Outsourced Symmetric Private Information Retrieval

=

Searchable Encryption in Multi-Client Setting

David Cash, Stanislaw Jarecki, Charanjit Jutla,

Hugo Krawczyk, Marcel Rosu, Michael Steiner

Supported by US IARPA SPAR Program

CRYPTO'13, CCS'13, on-going (submission/preparation)

Talk Plan

- Encrypted Cloud Storage and Searchable Encryption
- The IARPA SPAR Searchable Encryption Project
- Technical Overview
  (conjunctive search on encrypted data)
- Research Challenges

The Data-in-the-Cloud Conundrum

- Our data in the cloud: email, file backups, financial info, etc.
- Data is visible to the cloud server (hopefully encrypted but with *their* keys), and to anyone with access to that server

- Q: Why not encrypt it with your (data owner) own keys?
- *A*: Because we want the cloud to search the data (e.g. gmail)
- **Can we keep the data encrypted and search it too?**

Encrypted Search I (SSE)

- DB owner *outsources* its data to a cloud server such that:
- Data Owner:
  - pre-processes data, outsources to cloud server, keeps only a cryptographic key, later runs queries to retrieve/decrypt matching documents
- Cloud Server:
  - gets all DB documents in encrypted form
  - gets index information (metadata) in encrypted form
  - responds to read queries by returning matching encrypted records
  - does not learn the searched terms or DB plaintext information (but assume that some leakage on data-access and query patterns allowed)

Encrypted Search II (Multi-Client SSE)

- Data Owner outsources DB to cloud server which (as before):
  - keeps all records and index information in encrypted form
  - responds to read queries by returning matching encrypted records
  - does not learn the searched terms or any plaintext information on the DB (although some access-pattern leakage allowed)
- While Data Owner:
  - *can delegate search to third-party clients* (via search tokens)
  - such that clients can search through authorized queries but learn nothing about data not matching the authorized queries
  - multiple and adversarial clients (fully malicious in our solutions)

Encrypted Search III (PIR-SSE)

- As scenario II

  PLUS

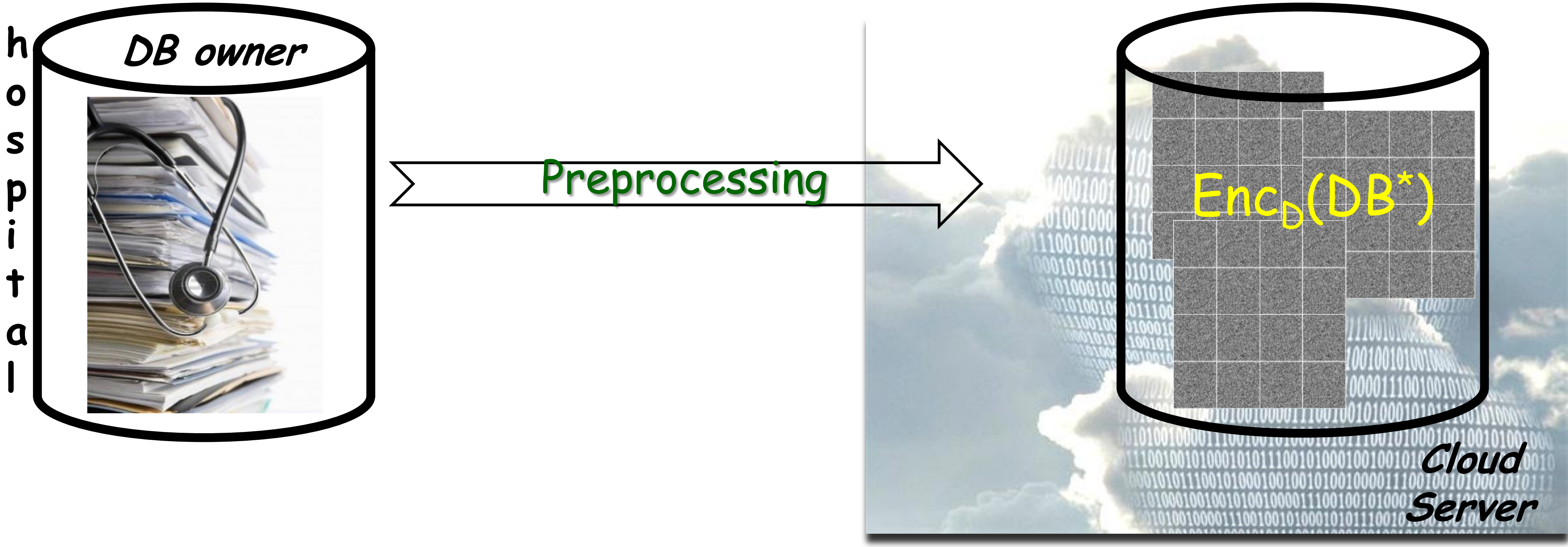
- Data Owner can authorize clients to perform queries according to some prescribed policy

  (i.e., determine the query compliance and provide the corresponding tokens)

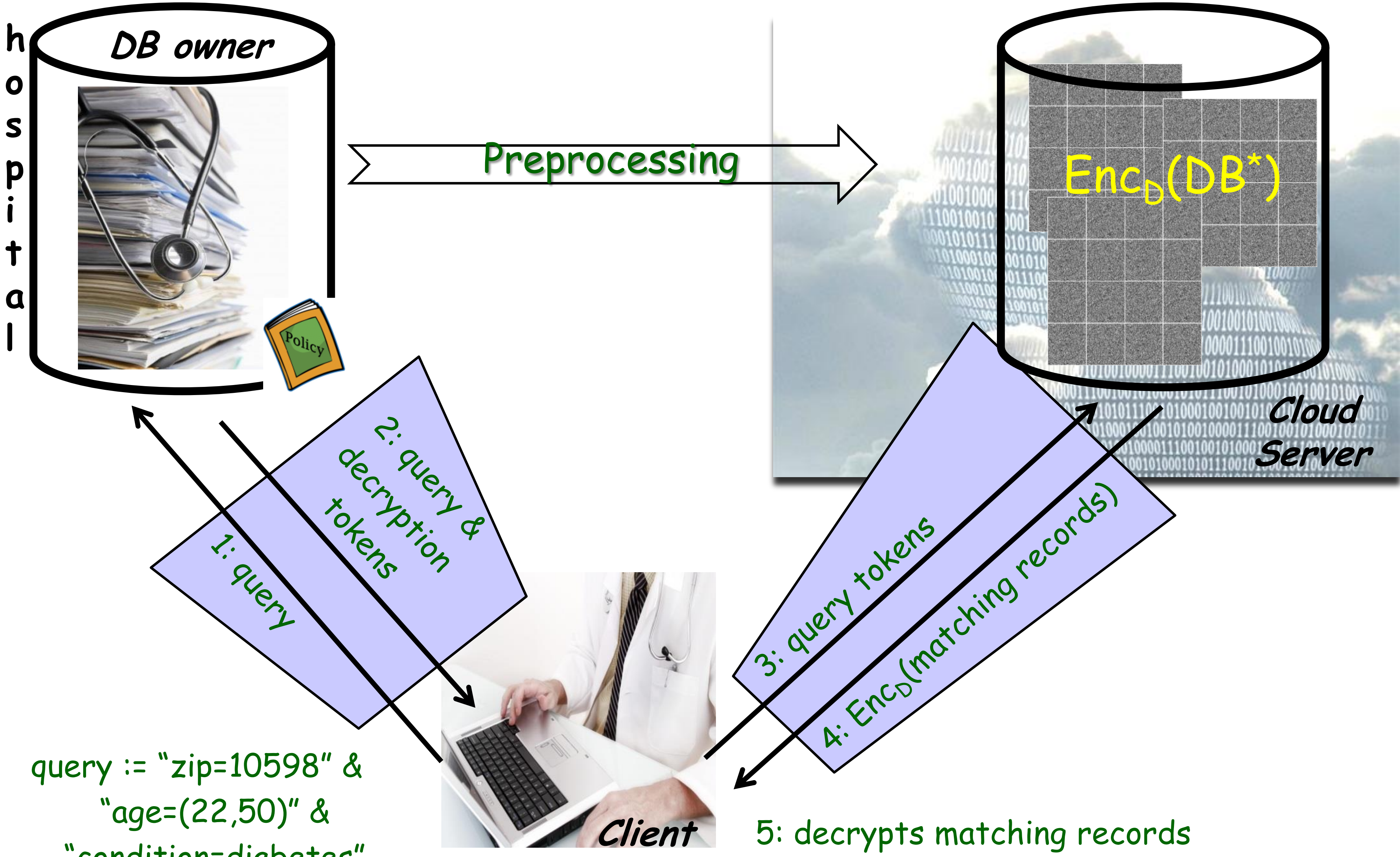
- ... but she has to do so without learning the searched terms

  - Data Owner and Cloud Server do not collude
    (otherwise strong performance limitations of PIR)

PIR-SSE by Example: Medical DB



PIR-SSE by Example: Medical DB



SSE Application Examples

- Commercial examples
  - Data repositories (file system backup, email, databases)
  - Outsourced data service (e.g., processed census data, patents, research)
  - Regulatory/liability (e.g. medical records, commercial records)
- Judicial and intelligence examples (next…)

IARPA SPAR Program

- SPAR: Security and Privacy Assurance Research
- Very ambitious program:
  - PIR-SSE privacy requirements
  - Complex authorization scenarios (e.g. authorizing queries w/o learning them)
  - Wide range of query types: conjunctions, Boolean, range, substrings,…
  - Dynamic databases (support additions, deletions, modifications, caching)
  - **Huge databases**
    - Any Boolean query on 100,000,000 records, each w/ 300 searchable keywords
    - That's any Boolean query on $3*10^{10}$ = 30,000,000,000 record-keyword pairs…
    - Orders of magnitude above full Wikipedia encrypted search (which we do too)
  - **Formal analysis and proofs a MUST**

IARPA SPAR Motivating Applications (?)

- Searching for suspect in airline/hotel/IRS records
  - data owner should limit access but without learning who is being searched
- CIA accessing FBI records for targeted information
  - political/regulatory limits on what FBI/CIA can learn about each other
  - reduce agencies' reluctance to share information (9/11, Boston bombing)
- Recent news of US security agencies accessing phone/email DBs…
  - incentive for security agencies to enabling (preserving?) access while providing demonstrable privacy & accountability assurances

SSE State of the Art (Generic Solutions)

- Impractical
  - Send all data back to owner to decrypt and search
  - Use fully homomorphic encryption and send back only the encrypted result set
- Semi-practical
  - Run a search algorithm under an Oblivious RAM (ORAM) compiler
    - recent ORAM advances makes this less impractical than in the past, yet confined to relatively small DB's

SSE State of the Art (Single-Keyword SSE)

- Efficient SSE mechanisms known only for single-keyword search
  - Keyword search: Given one keyword return all documents that contain that keyword (e.g. find email containing "crypto", records with name "Bob", etc.)
  - Server allowed to learn the set of encrypted matching documents but not the keyword or plaintext data
  - Several works [SWP'00, Goh'03, CGKO'06, ChaKam'10, …] achieve:
    - "privacy optimal" (server learns DB size and encrypted result sets),
    - lots of room for implementation/performance improvement (small DBs restricted to RAM size, static data, inefficient adaptive solutions)
    - Some recent improvements on adaptive solutions and dynamic data for single-keyword search [KPR'12, KP'13, our work (in submission), …]

SSE State of the Art (Conjunctive SSE)

- Beyond Single-Keyword Search: Very little known
  - Conjunctions: Find all documents containing n keywords: $w_1, \ldots, w_n$
  - Existing solutions to conjunctive queries are either
    - "brute force": Do n single-keyword searches, compute the intersection (inefficient and very leaky…)
    - *linear in the number of documents* [GSW'04, BKM'05, BLL'06, PRVBM'11]

Crypto'13: SSE for Boolean Queries

- Practical Searchable Symmetric Encryption (SSE) with:
  - Support for any Boolean expression on keywords
    - Example: Search for messages with Alice as Recipient, not sent by Bob, and containing at least two of the words {searchable, symmetric, encryption}
    - Applies to both relational DBs (attribute-value) and free text (e.g. English)
  - Efficient for a large class of expressions
    - w_1 AND B(w_2,…,w_n) for <u>any</u> Boolean expression B (including negations)
    - in particular, *conjunctions on any number of terms*
    - … and complex examples as above (w_1 = "Alice as Recipient")
    - Any disjunction of above expressions

Highly Scalable System

- Search proportional to # documents matching the least frequent term
- Preprocessing scales linearly with DB size
- Validated on synthetic census data: 10Terabytes, 100 million records, > 100,000,000,000 indexed record-keyword pairs !
    - Equivalent to a DB with one record for each American and 400 keywords in each record (including textual fields)
- Other DB's: Enron email repository, ClueWeb (>> English Wikipedia)
- Query response time: Competitive w/ plaintext queries on indexed DB

Security

- Security-Performance trade-offs:
  - Leakage on (DB,query) information to the Cloud Server in the form of:
    - data access patterns (e.g. repeated retrieval)
    - query patters (repeated queries)
    - + additional leakage (more complex functions of DB and query history)
  - Can lead to statistical inference based on side information on data (application dependent), can be alleviated by masking techniques
  - No plaintext DB data or query ever revealed (other than via statistical inference)
- Security proofs: formal model and precise provable leakage profile
  - Leakage profile: provides upper bounds on what is learned by the server

Security Formalism

- Based on the simulation-based definitions given for SKS [CGKO,CK].
- There is an attacker S (cloud server), a simulator SIM, and a *leakage function* L(DB, queries):
  - Real: Attacker S chooses DB and queries (adaptively), gets encrypted DB and interacts with client running queries chosen by S
  - Ideal: Attacker S chooses DB and queries (adaptively), gets the output of SIM(L(DB,queries))

A SSE scheme is *semantically secure with leakage L* if for all attackers S, there is a simulator SIM such that S's view in both experiments are indistinguishable

➔ Server learns nothing beyond the specified leakage L even if it knows (and even if it chooses *adaptively*) the plaintext DB and queries

Crypto'13: Boolean Query SSE (basic ideas)

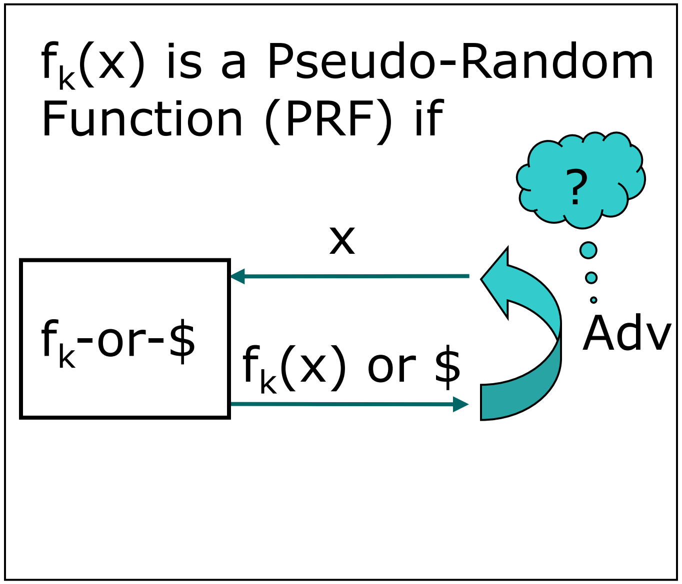
- Assume a conjunctive query $w_1, \ldots, w_n$ (extends to Boolean queries)

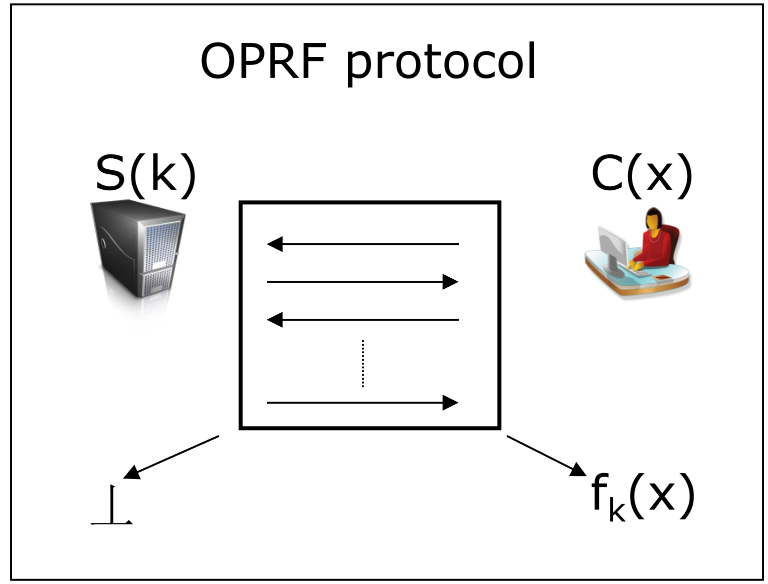
1. choose the *least frequent* conjunctive term ("s-term"), say w_1
2. find encrypted indexes of all records containing w_1 (w/o revealing w_1)
   - Based on a pre-computed encrypted index stored at server
   - $PRF_k(w) \rightarrow Enc(ind_1), Enc(ind_2), \ldots, Enc(ind_k)$
   - Non-trivial: Space-efficient storage of encrypted files whose length should be hidden from the server
     - Even less trivial: what if files range from 100B to 100MB, what if you need to update them and the daily update rate is a significant fraction of the DB?

Q1: How to compute PRF values obliviously?

Q2: How to determine indexes satisfying w_1 & … & w_n , and not just w_1 ?

Oblivious PRF Computation (OPRF) [NR'04, FIPR'05]



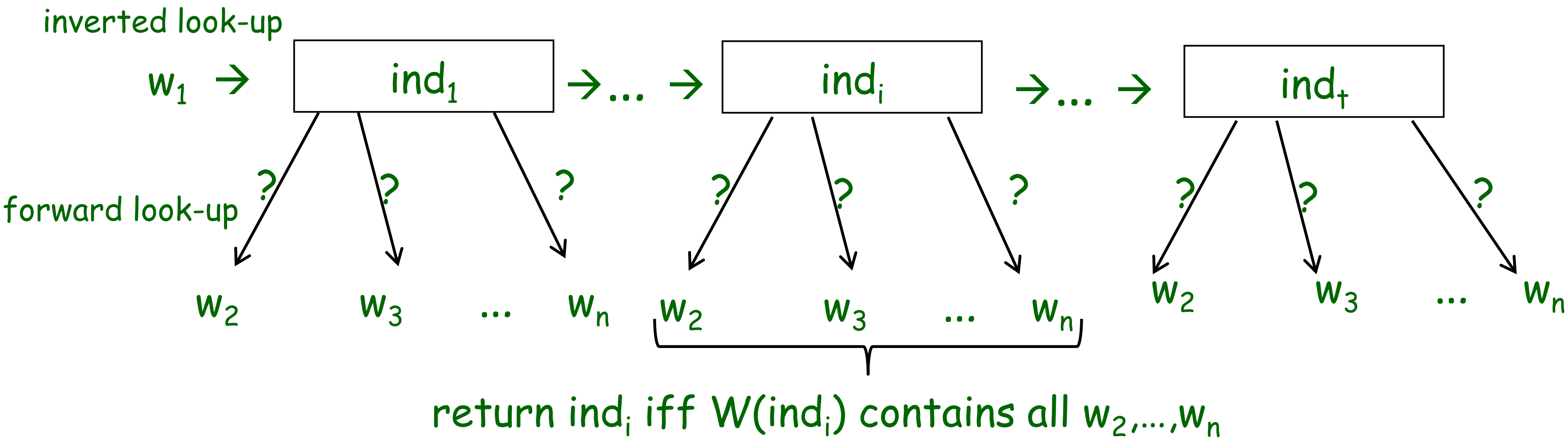


- Multiple instantiations ([Yao'82], [FIPR'01], [JL'09], [JL'10], …)
- Fastest (2 exp's/party) is Hashed-DH PRF: $F_k(x)=[H(x)]^k$
- Oblivious computation via "Blind DH Computation":

(C sends $a = [H(x)]^r$ to S, S replies with $b = a^k$, C computes $F_k(x)$ as $b^{1/r}$)

- OPRF with enforcing access policy on query x: extensions…

Standard Conjunctive Search on query = w_1 & w_2 & … & w_n



- Pre-computation: Build set xSet of hash values:

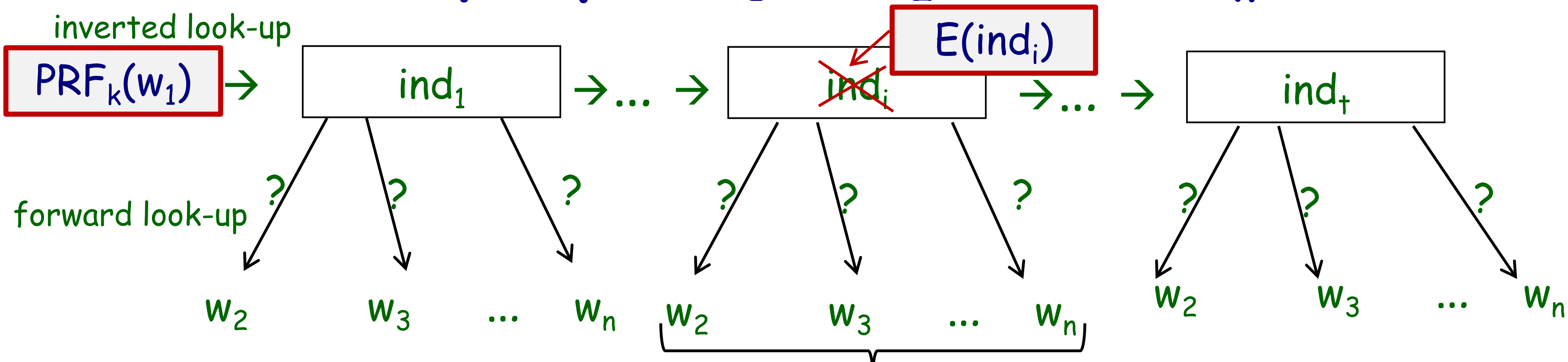
If record indexed at ind contains keyword w then add H(w,ind) to xSet

⇨ record(ind) contains keyword w iff H(w,ind) ∈ xSet

- Retrieval:

Return a tuple corresponding to ind iff H(w,ind) ∈ xSet, for j=2,…,n

SSE Conjunction Handling on query = w_1 & w_2 & … & w_n



return ind_i iff $H(w_j, ind_i) \in$ xSet for all w_j

- Implementation: Build set xSet of hash values:

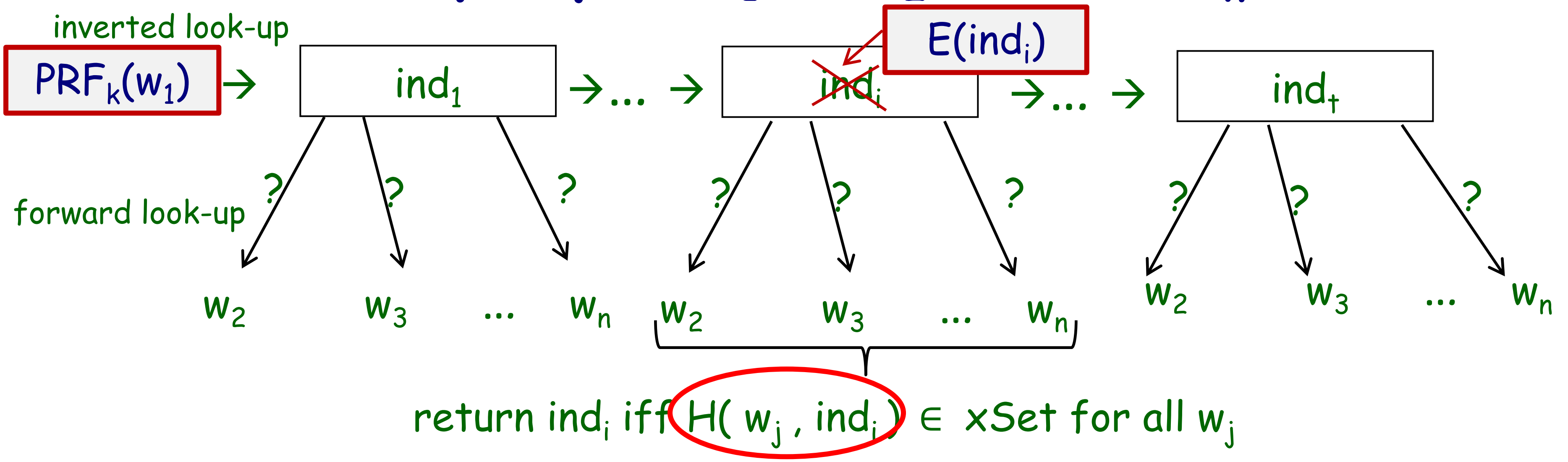
For each record index ind and each w in W(ind): add H(w,ind) to xSet

⇨ keyword w ∈ W(ind) iff H(w,ind) ∈ xSet

- EDB, during retrieval:

Return a tuple corresponding to ind iff H(w,ind) ∈ xSet, for j=2,…,n

ESPADA Conjunction Handling on query = w_1 & w_2 & … & w_n



Heart of the Crypto'13 conjunctive SSE:

Secure 2-Party Computation of value: H(w,ind)

Server's input: E(ind)

Client's input: $PRF_k(w)$
[+ decryption key for E]

Crypto'13 Conjunctive SSE Leakage

- Index size = upper bound on Σ_i $|DB(w_i)|$
- Number of terms in each conjunction
- Size of s-term set $|Rec(w_1)|$ (unavoidable?)
- Repeated usage of the s-term
- Size of $Rec(w_1 \wedge w_j)$ for j=2,…, n
- More, because function H(w,ind) is deterministic:
  - Leaks repeated usage of x-terms in two conjunctive queries if their s-terms have a non-empty intersection [⇨ repeat in the (w,ind) argument to the (deterministic) H function !]

Subsequent/Ongoing Work

- Upcoming in CCS'2013: Oblivious delegation to third-party clients
  - OPRF's with blinding factors which prevent mix-and-match of search terms across multiple queries
- Dynamic DBs': Support for data additions/deletions/modifications
- Richer queries: Range, substring, wildcards, …

SSE Challenges

- Leakage:
  - how to characterize it?
  - how to evaluate it?
- Tradeoffs: interplay security-performance (asymptotic & concrete)
  - functionality / privacy / (pre-)computation / space
- Close engineering-theory interaction
  - can't just throw a heavy weapon on the problem
- Provable security
  (especially if you are going to build/use the system)