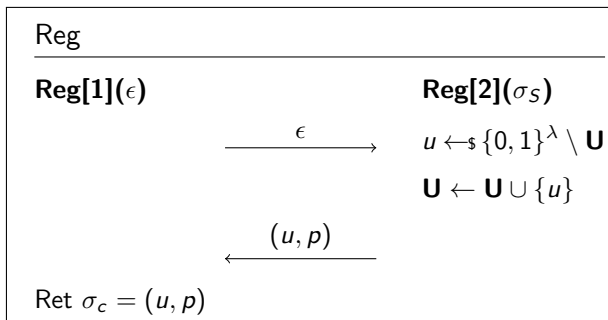
DD-Across Construction - Put

Put

Put[1]((u,p),m)   **Put[2]**(σ_S)

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

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

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

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

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

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

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

$t = (t_1, t_2)$

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

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

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

$$\xleftarrow{\quad t \quad}$$

Ret (t, k)

DD-Across Construction - Get

Get
***

Get[1]((u,p),t,k)   **Get[2](σ_S)**

$$\xrightarrow{\quad u, t \quad}$$

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

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

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

if $o = \bot$ **then**

     $C_\psi = \bot$

     $\Delta = \bot$

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

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

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

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

Ret m

Outline

- 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
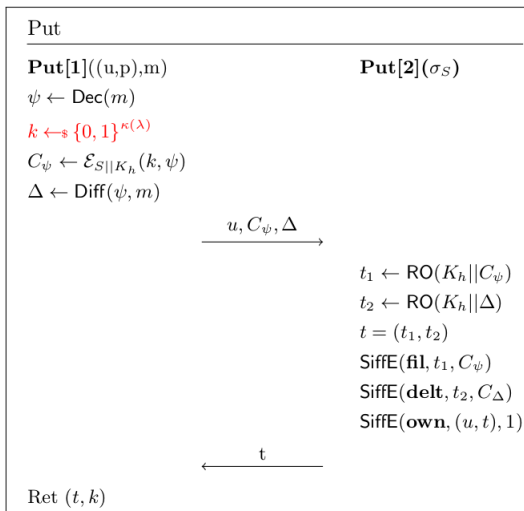
Put

Put[1]$((u,p),m)$	**Put[2]**(σ_S)
$\psi \leftarrow \mathsf{Dec}(m)$	
$k \leftarrow_\$ \{0,1\}^{\kappa(\lambda)}$	
$C_\psi \leftarrow \mathcal{E}_{S||K_h}(k, \psi)$	
$\Delta \leftarrow \mathsf{Diff}(\psi, m)$	

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

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

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

$t = (t_1, t_2)$

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

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

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

$$\xleftarrow{\quad t \quad}$$

Ret (t, k)

Figure: The Put protocol in game H_2

Figure: The Put protocol in game H_3

Put

Put[1]$((u,p),m)$	**Put[2]**(σ_S)
$\psi \leftarrow_\$ \{0,1\}^{\ell(\lambda)}$	
$k \leftarrow_\$ \{0,1\}^{\kappa(\lambda)}$	
$C_\psi \leftarrow \mathcal{E}_{S||K_h}(k, \psi)$	
$\Delta \leftarrow \mathsf{Diff}(\psi, m)$	

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

$K_h \leftarrow_\$ \{0,1\}^?$
$C_\psi \leftarrow_\$ \{0,1\}^{\ell(\lambda)}$
$t_1 \leftarrow \mathsf{RO}(K_h||C_\psi)$
$t_2 \leftarrow \mathsf{RO}(K_h||\Delta)$
$t = (t_1, t_2)$
$\mathsf{SiffE}(\mathbf{fil}, t_1, C_\psi)$
$\mathsf{SiffE}(\mathbf{delt}, t_2, \Delta)$
$\mathsf{SiffE}(\mathbf{own}, (u,t), 1)$

$$\xleftarrow{\quad t \quad}$$
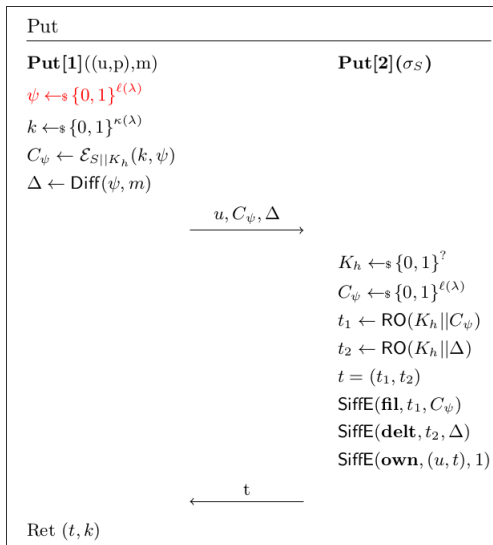
Ret (t, k)
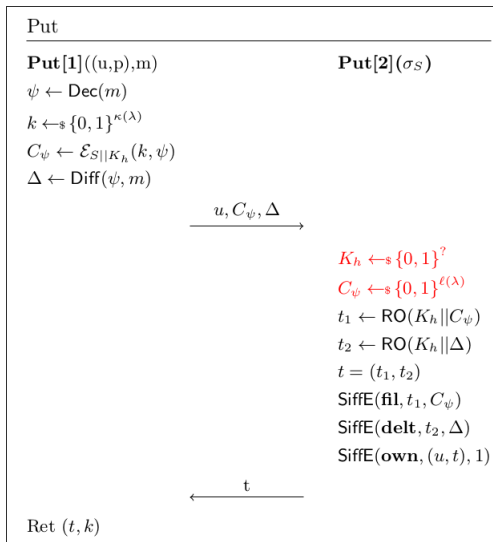
Figure: The Put protocol in game H_4

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.

For Further Reading I

📄 Mihir Bellare, Sriram Keelveedhi, Thomas Ristenpart
Message-Locked Encryption and Secure Deduplication.
Advances in Cryptology EUROCRYPT, 2013.

📄 Mihir Bellare, Sriram Keelveedhi
Interactive Message-Locked Encryption and Secure
Deduplication.
Public Key Cryptography EUROCRYPT, 2015.

Thank you

- Thank You