Tight Tradeoffs in Searchable Symmetric Encryption

Gilad Asharov
Cornell Tech

Gil Segev
Hebrew University

Ido Shahaf
Hebrew University

The Efficiency of SSE [CT14]

Space usage: The size of the encrypted database

- Want: $O(N)$ ($N =$ size of DB)

Read efficiency: The ratio between the number of bits the server reads with each query and the size of the result

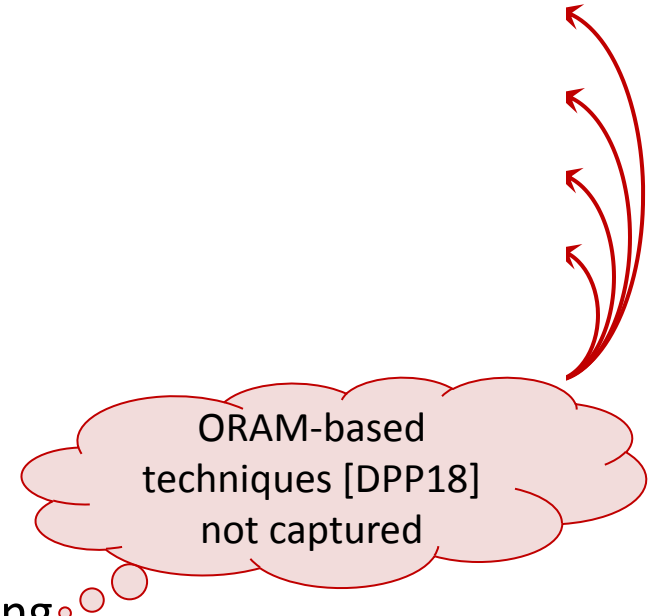
- Want: $O(1)$

Locality: The number of non-contiguous memory locations the server accesses with each query

- Want: $O(1)$

Existing Schemes and Lower Bounds

	Space	*Locality*	*Read efficiency*



We consider schemes that

- do not modify the encrypted database while searching
- have "minimal leakage" = size of database + size of each result + query repetition pattern

Existing Schemes and Lower Bounds (cont.)

Not all schemes have non-overlapping reads!

- n_w = number of documents associated with the queried keyword
- $\epsilon(n_w)$ = the unique number ϵ such that $N^{1-\epsilon} = n_w$

upper bounds and lower bounds?

(1) Under the assumption that no keyword appears in more than $N^{1-1/\log\log N}$ documents

Our Contributions

Current SSE Techniques

NEW

Pad-and-Split Framework

Main result:

- A tight lower bound
- Matches the scheme of [DP17]

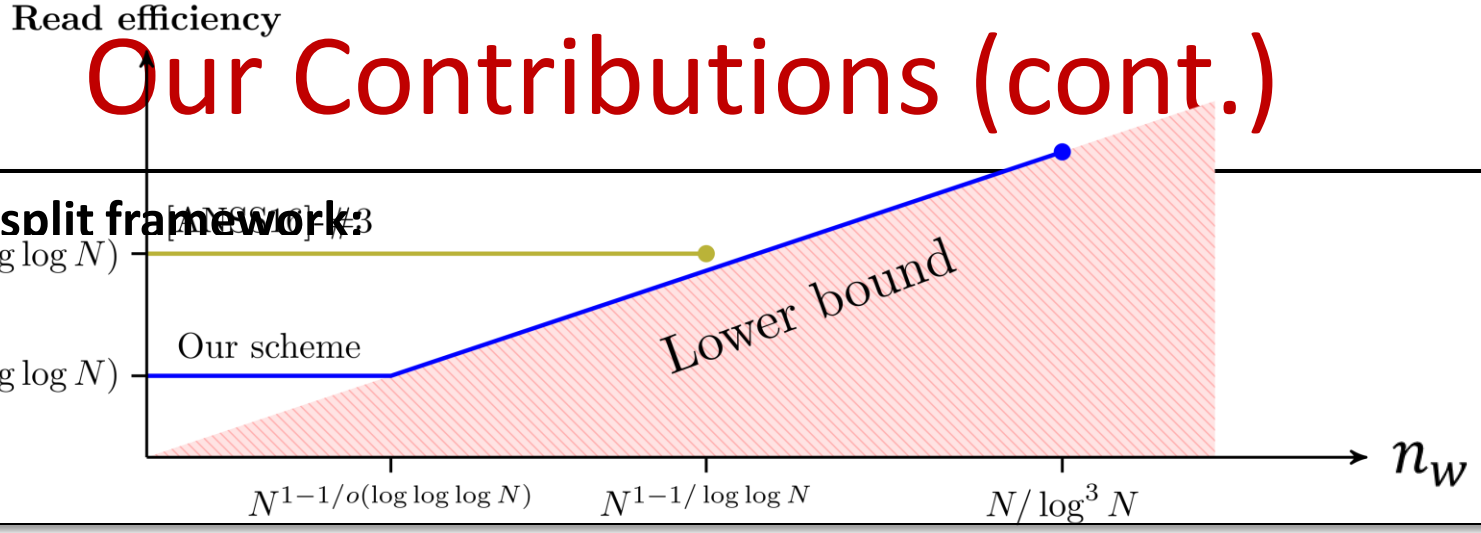
Statistical-Independence Framework

Main result:

- An improved upper bound
- Matches the [ANSS16] lower bound for a range of parameters

Our Contributions (cont.)

Pad-and-split framework:



Statistical-independence framework:

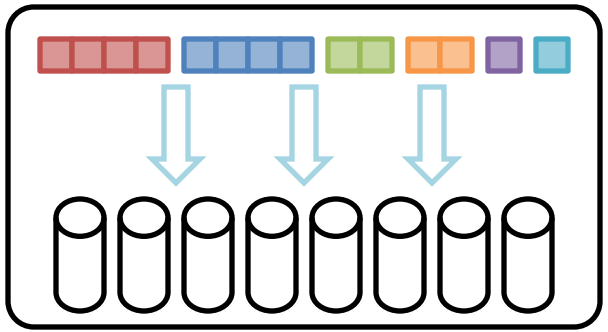
- Under the assumption that no keyword appears in more than $N/\log^3 N$ documents
- n_w = number of documents associated with the queried keyword
- $\epsilon(n_w)$ = the unique number ϵ such that $N^{1-\epsilon} = n_w$

This Talk: Our Scheme

	Space	*Locality*	*Read efficiency*
Our scheme	$O(N)$	$O(1)$	$\omega(1) \cdot \epsilon(n_w)^{-1} + O(\log\log\log N)$

See our paper for the pad-and-split framework and lower bound

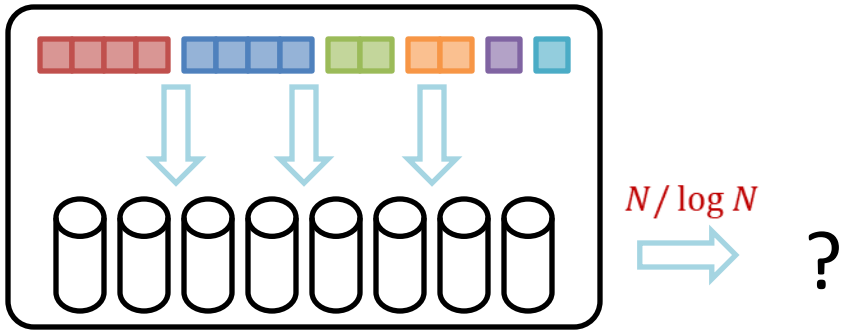
The [ANSS16] 2-Dim 2-Choice Allocation



- Allocate lists into bins with the following properties:
  - The elements of each list are placed in consecutive bins
  - The location of the first element of each list is in 1 out of 2 random bins
- To prevent overflowing bins, require that all lists are of length at most $N^{1-1/\log\log N}$ and set the bin size to $\tilde{O}(\log\log N)$

This yields: space $O(N)$, locality $O(1)$, read efficiency $\tilde{O}(\log\log N)$

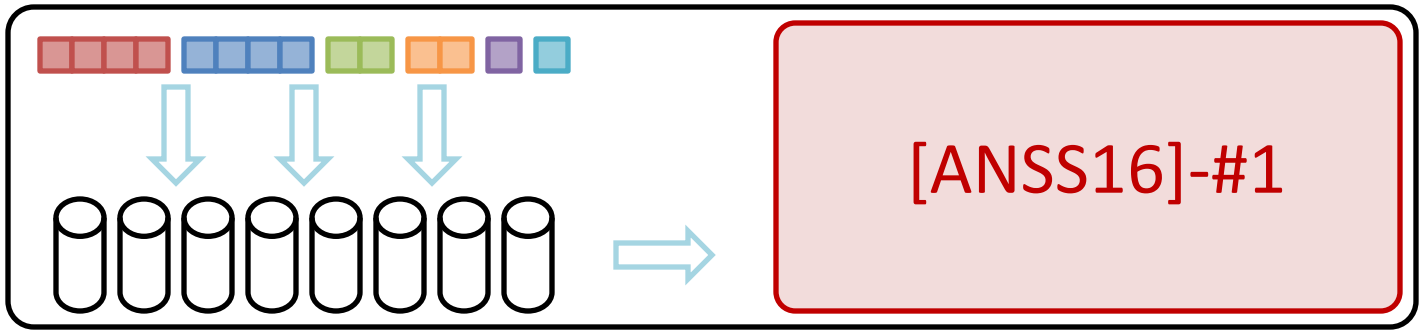
Our Approach: Allow Overflow!



- Assuming all lists are of length at most $N/\log^3 N$, if we set the bin size to $O(\log\log\log N)$ and the number of bins to $N/\log\log\log N$, then there are at most $N/\log N$ overflowing elements
  (with an overwhelming probability)

What should we do with the overflowing lists?

Naïve Attempt

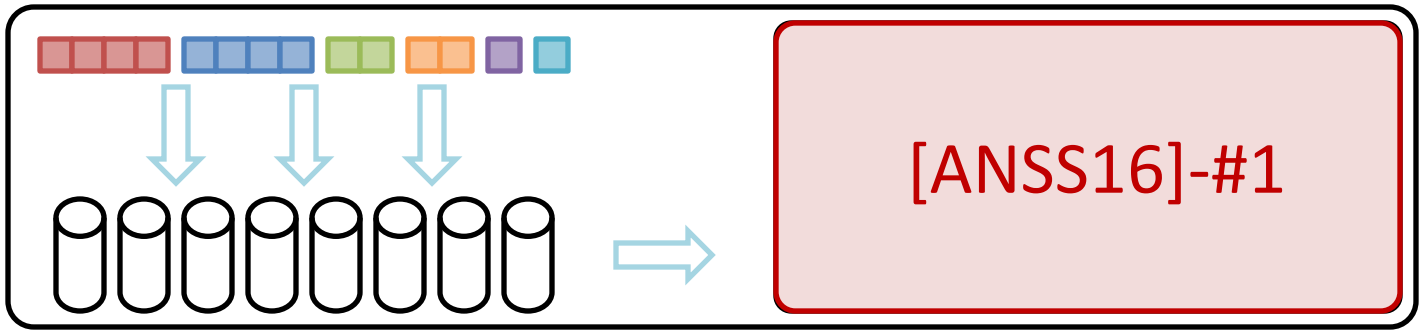


Use the [ANSS16]-#1 scheme for the overflowing lists

- Optimal locality and read efficiency
- Space $O(N' \cdot \log N') = O(N)$, where $N' = N / \log N$

The problem: Revealing which lists are overflowing leaks information!

Our Solution

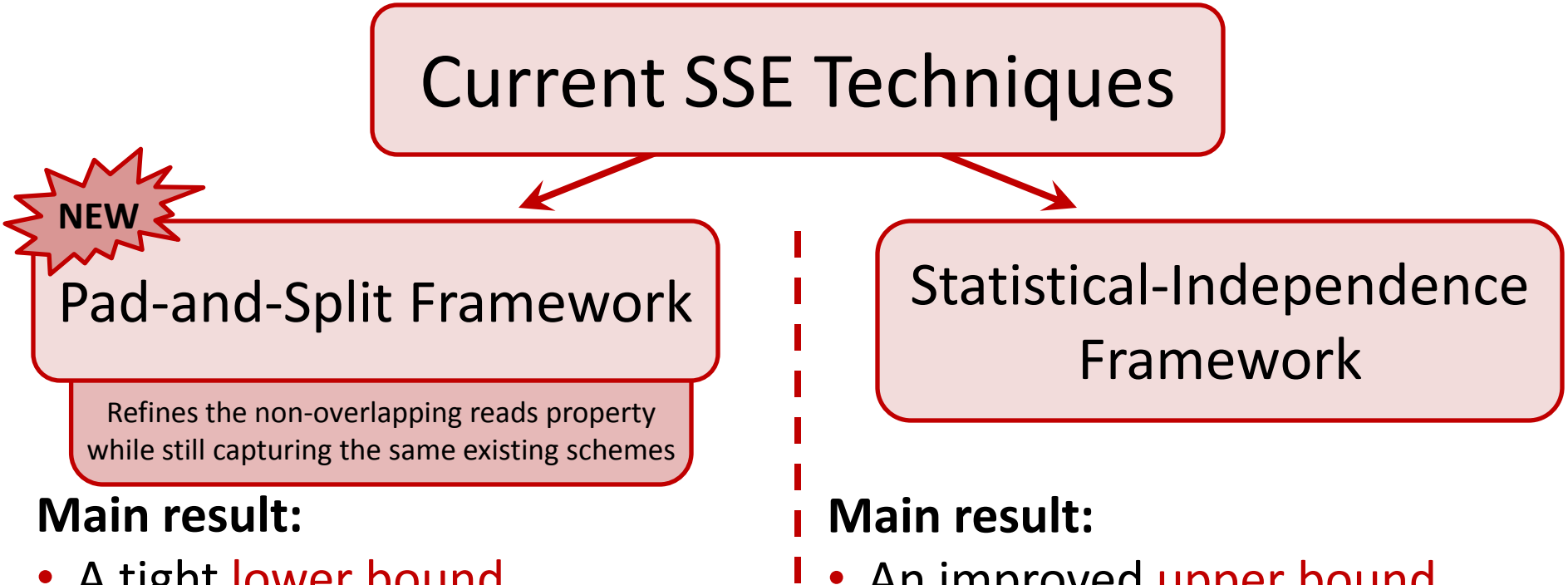


- Modify the [ANSS16]-#1 scheme such that it will not reveal whether a list is stored in it
- **Key idea:** Avoid rehashing using cuckoo hashing with a stash
  - Stash size depends on the length of the list

This yields: space $O(N)$, locality $O(1)$, read efficiency

Bin size — $O(\log\log\log N) + \omega(1) \cdot \epsilon(n_w)^{-1}$ — Stash size

Summary



Main result:

- A tight lower bound

Main result:

- An improved upper bound

Thank you!