

# Efficiency Lower Bounds and Optimal Constructions of Searchable Encryption

Raphael Bost<sup>1,2</sup>

<sup>1</sup>Direction Générale de l'Armement - Maîtrise de l'Information

<sup>2</sup>Université de Rennes 1

ESSA2, 10/07/2018, Bertinoro

# Security vs. Efficiency

Searchable encryption is all about a  
security-performance tradeoff

Nothing comes for free. Ever!

# Efficiency

Many possible measurements:

- Computational complexity
- Communication complexity
- Number of interactions
- Size of the encrypted database
- Size of the client's state
- Memory locality & read efficiency

# Security

We can evaluate the security

- formally: from the leakage in the security proofs
- practically: from actual attacks (e.g. leakage-abuse attacks)

# This presentation

Lower bounds on the efficiency of:

- static searchable encryption schemes hiding the repetition of search queries;
- dynamic searchable encryption schemes with forward-private updates;
- dynamic searchable encryption schemes secure against malicious adversaries.

# This presentation

We restricted ourselves to:

- symmetric searchable encryption (SSE)
- single-keyword search queries
- database structure: atomic keyword/document pairs (a.k.a. entries)

# Security model

- Indistinguishability-based security definition: two executions with the same leakage cannot be distinguished by an adversary
- Only the non-adaptive version of the definition is needed here

# Notations

- $N = |\text{DB}|$ : total number of entries
- $K$ : number of distinct keywords
- $|\text{DB}(w)| = n_w$ : number of entries matching  $w$
- $a_w$ : number of entries matching  $w$  inserted in the database
- $H = (\text{DB}, r_1, \dots, r_i)$ : query history ( $r_i$  can be a search query, or an update query)



# Schemes hiding the search pattern

- Static schemes only revealing the number of results of a query (hides the repetition of queries — the search pattern)
- Related to ORAM (# results of each query is 1)  
Called File-ORAM in [ACN<sup>+</sup>17]
- ORAM lower bound [GO96]:  $\Omega\left(\frac{\log N}{\log \sigma}\right)$

# Lower bound on search-pattern-hiding SSE

## Theorem

*Let  $\Sigma$  be a static SSE scheme leaking  $(N, K)$  and  $|\text{DB}(w)|$ . Then the complexity of the search protocol is*

$$\Omega\left(\frac{\log\left(\frac{\bar{N}(H,w)}{n_w}\right)}{\log|\sigma| \cdot \log\log\left(\frac{\bar{N}(H,w)}{n_w}\right)}\right)$$

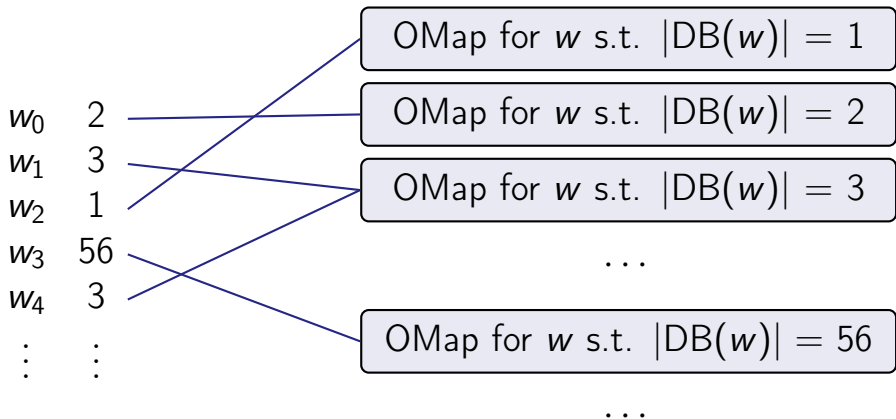
*where*

$$\bar{N}(H, w) = |\text{DB}| - \sum_{\substack{j=1 \\ |\text{DB}(w_j)| \neq |\text{DB}(w)|}}^i |\text{DB}(w_j)|.$$

# Explanations

- Suppose the client queries  $w$  and  $w'$  with  $|\text{DB}(w)| \neq |\text{DB}(w')|$ . The adversary knows from the leakage that  $w \neq w'$ .
- As  $w \neq w'$ , the adversary knows that the accessed entries will be different. Hence the term in  $\overline{N}$ .
- The order in which the entries are touched does not matter. Hence the binomial coefficient.
- The proof essentially proceeds as in [GO96].
- The  $\log \log$  term is an artefact.

# Tightness of the lower bound



Query complexity of an OMap of size  $n$ :  $\mathcal{O}(\log n)$ .  
The search complexity of the SE construction is  $\mathcal{O}(\log K)$ .

# Tightness of the lower bound

The previous construction breaks the lower bound when  $K \ll N$  (common case).

During setup, the *profile* of the database is leaked:  $(K_i)_{i=1}$  where  $K_i = \#\{w \text{ s.t. } |\text{DB}(w)| = i\}$ .

With a small additional leakage, we can break the lower bound on SP-hiding SSE.

# Forward Privacy

## File injection attacks [ZKP16]

Leaking information about the updated keywords leads to devastating adaptive attacks.

## Forward privacy

An update does not leak any information on the updated keywords (often, no information at all)

Introduced in [SPS14], must have security feature for modern dynamic schemes

# The cost of forward privacy

| Scheme                | Computation  |                         | Client Storage          | FP |                         |
|-----------------------|--|-------------------------|-------------------------|----|-------------------------|
|                       | Search   | Update                  |                         |    |                         |
| [CJJ <sup>+</sup> 14] | $\mathcal{O}(a_w)$   | $\mathcal{O}(1)$        | $\mathcal{O}(1)$        | ✗  |                         |
| [SPS14]               | $\mathcal{O}(a_w + \log N)$<br>$\mathcal{O}(n_w \log^3 N)$ | $\mathcal{O}(\log^2 N)$ | $\mathcal{O}(N^\alpha)$ | ✓  | Supports deletions well |
| Σοφος                 | $\mathcal{O}(a_w)$   | $\mathcal{O}(1)$        | $\mathcal{O}(K)$        | ✓  | TDP                     |
| [EKPE18]              | $\mathcal{O}(a_w)$   | $\mathcal{O}(1)$        | $\mathcal{O}(K)$        | ✓  | } write during search   |
| [KKL <sup>+</sup> 17] | $\mathcal{O}(a_w)$   | $\mathcal{O}(1)$        | $\mathcal{O}(K)$        | ✓  |                         |
| Diana                 | $\mathcal{O}(a_w)$   | $\mathcal{O}(\log a_w)$ | $\mathcal{O}(K)$        | ✓  | CPRF                    |
| FAST                  | $\mathcal{O}(a_w)$   | $\mathcal{O}(1)$        | $\mathcal{O}(K)$        | ✓  |                         |

# Lower bound on forward-private SE

## Theorem

*Let  $\Sigma$  be a forward-private SSE scheme. Then the sum of the amortized complexity of the search and update protocols is*

$$\Omega\left(\frac{\log K}{\log |\sigma| \cdot \log \log K}\right)$$

## Fragile proof

There might be some issues with the proof.  
Details are important (thanks Tarik!).



# Tightness of the FP lower bound

- $\Sigma\phi\phi\sigma$ , KKLPK, EKPE and FAST show that the lower bound is tight ( $|\sigma| = K$ ).
- FAST shows that the lower bounds can be reached relying only on a PRF, without rewriting the DB during the search algorithm to 'cache' the results.
- Outsource the client's counter map using an oblivious map data structure.  
 $|\sigma| = \mathcal{O}(1)$ ,  $\mathcal{O}(\log K)$  search & update complexity.
- Open question: is there a middle point?  
e.g.  $|\sigma| = \mathcal{O}(\sqrt{K})$  &  $\mathcal{O}(1)$  update complexity.

# Verifiable Searchable Encryption

The security against malicious adversaries can be split in two parts.

## Confidentiality

No information leaks about the DB/query.  
Often simple (single interaction).

## Soundness (integrity)

The server cannot return incorrect results.  
Does not depend on the leakage.

# Memory checking

## Problem

How to outsource memory to an untrusted party, while ensuring authenticity and using limited trusted local storage?

Lower bound [DNRV'09]: a memory checker outsourcing  $n$  values, with  $|\sigma| < n^{1-\varepsilon}$  for some  $\varepsilon > 0$  has computational overhead

$$\Omega\left(\frac{\log n}{\log \log n}\right).$$

# VSSE lower bound

Using a simple reduction from memory checking, we get a lower bound on verifiable SSE schemes.

## Theorem

*Let  $\Sigma$  be a dynamic verifiable SSE scheme with  $|\sigma| < K^{1-\varepsilon}$  for some  $\varepsilon > 0$ . Then the computational complexity of the search or of the update protocol is*

$$\Omega\left(\frac{\log K}{\log \log K}\right).$$

# A practical VSSE lower bound

Using a less generic result on hash-based memory checker from [TT05], we can improve the lower bound to

$$\Omega\left(\log \frac{K}{|\sigma|}\right).$$

# Why is this interesting?

- This lower bound does not depend on the leakage.
- If a scheme, hides the search pattern, or is forward-private, we should be able to get verifiability for free:  $\Omega\left(\frac{\log K}{\log |\sigma| \cdot \log \log K}\right)$  vs.  $\Omega\left(\log \frac{K}{|\sigma|}\right)$ .
- And we can ...

# Set hash functions [CDvD<sup>+</sup>03]

Some kind of incremental hashing [BM97]:

- The value of the hash does not depend on the order
- It is easy to compute  $\mathcal{H}(A \cup \{x\})$  from  $\mathcal{H}(A)$  and  $x$ .  
More generally  $\mathcal{H}(A \cup B) = \mathcal{H}(A) +_{\mathcal{H}} \mathcal{H}(B)$
- It is easy to compute  $\mathcal{H}(A \setminus \{x\})$  from  $\mathcal{H}(A)$  and  $x$ .  
More generally  $\mathcal{H}(A \setminus B) = \mathcal{H}(A) -_{\mathcal{H}} \mathcal{H}(B)$

# Collision resistance of set hash functions

It must be hard for an adversary to find two different sets hashing to the same value.

## Definition of collision resistance

$$\text{Adv}_{\mathcal{H},A}^{\text{col}}(\lambda) = \mathbb{P}[K \xleftarrow{\$} \mathcal{K}, (S, S') \leftarrow A(K) : \\ S \neq S' \wedge \mathcal{H}_K(S) \equiv_{\mathcal{H}_K} \mathcal{H}_K(S')]$$

$\mathcal{H}$  is collision resistant if  $\text{Adv}_{\mathcal{H},A}^{\text{col}}(\lambda)$  is negligible in  $1^\lambda$ .

Efficiently instantiable using elliptic curves [MSTA16].



# Generic VSSE

Two simple ideas:

1. For each keyword  $w$ , store  $H(DB(w))$  in a table  $T$   
When searching for  $w$  and returned the result set  $R$ ,  
check that  $H(R) = T$ .  
When updating on  $w$ , update  $H(DB(w))$   
incrementally.
2. Outsource  $T$  using a verifiable oblivious map

# Generic VSSE

- Additional client storage:  $\mathcal{O}(1)$
- Additional server storage:  $\mathcal{O}(K)$
- Computational overhead:  $\mathcal{O}(\log K + |\text{DB}(w)|)$
- Additional leakage:  $K$  (from the size of the OMap)  
Can be applied to any forward-private scheme to make it verifiable

# Conclusion



- Three lower bounds showing the tradeoffs between security and efficiency
- They are very fragile. Can they be extended to a more general setting?
- Forward-private schemes: is there a lower bound on the locality? Which parameter does it involve?

Slides: <https://r.bost.fyi/slides/essa2.pdf>

# References I



-  Gilad Asharov, T-H. Hubert Chan, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi, *Oblivious computation with data locality*, Cryptology ePrint Archive, Report 2017/772, 2017, <http://eprint.iacr.org/2017/772>.
-  Mihir Bellare and Daniele Micciancio, *A new paradigm for collision-free hashing: Incrementality at reduced cost*, EUROCRYPT'97 (Walter Fumy, ed.), LNCS, vol. 1233, Springer, Heidelberg, May 1997, pp. 163–192.

# References II

-  Dwaine E. Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend, and G. Edward Suh, *Incremental multiset hash functions and their application to memory integrity checking*, ASIACRYPT 2003 (Chi-Sung Lai, ed.), LNCS, vol. 2894, Springer, Heidelberg, November / December 2003, pp. 188–207.
-  David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner, *Dynamic searchable encryption in very-large databases: Data structures*

# References III

*and implementation*, NDSS 2014, The Internet Society, February 2014.



-  Mohammad Etemad, Alptekin Küpçü, Charalampos Papamanthou, and David Evans, *Efficient dynamic searchable encryption with forward privacy*, PoPETs **2018** (2018), no. 1, 5–20.
-  Oded Goldreich and Rafail Ostrovsky, *Software protection and simulation on oblivious RAMs*, Journal of the ACM **43** (1996), no. 3, 431–473.

# References IV


-  Kee Sung Kim, Minkyu Kim, Dongsoo Lee, Je Hong Park, and Woo-Hwan Kim, *Forward secure dynamic searchable symmetric encryption with efficient updates*, ACM CCS 17 (Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, eds.), ACM Press, October / November 2017, pp. 1449–1463.
-  Jeremy Maitin-Shepard, Mehdi Tibouchi, and Diego F Aranha, *Elliptic curve multiset hash*, The Computer Journal **60** (2016), no. 4, 476–490.



# References V

-  Emil Stefanov, Charalampos Papamanthou, and Elaine Shi, *Practical dynamic searchable encryption with small leakage*, NDSS 2014, The Internet Society, February 2014.
-  Roberto Tamassia and Nikos Triandopoulos, *Computational bounds on hierarchical data processing with applications to information security*, ICALP 2005 (Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, eds.), LNCS, vol. 3580, Springer, Heidelberg, July 2005, pp. 153–165.

# References VI

-  Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou, *All your queries are belong to us: The power of file-injection attacks on searchable encryption*, 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016., 2016, pp. 707–720.