Security-Efficiency Tradeoffs in Searchable Encryption

Lower Bounds and Optimal Constructions

Raphael Bost[1] Pierre-Alain Fouque[2]

[1]Direction Générale de l'Armement - Maîtrise de l'Information

[2]Université de Rennes 1

PETS 2019, 17/07/2019, Stockholm

Searchable Encryption

- Outsource data to an untrusted server.

Searchable Encryption

- Outsource data to an untrusted server.
- Perform search operations ...

Searchable Encryption

- Outsource data to an untrusted server.
- Perform search operations ...
- ... efficiently (sublinear in the database size).

Searchable Encryption

- Outsource data to an untrusted server.
- Perform search operations ...
- ... efficiently (sublinear in the database size).
- Allow some leakage to improve performance.

Security *vs.* Efficiency

Searchable encryption is all about a security-performance tradeoff

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.

This presentation

We restricted ourselves to:

- symmetric searchable encryption (SSE)

This presentation

We restricted ourselves to:

- symmetric searchable encryption (SSE)
- single-keyword search queries

This presentation

We restricted ourselves to:

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

Notations

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

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

Security model

$H^0 = (\mathrm{DB}^0, r_1^0, \ldots, r_i^0)$ $H^1 = (\mathrm{DB}^1, r_1^1, \ldots, r_i^1)$

Security model

$H^0 = (\mathrm{DB}^0, r_1^0, \ldots, r_i^0)$ $\qquad$ $H^1 = (\mathrm{DB}^1, r_1^1, \ldots, r_i^1)$

$\Downarrow$ Leakage $\Downarrow$

$$\mathcal{L}(H^0) = \mathcal{L}(H^1)$$

Security model

$H^0 = (\mathrm{DB}^0, r_1^0, \ldots, r_i^0)$ $H^1 = (\mathrm{DB}^1, r_1^1, \ldots, r_i^1)$

Security model

$$H^0 = (\mathrm{DB}^0, r_1^0, \ldots, r_i^0) \qquad\qquad H^1 = (\mathrm{DB}^1, r_1^1, \ldots, r_i^1)$$

$$\Downarrow \quad \text{Execute} \quad \Downarrow$$

$$\tau^0 \quad \approx \quad \tau^1$$

Schemes hiding the search pattern

- Static schemes only revealing the number of results of a query (hides the repetition of queries)

$$\mathcal{L}(\mathsf{DB}, w_1, \ldots, w_i) = ((N, K), n_{w_1}, \ldots, n_{w_n})$$

Schemes hiding the search pattern

- Static schemes only revealing the number of results of a query (hides the repetition of queries)

$$\mathcal{L}(\mathrm{DB}, w_1, \ldots, w_i) = ((N, K), n_{w_1}, \ldots, n_{w_n})$$

- Related to ORAM (# results of each query is 1)
  Called File-ORAM in [ACN$^+$17]

Schemes hiding the search pattern

- Static schemes only revealing the number of results of a query (hides the repetition of queries)

$$\mathcal{L}(\mathrm{DB}, w_1, \ldots, w_i) = ((N, K), n_{w_1}, \ldots, n_{w_n})$$

- Related to ORAM (# results of each query is 1)
  Called File-ORAM in [ACN+17]
- ORAM lower bound [GO96]: $\Omega\left(\frac{\log N}{\log \sigma}\right)$

Schemes hiding the search pattern

- Expect something like:

$$\Omega\left(\frac{n_w \log N}{\log \sigma}\right) = \Omega\left(\frac{\log N^{n_w}}{\log \sigma}\right)$$

Schemes hiding the search pattern

- Expect something like:

$$\Omega\left(\frac{n_w \log N}{\log \sigma}\right) = \Omega\left(\frac{\log N^{n_w}}{\log \sigma}\right)$$

- In a single result, entries are not duplicated

$$\Omega\left(\frac{\log(N \cdot (N-1) \cdot \cdots \cdot (N-n_w))}{\log \sigma}\right)$$

Schemes hiding the search pattern

- Expect something like:

$$\Omega\left(\frac{n_w \log N}{\log \sigma}\right) = \Omega\left(\frac{\log N^{n_w}}{\log \sigma}\right)$$

- In a single result, entries are not duplicated

$$\Omega\left(\frac{\log(N \cdot (N-1) \cdot \dots \cdot (N - n_w))}{\log \sigma}\right)$$

- The order of the returned elements does not matter

$$\Omega\left(\frac{\log \binom{N}{n_w}}{\log \sigma}\right)$$

Schemes hiding the search pattern

- Suppose the client queries w and w' with $|\mathrm{DB}(w)| \neq |\mathrm{DB}(w')|$. The adversary knows from the leakage that $w \neq w'$.

Schemes hiding the search pattern

- Suppose the client queries w and w' with $|\mathsf{DB}(w)| \neq |\mathsf{DB}(w')|$. The adversary knows from the leakage that $w \neq w'$.
- As $w \neq w'$, the adversary knows that the accessed entries will be different.

Schemes hiding the search pattern

- Suppose the client queries w and w' with $|\mathrm{DB}(w)| \neq |\mathrm{DB}(w')|$. The adversary knows from the leakage that $w \neq w'$.
- As $w \neq w'$, the adversary knows that the accessed entries will be different. The number of entries to consider is

$$\overline{N}(H, w) = N - \sum_{n \in \{|\mathrm{DB}(w_j)| \neq |\mathrm{DB}(w)|\}} n.$$

Lower bound on search-pattern-hiding SSE

Theorem

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

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

where

$$\overline{N}(H,w) = |\mathrm{DB}| - \sum_{n\in\{|\mathrm{DB}(w_j)|\neq|\mathrm{DB}(w)|\}} n.$$

Tightness of the lower bound

w_0	2
w_1	3
w_2	1
w_3	56
w_4	3
$\vdots$	$\vdots$

Tightness of the lower bound

w_0	2
w_1	3
w_2	1
w_3	56
w_4	3
$\vdots$	$\vdots$

OMap for w s.t. $|\mathsf{DB}(w)| = 1$

OMap for w s.t. $|\mathsf{DB}(w)| = 2$

OMap for w s.t. $|\mathsf{DB}(w)| = 3$

$\cdots$

OMap for w s.t. $|\mathsf{DB}(w)| = 56$

$\cdots$

Tightness of the lower bound

w_0	2
w_1	3
w_2	1
w_3	56
w_4	3
$\vdots$	$\vdots$

OMap for w s.t. $|\mathrm{DB}(w)| = 1$

OMap for w s.t. $|\mathrm{DB}(w)| = 2$

OMap for w s.t. $|\mathrm{DB}(w)| = 3$

...

OMap for w s.t. $|\mathrm{DB}(w)| = 56$

...

Tightness of the lower bound

w_0	2
w_1	3
w_2	1
w_3	56
w_4	3
$\vdots$	$\vdots$

OMap for w s.t. $|\mathrm{DB}(w)| = 1$

OMap for w s.t. $|\mathrm{DB}(w)| = 2$

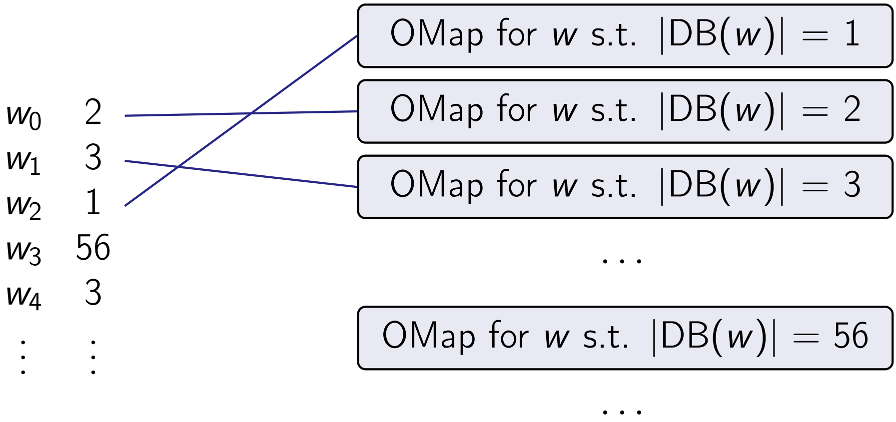
OMap for w s.t. $|\mathrm{DB}(w)| = 3$

$\dots$

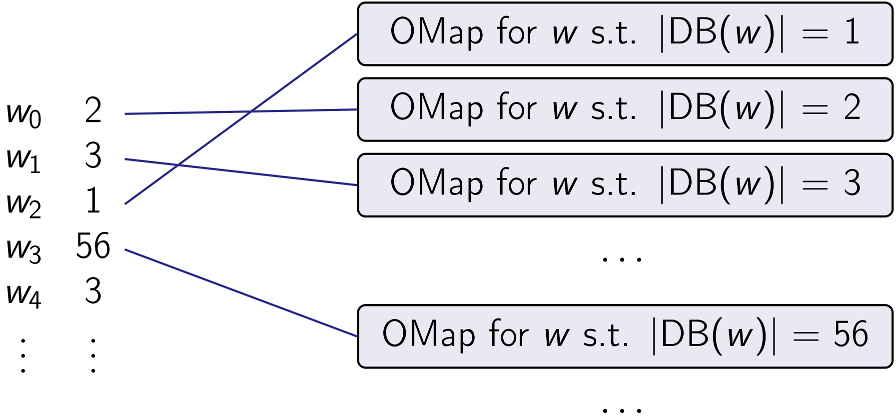
OMap for w s.t. $|\mathrm{DB}(w)| = 56$

$\dots$

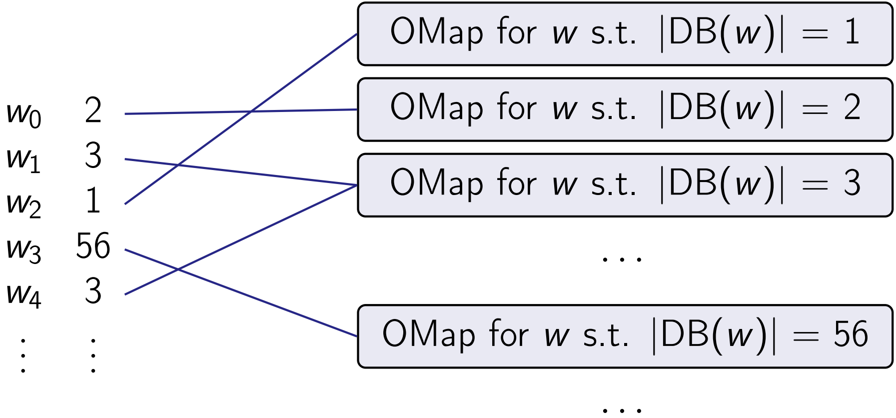
Tightness of the lower bound



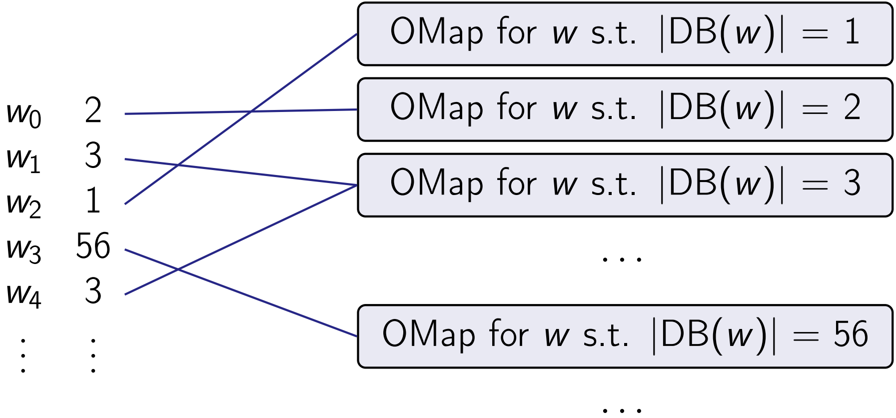
Tightness of the lower bound



Tightness of the lower bound



Tightness of the lower bound



Query complexity of an OMap of size n: $\mathcal{O}\left(\log^2 n\right)$.
The search complexity of the construction is $\mathcal{O}\left(\log^2 K\right)$.

Tightness of the lower bound

When $K \ll N$, the previous construction breaks the lower bound.

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

Tightness of the lower bound

When $K \ll N$, the previous construction breaks the lower bound.

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

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

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)

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: Search	Computation: Update	Client Storage	FP	
[CJJ+14]	$\mathcal{O}(a_w)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✗	
[SPS14]	$\mathcal{O}(a_w + \log N)$ $\mathcal{O}\left(n_w \log^3 N\right)$	$\mathcal{O}\left(\log^2 N\right)$	$\mathcal{O}(N^\alpha)$	✓	Supports deletions well
$\Sigma o\phi o\varsigma$	$\mathcal{O}(a_w)$	$\mathcal{O}(1)$	$\mathcal{O}(K)$	✓	TDP
[EKPE18]	$\mathcal{O}(a_w)$	$\mathcal{O}(1)$	$\mathcal{O}(K)$	✓	write during search
[KKL+17]	$\mathcal{O}(a_w)$	$\mathcal{O}(1)$	$\mathcal{O}(K)$	✓	(write during search)
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 Σ be a forward-private SSE scheme. Then either the update complexity of an update is

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

or the complexity of a search is

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

t is the number of updates since the last search query.

Tightness of the FP lower bound

- $\Sigma o\phi o\varsigma$, 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.
  $|\sigma| = \mathcal{O}(1)$, $\mathcal{O}\left(\log^2 K\right)$ search & update complexity.
- Open question: is there a middle point?
  e.g. $|\sigma| = \mathcal{O}\left(\sqrt{K}\right)$ & $\mathcal{O}(1)$ update complexity.

Conclusion

- Two lower bounds showing the tradeoffs between security and efficiency
- These bounds are (essentially) tight

Thank you!

Slides: https://r.bost.fyi/slides/PETS19.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.

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

References II

- 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 III

- 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 2017 (Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, eds.), ACM Press, October / November 2017, pp. 1449–1463.
- Emil Stefanov, Charalampos Papamanthou, and Elaine Shi, *Practical dynamic searchable encryption with small leakage*, NDSS 2014, The Internet Society, February 2014.

References IV

Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou, *All your queries are belong to us: The power of file-injection attacks on searchable encryption*, USENIX Security 2016 (Thorsten Holz and Stefan Savage, eds.), USENIX Association, August 2016, pp. 707–720.