Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions

Reza Curtmola, Juan Garay, Seny Kamara, Rafail Ostrovsky

Work appeared in CCS'06

Presented by: Rachit Garg and Aravind Birudu

Motivation

- Searching is the primarily way we access data. We google stuff for most things today.
- We are outsourcing more and more of our data to third parties and we trust them less and less.
- Examples:
  Industry: https://numer.ai/, for open source(data) machine learning
  Governments: Aadhaar Database
  Research Scientists: Genomic Data

The components

- Ways to encrypt data
- Model of the paper
- Prior Work
- Revisiting previous security definitions for SSE
- Two new notions of security for SSE
    - “Non-adaptive” security
    - “Adaptive” security
- New constructions
- Further work

Ways to compute on encrypted data

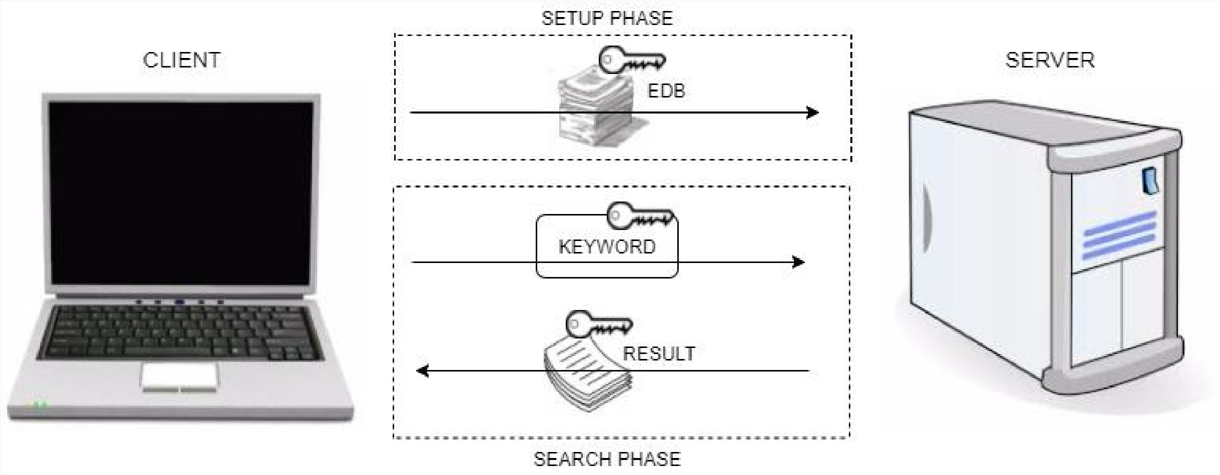
We know of six different ways to search on encrypted data, each based on one of the following cryptographic primitives:

- property-preserving encryption
- functional encryption
- fully-homomorphic encryption
- searchable symmetric encryption
- oblivious RAMs
- secure two-party computation

The Model

Client uploads documents and an additional data structure to the server. This additional structure helps in searching in the database.

Goal: The leakage to the server should be minimum while the scheme being efficient

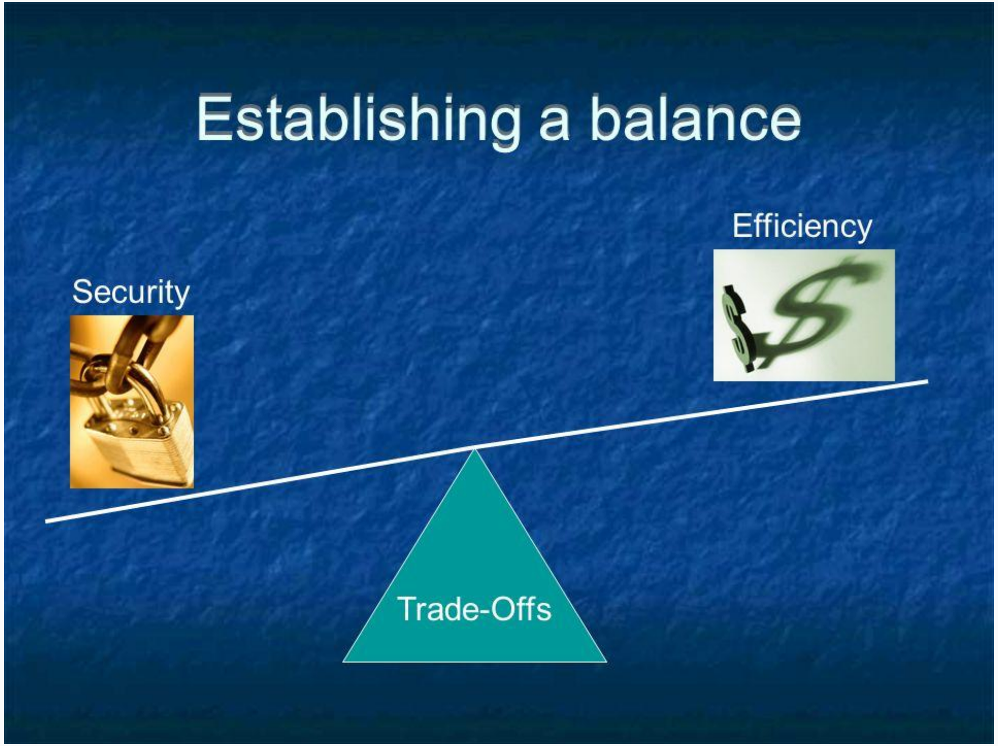


Security Definitions

- **History**: History of the queries, includes the document collection and the word queried.
- **Access Pattern**: Document indices returned by the queries.
- **Search Pattern**: Indicates what are the keywords that are searched for.
- **Trace**: Contains the length of the database, the access pattern and the search pattern.

ORAMs

- SSE through oblivious RAMs.
  - ORAM’s can simulate data structures in a secure way, can support conjunctive queries.
  - Hides everything, even the access patterns.
  - Efficiency is logarithmic number of rounds for each read/write Lower bound shown by Goldreich and Ostrovsku.
  - Boneh-Kushilevitz-Ostrovsky-Skeith showed sqrt{DB} communication with constant rounds.



Get more efficiency! By leaking the access pattern but nothing else.

Prior Work

- Previous Attempts
  - “Practical techniques for searches on encrypted data” [SWP00]
  - “Secure Indexes” [Goh03]
  - “Privacy-preserving keyword searches on remote encrypted data” [CM05]

- [SWP00,Goh03,CM05]: “A secure SSE scheme should not leak anything beyond the outcome of a search”

[SWP00] : “any function of the plaintext that can be computed from the ciphertext can be computed from the length of the plaintext”
Issue: adversary gets to see search outcomes and search pattern

New Definition: “any function about the documents and the keywords that can be computed from the encrypted documents, the index and the trapdoors can be computed from the length of the documents, the search outcomes and the search pattern” (**adaptive and non adaptive!**)

IND2-CKA: indistinguishability against chosen-keyword attacks(used by [Goh03]) : “any function of the documents that can be computed from the encrypted documents and the index can be computed from the length of the documents and the search outcomes.

[CM05]: Any function that can be computed about the documents and keywords that can be computed from the encrypted documents, the index and the trapdoors can be computed from the length of the documents and the search outcomes. **(non adaptive)**

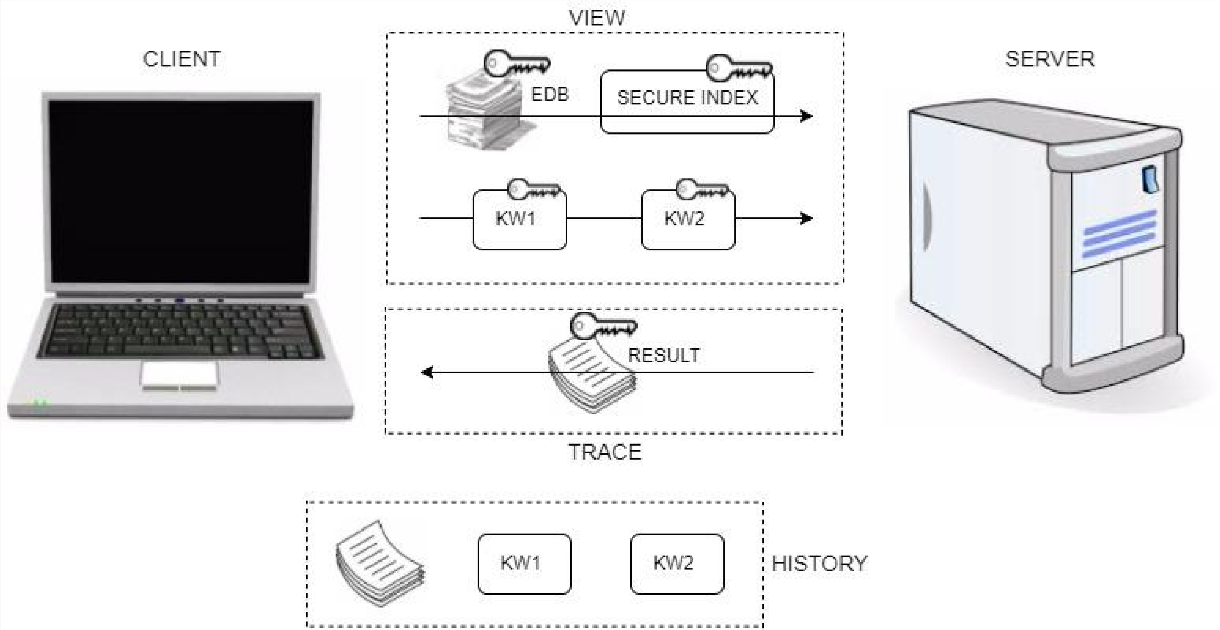
Question: Why not prove index secure in the sense of IND2-CKA and trapdoors “secure” using another definition?

They showed that there exists an SSE scheme that has IND2-CKA indexes and trapdoors that are “secure” but when taken together, adversary can recover keyword.

Algorithms

- **Keygen(1^k)**: outputs symmetric key K
- **BuildIndex(K, {D_1, ..., D_n})**: outputs secure index I
- **Trapdoor(K, w)**: outputs a trapdoor T_w
- **Search(I, T_w)**: outputs identifiers of documents containing w (id_1, ..., id_m)

SSE Model



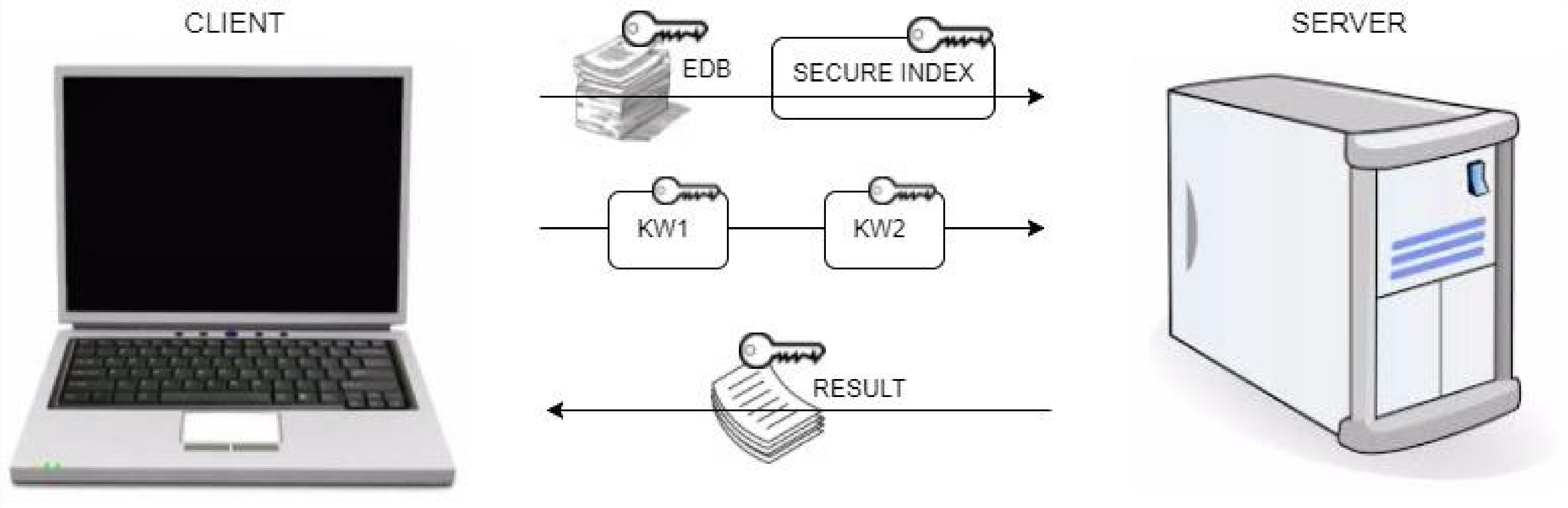
VIEW
CLIENT
SERVER
EDB
SECURE INDEX
KW1
KW2
RESULT
TRACE
KW1
KW2
HISTORY

Adaptiveness

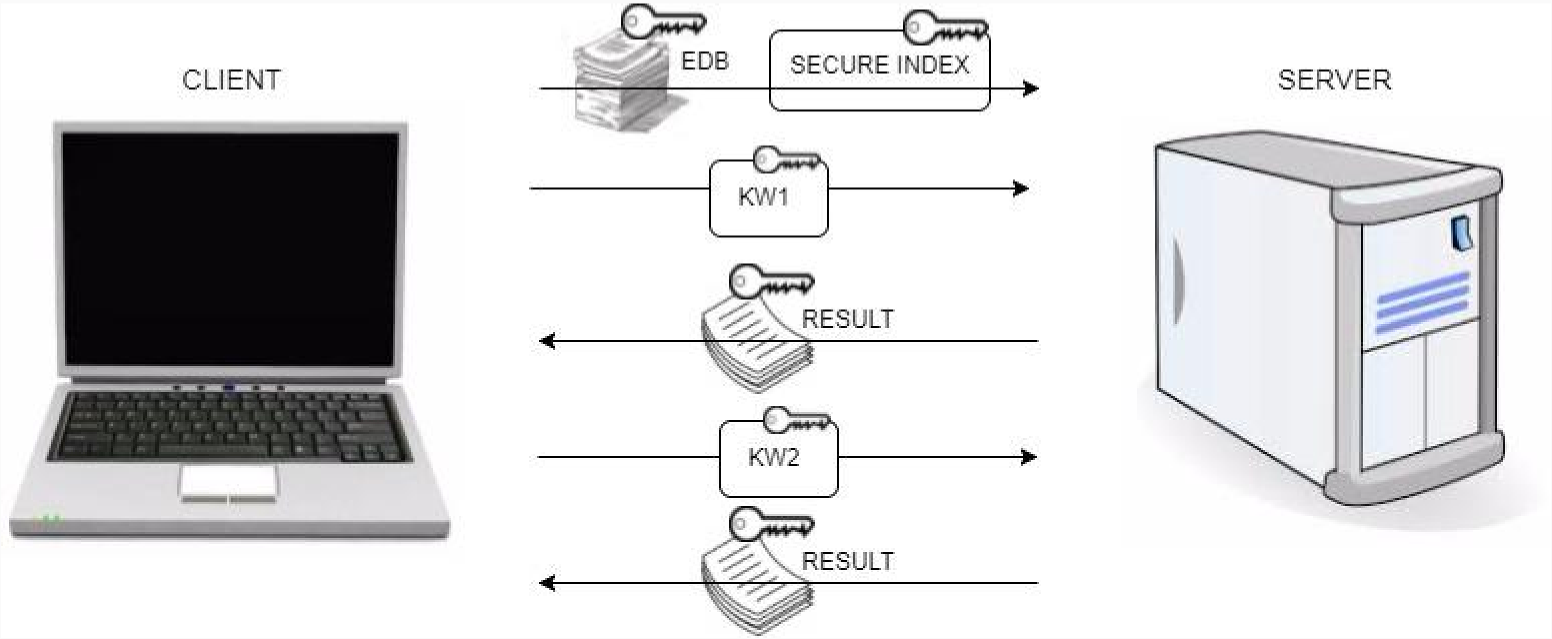
Non Adaptive: Non-adaptive adversaries make search queries without seeing the outcome of previous searches.

Adaptive: Adaptive adversaries can make search queries as a function of the outcome of previous searches.

NON ADAPTIVE



ADAPTIVE



Inverted Index Solution

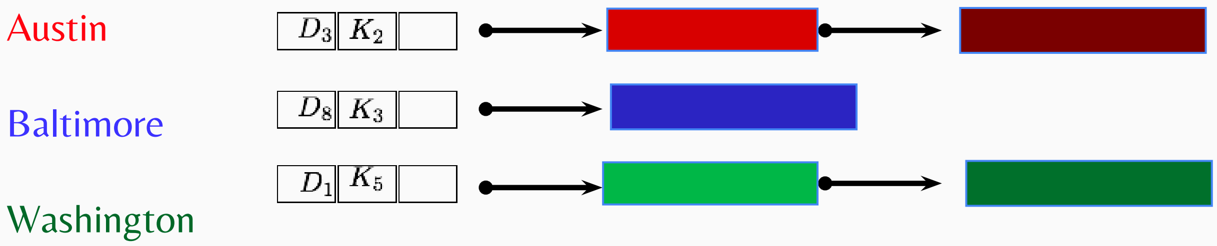
- For each distinct keyword $w_i \in \delta(D)$, a linked list L_i is created that points to all documents containing that keyword.
- We then store all the nodes of all the lists in the array A permuted in a random order and encrypted with randomly generated keys.
- Before encrypting the j th node of list L_i , it is augmented with a pointer (with respect to A) to the $(j + 1)$-th node of L_i , together with the key used to encrypt it.
- Note that by storing the nodes of all lists L_i in a random order, the length of each individual L_i is hidden.
- We then build a look-up table T that allows one to locate and decrypt the first node of each list L_i .

Inverted Index Solution

- The client generates both A and T based on the plaintext document collection D, and stores them on the server together with the encrypted documents.
- When the user wants to retrieve the documents that contain keyword w_i, it computes the decryption key and the address for the corresponding entry in T and sends them to the server.
- The server locates and decrypts the given entry of T, and gets a pointer to and the decryption key for the first node of L_i. Since each node of L_i contains a pointer to the next node, the server can locate and decrypt all the nodes of L_i, revealing the identifiers in $D(w_i)$.

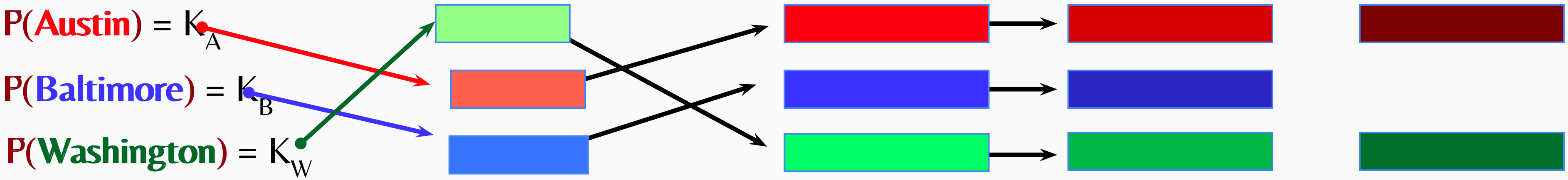
Construction:- The Inverted Index Solution

- Building a Secure Index



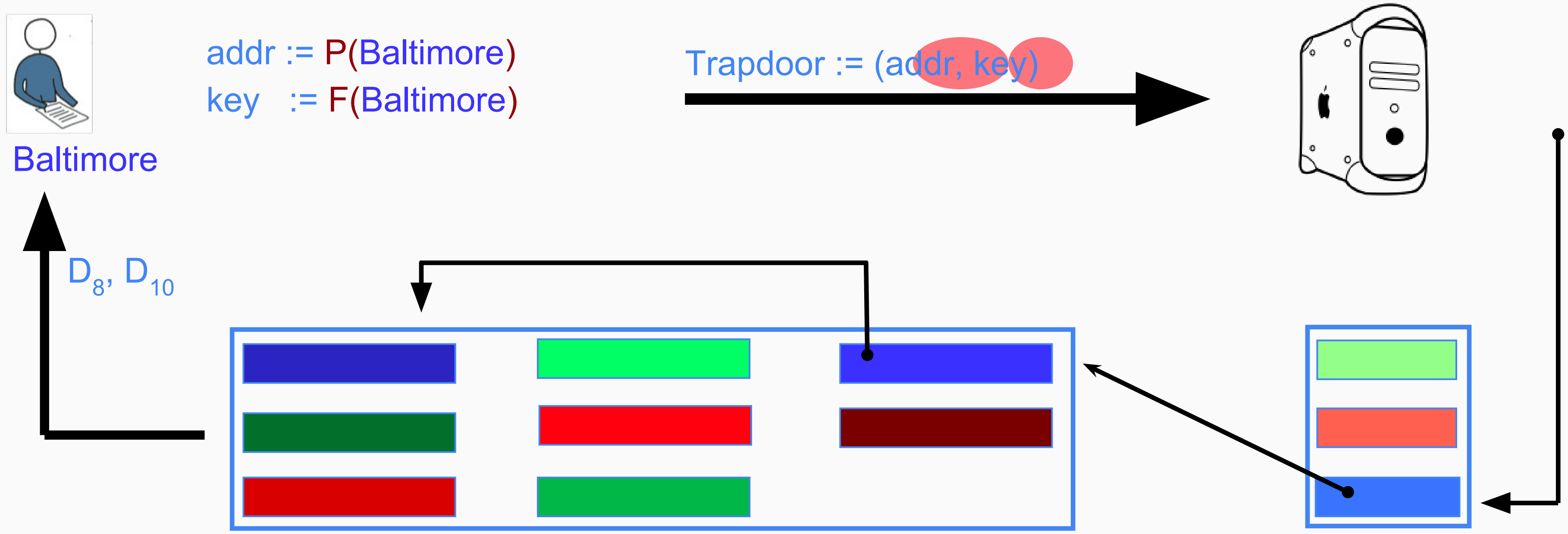
Slide taken from
web.cs.ucla.edu/~rafail/PUBLIC/SSE.ppt

- Building a Secure Index
- P: PRP
- F: PRF



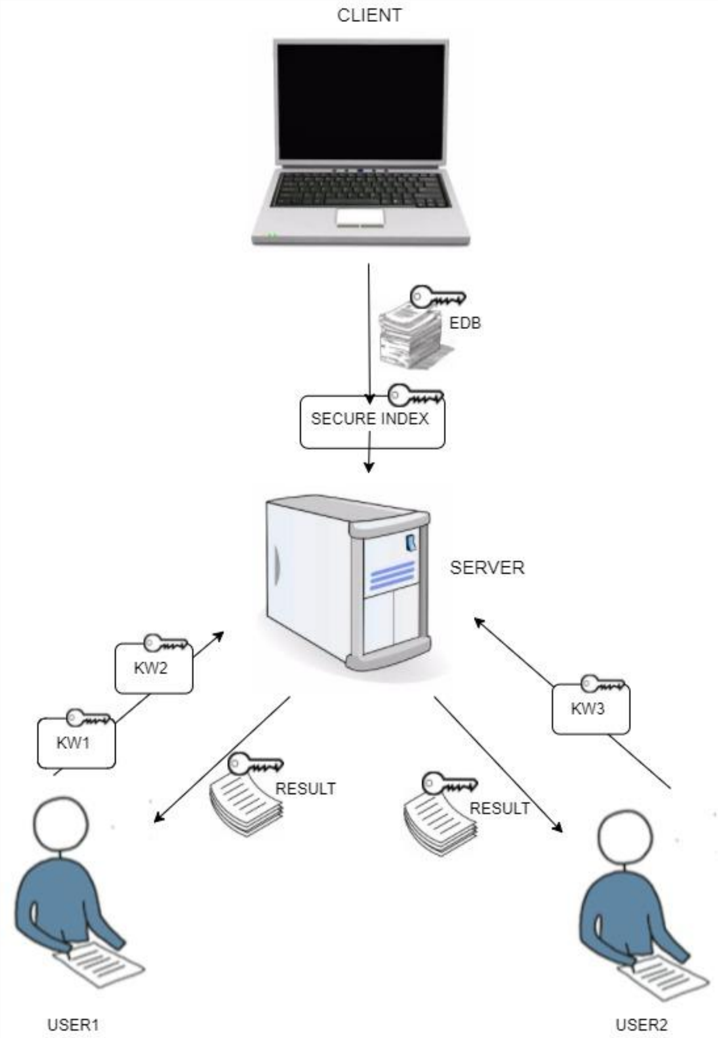
Slide taken from
web.cs.ucla.edu/~rafail/PUBLIC/SSE.ppt

Searching



Slide taken from web.cs.ucla.edu/~rafail/PUBLIC/SSE.ppt

Multi - User SSE



Multi User SSE

- Indexes and trapdoors require same security notions as single-user SSE
- Revocation: owner can revoke searching privileges robust against user collusions
- Anonymity: server should not know who initiated search
- Simple construction that transforms single-user SSE schemes to multi-user SSE schemes.

Conclusion Table

Properties	ORAM	Other Solutions	This paper
Hides access patterns	yes	no	no
Server computation	$O(\log^2 n)$	O(n)	O(1)
Server storage	O(n logn)	O(n)	O(n)
Number of rounds	log n	1	1
communication	$O(\log^2 n)$	O(1)	O(1)
Adaptive adversaties	yes	no	no

Further Work

- This was a foundational work in SSE, it defined the necessary security definitions. There have been several improvements to the solution described in this paper.
- The use of FKS dictionaries have been removed.
- The inverted index solution is a static scheme, dynamic schemes have been proposed.
- Cash, Jarecki, Jutla, Krawczyk, Rosu and Steiner have extended inverted index solution to handle boolean queries.

Thanks!