Highly-Functional Highly-Scalable Search on Encrypted Data

Hugo Krawczyk, IBM

Joint work with IBM-UCI teams:

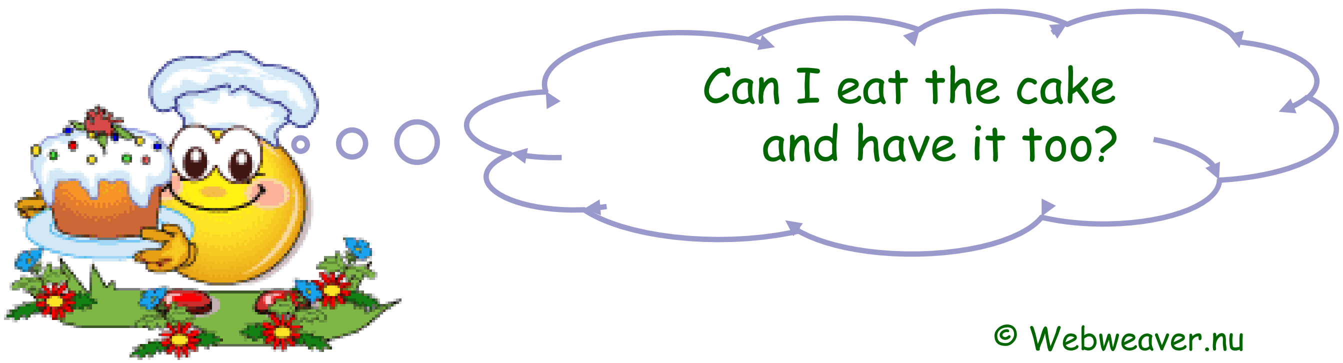
David Cash, Sky Faber, Joseph Jaeger, Stas Jarecki, Charanjit Jutla, Quan Nguyen, Marcel Rosu, Michael Steiner

DIMACS Big Data Workshop – 12/15/2015


Your data is in the cloud.
Do You?

The Data-in-the-Cloud Conundrum

- Your data in the cloud: email, backups, financial/medical info, etc.
- Data is visible to the cloud and to anyone with access (legitimate or not)
  - At best, data is encrypted "at rest" with the server's keys and decrypted upon use
- Q: Why not encrypt it with your (data owner) own keys?
- A: Utility, e.g. allow the cloud to search the data (e.g. gmail)
- **Can we keep the data encrypted and search it too?**

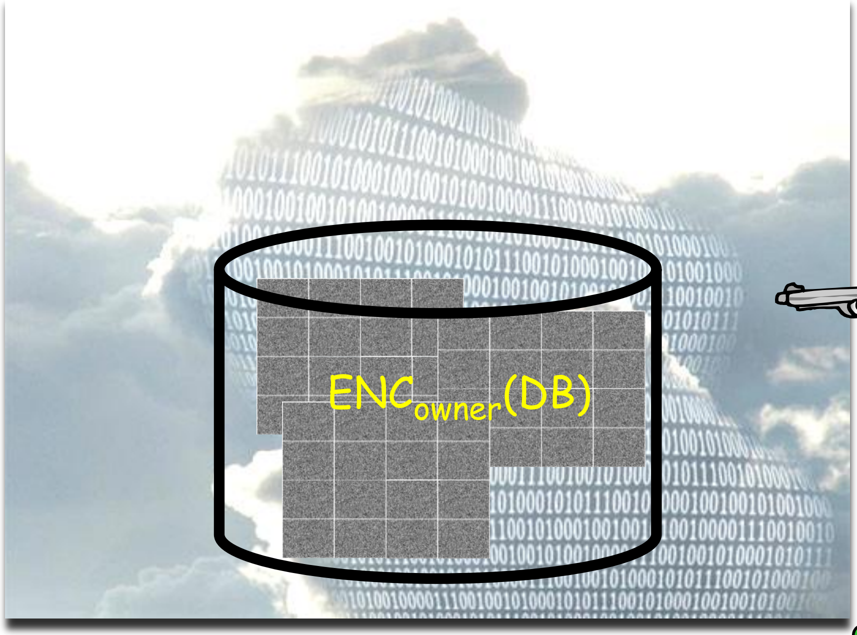


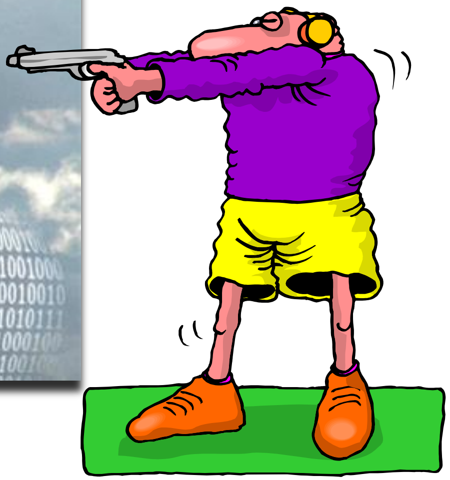
SSE: Searchable Symmetric Encryption

- Owner outsources data to the cloud: Pre-processes data, stores the processed and encrypted data at the cloud server
  - Keeps a small state (e.g. a cryptographic key)
  - Later, sends encrypted queries to be searched by the server
    - e.g. return all emails with *Alice as Recipient, not sent by Bob, and containing at least two of the words {searchable, symmetric, encryption}*
- Goal: Server returns the encrypted matching documents w/o learning the plaintext query or plaintext data
  - Some forms of statistical leakage allowed: data access patterns (e.g. repeated retrieval, size info), query patterns (e.g., repeated queries), etc.
    - Plaintext data/queries *never directly exposed*, but *statistical inference* possible
- Protects against break-ins, cloud insiders, even "surveillance attacks"

With SSE…

The cloud cannot disclose your data… ***not even at gun point!***





SSE before 2013

- Generic tools: FHE, ORAM, PIR
  - very expensive,
  - great* security
    - *assumes *all* raw data is ORAM-encrypted, o/w leakage via access patterns

- Deterministic + order preserving encryption: e.g. CryptDB [PRZB'11]
  - Practical but significant leakage (see Seny Kamara's talk)

Deterministic and order preserving

Name	Lastname	Age
Elaine	Samuels	24
Mary	Stein	37
Jim	Stein	81
John	Sommers	3
Mary	Williams	17
John	Garcia	43
John	Gould	52

Name	Lastname	Age
Ge5$#u	Q*6sh#	223
E89(%y	$\boxed{\text{2@\#3Br}}$	340
2Tr^#7	$\boxed{\text{2@\#3Br}}$	736
$\boxed{\text{qM@9*h}}$	gYv6%t	34
E89(%y	X%3oL7	160
$\boxed{\text{qM@9*h}}$	wnM7#1	308
$\boxed{\text{qM@9*h}}$	8vy8$Z	475

SSE before 2013

- Generic tools: FHE, ORAM, PIR
  - very expensive,
  - great* security
    - *assumes *all* raw data is ORAM-encrypted, o/w leakage via access patterns

- Deterministic + order preserving encryption: e.g. CryptDB [PRZB'11]
  - Practical but significant leakage (see Seny Kamara's talk)

- Name of the game: Security-Functionality-Performance

Tradeoffs

SSE before 2013 (cont.)

- Dedicated SSE solutions:
  - Single-Keyword Search (SKS) [SWP'00, Goh'03, CGKO'06, ChaKam'10, …]
    - "privacy optimal" (if we don't count encrypted query results as leakage)
  - Conjunctions: Very little work
    - naive (n single-keyword searches),
    - GSW'04: structured-data, LINEAR in DB, communication-pairings tradeoff
- Practicality limitations
  - single-keyword only support, limited support for dynamic data
  - non-scalable design (esp. adaptive solutions), no I/O support for large DBs
  - little experimentation/prototyping

This work: Extends SSE in 4 dimensions

1. Functionality (well beyond single-keyword search):

   - Conjunctions
   - General Boolean expressions (on keywords)
   - Range queries
   - Substring/wildcard queries, phrase queries

   Search on *structured data* (relational DBs) as well as *free text*

2. Scalability:

   - *terabyte*-scale DB, *millions* documents/records, *billions* indexed document-keyword pairs
   - *Dynamic* data
   - Validated implementation, tested by a third party (IARPA, Lincoln Labs)

3. Provability: "imperfect security" but with provable leakage profiles (establishing upper bounds on leakage), well-defined adversarial models

This work: extends SSE in 4 dimensions

4. New application settings and trust models

- Multiple clients: Data owner D outsources Encrypted DB to cloud; clients run queries at the cloud but *only for queries authorized* by D
  - Leakage to cloud as in basic SSE, client only learns documents matching authorized queries (policy-based authorization enforced by data owner)
- Blind authorization: As above but authorizer enforces policy without learning the queried values (we call it "*Outsourced Symmetric PIR*")
  - Assumes non-collusion between cloud and data owner

- Note: multi-reader, single-writer system (no public key encryption)

Example Applications

- Example: Hospital outsources DB, provides access to clients (doctors, administrators, insurance companies, etc.)
  - Policy-based authorization on a client/query-basis
  - Hospital doesn't need to learn the query, only (blindly) enforce policy
    - Good for security, privacy, regulations
- Warrant scenario (extended 4-party setting)

  Obama's 3rd Party Solution (phone data)

  - Judge provides warrant for a client C (e.g. FBI) to query a DB
  - DB owner enables access but only to queries allowed by judge
  - DB owner does not learn warrant content or queries
  - Client C (e.g., FBI) gets the matching documents for the allowed queries and nothing else

Large-Scale & Functional Implementation

- Support for arbitrary Boolean queries on all 3 (extended) SSE models
- Validated on synthetic census data: 10Terabytes, 100 million records, > 100,000,000,000=10^{11} indexed record-keyword pairs !
  - Equivalent to a DB with one record for each American household and 1000 keywords in each record and any boolean query (including textual fields)
  - Smaller DB's: Enron email repository, ClueWeb (>> English Wikipedia)
- Support for range queries, substring/wildcards, phrase queries (5x perf. cost)
- Dynamic data: Supports additions, deletions and modifications of records

Scalability

- Preprocessing scales linearly w/ DB size (minutes-days for above DBs)
  - Careful data structure, crypto and I/O optimizations
    - Can benefit on any improvement on single-keyword search
- Search proportional to # documents matching the least frequent term: $w_1 \wedge B(w_2,\dots, w_n)$
  - *Single round* to retrieve matching document indexes (tokens from client to server, matching indices back; retrieve encrypted documents)
  - Query response time: Competitive w/ plaintext queries on indexed DB

4 seconds: fname='CHARLIE' AND sex='Female' AND
NOT (state='NY' OR state='MA' OR state='PA' OR state='NJ)
on 100M records/22Billion index entries US-Census DB

Crypto Design-Engineering Synergy

- Major effort to build I/O-friendly data structures
  - Critical decision: Do not design for RAM-resident data structures (it severely *limits scalability*)
  - Challenge: need to avoid random access (e.g., avoid Bloom filters on disk)
    - Need *randomized* data structures to reduce leakage and need *structured* ones to improve I/O performance (locality of access)
- Cryptographic index based on elliptic curve cryptography (optimized for very fast exponentiation, esp. with same-base)
  Typically: I/O and network latency dominate cost

  500,000/sec, 8 cores, same-base opt , 100-1000 per IO

  - On a midsize storage system: ~300 IOPS (I/O Operations Per Second)
  - ~1000 expon's per random I/O access (133 w/o same-base optimization)
- Data encryption uses regular symmetric crypto (e.g., AES)

Security: The challenge of being imperfect

- Good news: Semantic security for data; no deterministic or order preserving data encryption
- But: Security-Performance trade-offs → Leakage to server
  - Leakage in the form of access patterns to retrieved data and queries
    - Data is encrypted but server can see intersections b/w query results (e.g. identify popular document, intersection b/w results of two ranges, etc.)
    - Server learns query function (not values/attrib's); identifies repeated query
  - Additional specific leakage (more complex functions of DB and query history):
    - E.g. we leak $|Doc(w_1)|$ and in query $w_1 \wedge w_2 \wedge \ldots \wedge w_n$ we leak $|Doc(w_1 \wedge w_i)|$
    - E.g. the server learns if two queries have the same w_1 (other terms are hidden)
- Leads to statistical inference based on side information on data (effect depends on application), masking techniques may help

Security: The challenge of being imperfect

- Security proofs: Formal model and precise provable leakage profile
  - Leakage profile: provides upper bounds on what's learned by the attacker
  - Security modeling and definitions follow simulation paradigm [CGKO, CK]
- Syntactic leakage vs "semantic leakage"
  - Need to assess on an application basis and relative to a-priori knowledge
  - For example, formal leakage proven even if attacker can choose data and queries – but in practice, in this case, semantic leakage will be substantial
  - Yet, we expect in many cases to provide meaningful (if imperfect) security (in particular, relative to property-preserving solutions)
- Detour: Is CryptDB sufficient in practice? Who is the attacker? Enough to not being the weakest link? What do regulations say?

Security Formalism (adversarial server)

- Based on the simulation-based definitions given for SKS [CGKO,CK].
- There is an attacker E (acting as the server), a simulator Sim and a *leakage function* L(DB, queries):
  - Real: Attacker E chooses DB and gets the pre-processed encrypted DB, then interacts with client on adaptively chosen queries
  - Ideal: Attacker E chooses DB and queries (adaptively), E gets Sim(L(DB)) and Sim(L(DB,queries))

  A SSE scheme is *semantically secure with leakage L* if for all attackers E, there is a simulator Sim such that the views of E 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

Basic ideas

- Focus on conjunctions $w_1,\ldots,w_n$ (will be extended to Boolean queries)

1. Choose the *least frequent* conjunctive term, say w_1 ("s-term"), find encrypted indexes of documents containing w_1 (w/o revealing w_1)
   - Pre-computed encrypted index stored at Eddie (part of EDB): $\forall$ w, Enc(w) → Enc(ind_1), Enc(ind_2), … , Enc(ind_k)
2. For each w_j and ind_i, check if w_j appears in ind_i.
   - Uses an "oracle" that given Enc(ind) and Enc(w) says if keyword w appears in document ind (without revealing ind or w)
   - Oracle implemented as a function H(ind,w) and a set Hset stored at the server of all values H(ind,w) such that w appears in record ind
   - Server computes H(ind,w) jointly (and "non-interactively") with client; server does not learn w or ind (it is encrypted), client learns nothing
     - computation based on DH-based Oblivious PRF

Columbia/Bell Labs Solution (Blind Seer)

- Parallel work: Same IARPA project – papers at [Oakland'14, 15]
- Elegant solution based on Bloom filter trees with Garbled Yao for privacy and authorization
  - Conceptually simpler than ours
  - Uses MPC techniques (Yao) instead of homomorphic operations
  - Less scalable: Bloom filters are *inherently* random access
    →DB sizes limited by the size of RAM
  - Single client
  - Incomparable leakage (e.g., Bloom filter path vs. w_1-related leakage)

Research Questions

- Leveraging other tools (carefully): MPC, ORAM, homomorphic encryp'n
- Fundamental limits (leakage-computation tradeoffs), e.g.:
  - leakage from returned ciphertexts (ORAM helps but at significant cost)
  - Frequency of w_1 (least frequent term) (reduction from 3SUM)
- "Semantic leakage": Proving formal leakage is nice but how bad is it for a given particular application, what forms of masking can help?
  - Can we have a theory to help us reason about it (cf. differential privacy)?
  - A theory of leakage composition? Guidance for masking techniques
  - Attacks welcome! (Also easier to get accepted to conferences ☺)
- Characterizing *privacy-friendly* plaintext search algorithms/data str.
- A more complete SQL query set (esp. joins)

Summary

- Great progress relative to work on single-keyword single-client SSE
  - Rich queries: General Boolean queries (structured data, free text), Plus: range, substring, wildcards, phrase, proximity
  - Huge DBs: 10 TB, 100M records, 10^{11} indexed keyword-document pairs
    - EDB creation linear in DB size, queries competitive with MySQL
  - Single- and Multi-Client models, policy-based delegation of queries
  - Authorization w/o learning query ("Outsourced Symmetric PIR")
  - Privacy, insider security, surveillance protection, warrant enforcement
- Imperfect security: Leakage from access- and query-patterns, but well defined leakage profiles, and simulation-based adaptive security
- Many challenging theoretical and engineering questions
  - Going for practice? Don't forget simplicity, engineering and… proofs!

Join the (multi) Party…

- *An exciting & large space to explore with many many research opportunities!*
- *… and many practical applications*
  - Very timely given cloud migration, explosion of private info, and strong attackers (including surveillance, espionage, mafia, and just hackers…)
- *An opportunity for sophisticated crypto in the real world?*

Thanks!

- Crypto'2013: Boolean search, single client   eprint.iacr.org/2013/169
- CCS'2013: Multi-client, Blind authorization   eprint.iacr.org/2013/720
- NDSS'2014: Dynamic data, implementation   eprint.iacr.org/2014/853
- ESORICS 2015: Range, Substrings, Wildcards, Phrases   2015/927