Secure indexes and other oblivious search structures

(Privaatne otsing: indeksid ning alternatiivid)

Sven Laur
swen@math.ut.ee

Helsinki University of Technology

Basic motivation

Secure storage problem

- Client Alice does not have skills for data protection.
- Service provider Bob offers:
  - easy access,
  - long-term integerity protection.
- However, Bob can expose data to third parties.
- Alice needs a system to securely store, retrieve, alter and search data.

Desired and achievable features

- Encryption of stored documents provides confidenciality.
- Access patterns of documents remains unhidden.
  - Bob learns which documents are retrieved.
  - Bob learns which documents are modified.
- Additional structures allow keyword search over encrypted documents.
  - Search structure is generated by Alice.
  - Only Alice can start the search.
  - The search query is relatively short.
  - Most of computations are done by Bob.

Formal specification

KeyGen:

Given public parameters, generate the master key $\mathcal{K}$.

MakeTrapdoor:

Given a word $w \in \mathcal{S}$ and $\mathcal{K}$, compute a trapdoor T_w.

BuildIndex:

Given a collection of words $W \subseteq \mathcal{S}$ and $\mathcal{K}$, compute index I_W.

SearchIndex:

Given a trapdoor T_w for a word $w \in \mathcal{S}$ and an index I_W, determine whether w belongs to W or not, i.e. return 1 for $w \in W$ and 0 otherwise.

Informal security requirements

- Bob should learn only search results.
- Indices of similar documents should look uncorrelated.
- It must be hard to generate new trapdoors from revealed ones.
- It must be hard to reconstruct the keyword from trapdoor.
- The system should remain secure even if Bob has total control over the content of indices.

Formal security game (1)

Setup Phase

Adversary chooses public parameters of the secure index system.

Challenger runs the $\textsc{KeyGen}$ algorithm with the selected parameters and obtains the master key $\mathcal{K}$.

Query Phase

Adversary can adaptively choose collections of keywords $W \subseteq \mathcal{S}$ and query corresponding indices I_W from Challenger.

Adversary can adaptively query trapdoors T_w for all $w \in \mathcal{S}$ and test whether an arbitrary index I contains w.

Formal security game (2)

Challenge Phase

Adversary chooses two word collections $W_0, W_1 \subseteq \mathcal{S}$ such that $|W_0| = |W_1|$ and no trapdoors have been queried for words $w \in W_0 \Delta W_1$.

Challenger chooses randomly $b \in \{0, 1\}$ and sends an index I_{W_b} to Adversary.

Guessing Phase

Adversary can do the same operations as on the **Query Phase** except querying the trapdoors T_w for $w \in W_0 \Delta W_1$.

Adversary should output 0 or 1.

Formal security game (3)

Definition. *Indexing scheme $\mathcal{I}$ is semantically secure if any reasonable adversary has a negligible advantage in the guessing game*

$$\mathsf{Adv}^{\mathrm{LR}}_{\mathcal{I}}(\mathcal{A}) := 2 \cdot \left|\Pr\left[\mathcal{A} \text{ outputs correct quess } \right] - \frac{1}{2}\right| < \epsilon$$

- A should complete in t timesteps.

- A can adaptively choose keywords and word collections:
  - index queries contain less than q_1 words (with repetitions);
  - less than q_2 trapdoors are revealed;
  - challenge collections W_0 and W_1 contain less than q_3 words.

All about Bloom filters

Word mask

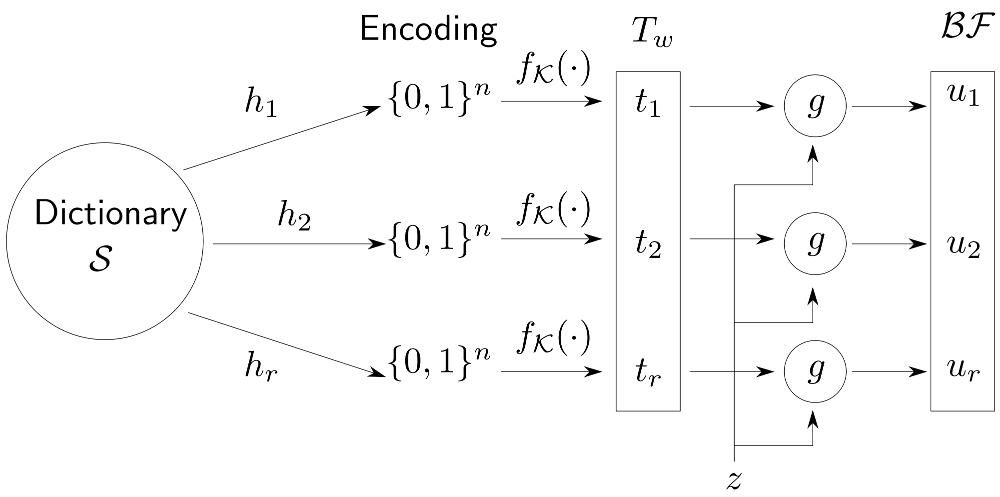
1:							•
2:		•					
3:					•		
4:			•				

Collection of words

1:	•	•					•
2:		•		•		•	
3:	•		•		•		
4:			•	•	•		

- The number of layers determines the rate of false positives.
- The bullet at each layer is chosen by a hash function.
- Bloom filter is history independent.
- Next we make Bloom filters secure.

Z-index scheme



- Collision resistant hash functions $h_1, \ldots, h_r$ are public.

- The master key $\mathcal{K}$ is used to create trapdoor vectors $T_w = (t_1, \ldots, t_r)$.

- Pseudorandom functions $g_{t_i}(\cdot)$ give correlation resistance.

Something leaks from Z-index

- If Adversary manages to find collisions $h_i(w_1) = h_j(w_2)$ for some $w_1, w_2 \in \mathcal{S}$.

- If Adversary can predict $f_{\mathcal{K}}(\cdot)$, given some freely chosen trapdoors

$$T_w = [f_{\mathcal{K}}(s_1), \ldots, f_{\mathcal{K}}(s_r)], \; s_i = h_i(w).$$

- If Adversary can predict $g_{t_i}(\cdot)$, given some freely chosen values $g_{t_i}(z)$.

- If Adversary can invert $f_{\mathcal{K}}(\cdot)$.

Correlation resistance

Let trapdoors $T_w \in \{0,1\}^n$ be chosen randomly.

- In Query Phase:
  - $\textsc{BuildIndex}$ allows to compute $g_s(z)$ for (freely chosen) z.
  - $\textsc{MakeTrapdoor}$ allows to reveal secret key s, given sequence of observed plaintext chipertext pairs $[z_1, g_s(z_1)], \ldots, [z_k, g_s(z_k)]$.

- In Challenge Phase:
  - Adversary chooses two sets of unknown keys $\{t_1, \ldots, t_\ell\}$ and $\{t'_1, \ldots, t'_\ell\}$

- In Guessing Phase:
  - Adversary must decide whether Challenger chose $\{t'_1, \ldots, t'_\ell\}$ or $\{t'_1, \ldots, t'_\ell\}$

Multi-key encryption oracle

Oracle $\mathcal{O}_g^{\mathrm{mk}}$

Commands

$$\begin{Vmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \\ \vdots \end{Vmatrix} \quad \Longleftarrow$$

$$\textsc{Fetch}(i, r) = g_{t_i}(r)$$

$$\textsc{Reveal}(i) = t_i$$

$$\textsc{Fetch}^*(i_1, \ldots, i_\lambda, r) = \begin{cases} g_{x_{i_1}}(r), \ldots, g_{x_{i_\lambda}}(r), \\ y_1, \ldots, y_\lambda \stackrel{r}{\leftarrow} \mathbb{Z}_m. \end{cases}$$

Function g is strongly indistinguishable iff

$$\mathsf{Adv}_g^{\text{s-ind}}(\mathcal{A}) := \left| \Pr\left[\mathcal{A}^{\mathcal{O}_g^{\mathrm{mk}}(1)} = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}_g^{\mathrm{mk}}(0)} = 1\right] \right| < \epsilon.$$

Putting things together

Theorem 1. [Informal] *Z-index scheme is semantically secure if*

- $h_1, \ldots, h_k$ *are collision resistant;*
- f *is a pseudorandom function;*
- g *is strongly indistinguishable.*

Theorem 2. [Informal] *If* g *is a pseudorandom function then it is also strongly indistinguishable. The security drop is almost proportional to number of observed keys.*

Shared indices. Access control

Alice and Carl want to build a summary index.

- Both of them separately should not be able to create trapdoors.

- Can be implemented with exponentation operation.

Alice allows Carl to search in the search structure.

- Carl should not be able to create trapdoors alone.

- Alice should not learn Carls queries.

- Can be implemented with homomorpic encryption.

More open questions

Usually more complex queries include AND and OR operators.
The Z-index scheme reveals results of individual queries.

- How to construct indexing scheme with AND or OR trapdoors?
  - Trivial solutions exist but they do not scale well.
- How to construct efficient oblivious indexing schemes?
  - Trivial solutions exist but they do not scale well.
- How to construct hybrid indexing schemes?
  - Extremely useful in practice.
  - No constructions are published.