# Secure storage in the cloud using property preserving encryption

Kenny Paterson

Information Security Group

ROYAL HOLLOWAY UNIVERSITY OF LONDON

# Overview

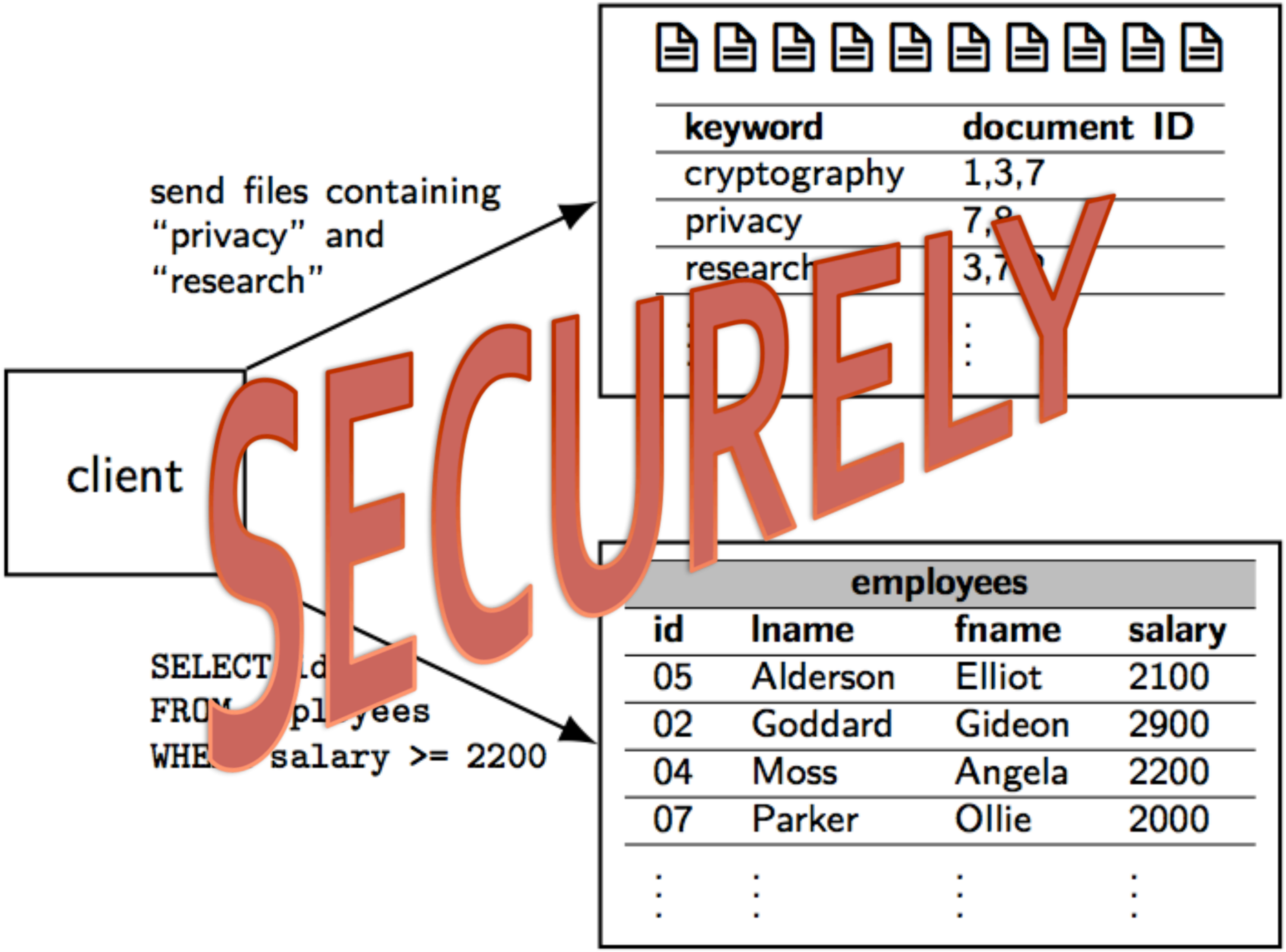
1. Application scenarios.
2. Deterministic encryption and search.
3. OPE/ORE and range queries.
4. Analysing access pattern leakage from range queries.

# Application scenarios

# Application Scenarios

- Data owners wish to **securely** outsource storage to cloud providers whilst preserving capability for users to query data in various ways.
- What kinds of queries?
- What kinds of users?
- What kinds of data?
- What kinds of query?
- What kinds of adversary?
- Meta: Why not just use FHE and be done?

# Two scenarios, one picture



# Scenario 1: Searchable File Storage

- Owner has large collection of files, indexed by keywords.
- Owner **encrypts** files and stores these on remote server.
- Owner **encodes** keywords in such a way that keyword searches can still be carried out.
- Encoded keywords also stored on server, as an encoded **index.**
- Owner sends **search token** to server; server uses token and index to find identifiers for matching files.
- Matching file identifiers are returned to owner.

# Scenario 2: Database Encryption

- Data owner has a large database of records; each record has multiple fields.

- Owner encrypts data in each field in such a way that **standard database queries** can still be carried out.

- **Basic**: simple searches.
  - "Give me all records in which surname = Dubois".
- **Advanced**: compound searches.
- **More advanced**: range queries
  - "Give me all records with ages between 21 and 30".
- **Finally**: arbitrary SQL queries.*



*Other db query languages are available.

# Searchable Encryption

**Solution for Scenario 1: Searchable Encryption.**

- Naïve scheme: owner uses IND-CPA symmetric encryption for files and $\mathrm{PRF}_K(kw)$ as encoding of keyword *kw*.

- Store encrypted files and encoded keywords per file on server.

- Owner sends $tok = \mathrm{PRF}_K(kw)$ to server; server matches *tok* against encoded keywords; returns matching files.

- Can use an inverted index and file identifiers: server stores database of tuples $(tok, (fid_1, fid_2, \ldots))$.

# Security Analysis

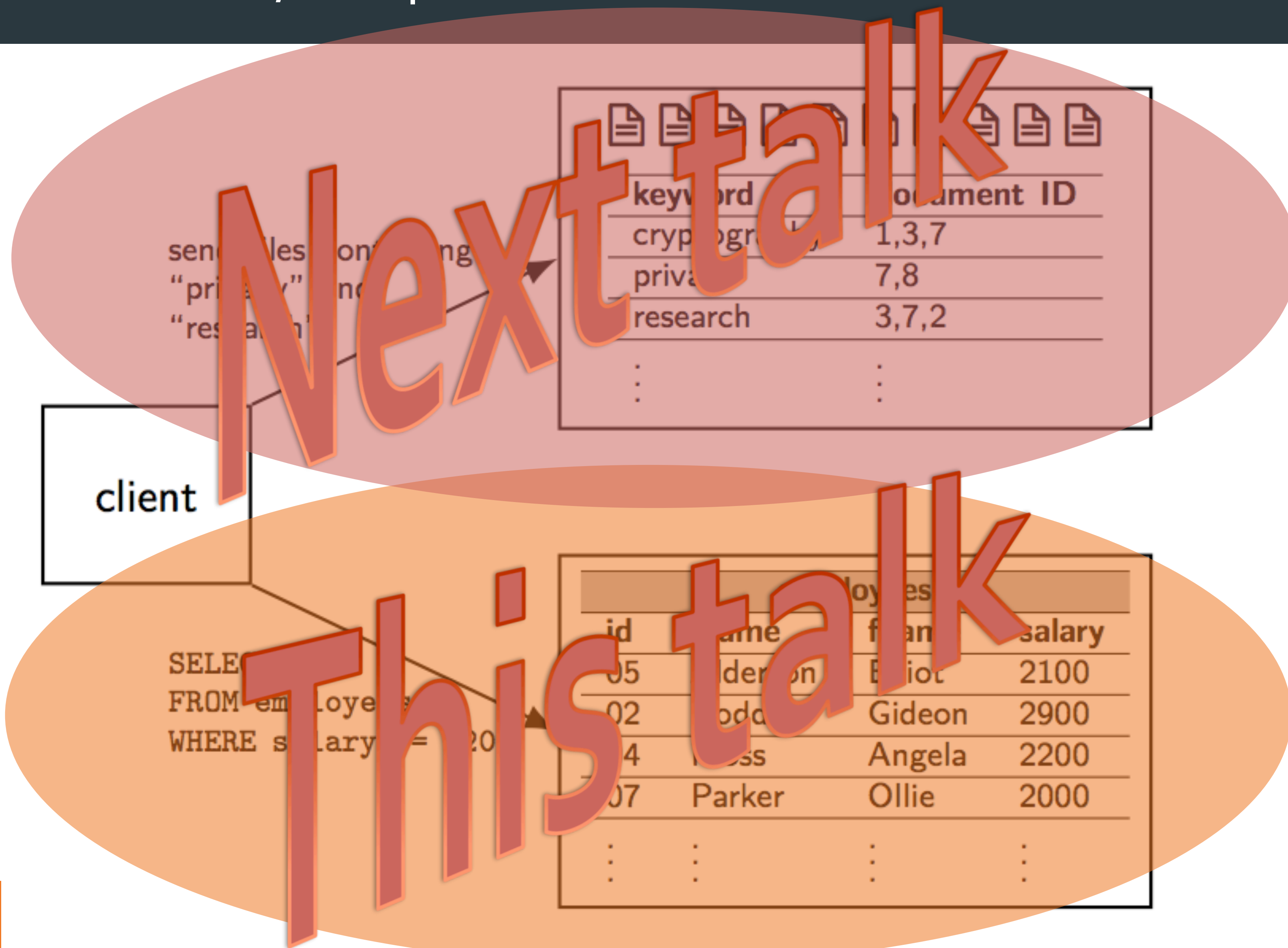
- Adversarial objectives?
  - Keyword recovery, recovery of file contents,… ?

- Adversarial capabilities?
  - "Snapshot", "Honest-but-curious", "Fully malicious".
  - Can/cannot **observe** queries; can/cannot **make** queries; can/cannot **inject** files.

- What about auxiliary information?
  - What if the adversary has a representative data sample or keyword sample?

- Cash *et al*. (CCS15): detailed analysis of different attacks models, leakage profiles, etc. against **SE schemes** in general: **Leakage Abuse Attacks**.
- Fuller *et al.* (S&P17): SoK paper on cryptographically protected database search.

# Two scenarios, one picture



# Deterministic Encryption

**Partial solution for Scenario 2: DE**

- Simplest possible scheme: owner uses deterministic encryption scheme (KGen, Enc, Dec) to encrypt each column of the database using a per-column key *K.*
- Server can store the encrypted data on server in a traditional database.
- To find matches with value $x$ in a column, send search query for $y = \mathrm{Enc}_K(x)$ to server.
- Server finds matches on $y$ and returns full encrypted records to client.
- Client decrypts returned records using per column keys.
- Use of DE **preserves equality** of plaintexts and allows simple searches.
- (Very similar to naïve SE, with PRF replaced by Enc/Dec).

# Property Preserving/Revealing Encryption (PPE/PRE)

**More general solution for Scenario 2: PPE/PRE**

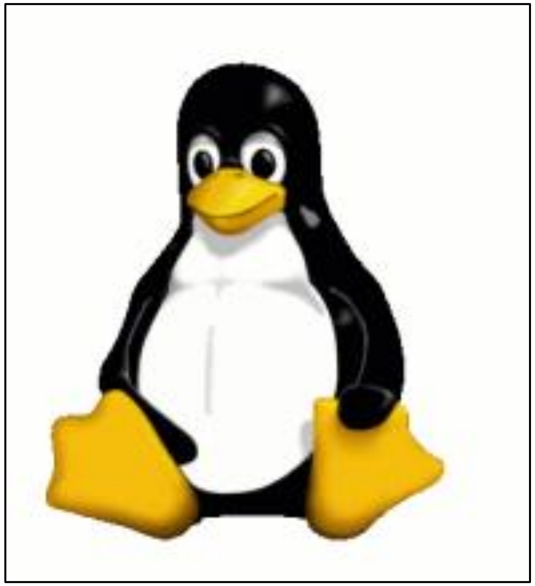
- Generalises idea of “equality preserving/revealing” property of DE.
- Main example: Order Preserving/Revealing Encrypion (OPE/ORE).
- **OPE: if $x < y$ then Enc($x$) < Enc ($y$).**
- **ORE**: there exists a (public) efficiently computable function “Order” such that:

$$x < y \quad \textbf{iff} \quad \textbf{Order(Enc(}x\textbf{), Enc(}y\textbf{)) = 1}$$

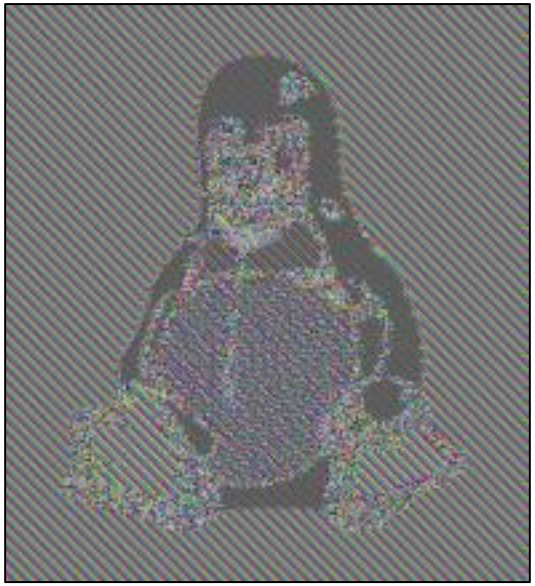
- OPE/ORE allows range queries!
- Client who wishes to query on range [$a$,$b$] instead sends query for range [Enc($a$), Enc($b$)] to server.

# Analysis of Deterministic Encryption

# Reminder: ECB information leakage



Tux the Penguin, the Linux mascot. Created in 1996 by Larry Ewing with The GIMP. lewing@isc.tamu.edu



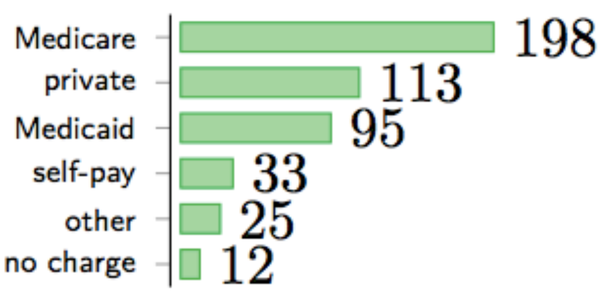
ECB-Tux

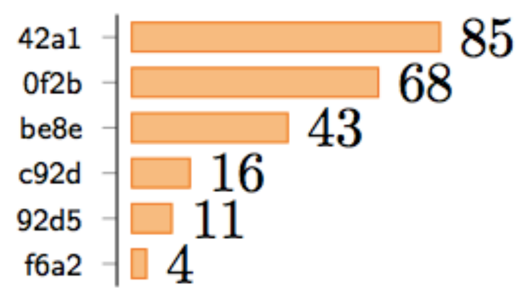
# Analysis of Deterministic Encryption

- DE is equality preserving, by design.
- DE therefore preserves frequencies of plaintexts in the ciphertexts, cf. monoalphabetic substitution cipher.
- Naveed-Kamara-Wright (CCS15): let's apply frequency analysis! (al-Kindi, 9th century.)
- **Assumption 1**: attacker has **auxiliary information** – a reasonably accurate estimate for the plaintext distribution.
- **Assumption 2**: attacker has a **snapshot** of the encrypted database.

# Analysis of Deterministic Encryption


| 135a0e |
| --- |
| · · · **7aec84** · · · |
| c92d |
| 0f2b |
| 42a1 |
| be8e |
| 92d5 |
| 42a1 |
| 0f2b |
| 42a1 |
| ⋮ |



# Frequency Analysis is Maximum Likelihood!

- Given a column of ciphertexts $\boldsymbol{y}$, frequency analysis matches:
  - Most frequent item in $\boldsymbol{y}$ with most frequent item in aux. dist.
  - Second most frequent item in $\boldsymbol{y}$ with second most frequent item in aux. dist.
  - etc.
- Defines a permutation $\boldsymbol{\pi}$ mapping plaintexts $x$ to ciphertexts $y$.
- This procedure is **maximum likelihood**, that is, it maximises the likelihood

$$L(\pi \mid \boldsymbol{y}) := \Pr(\boldsymbol{y} \mid \pi).$$

- **Proof**: fun exercise, see also eprint 2015/1158.

# Performance of Frequency Analysis Against DE

- Naveed-Kamara-Wright [CCS15] performed an **empirical investigation** of the performance of frequency analysis against DE.

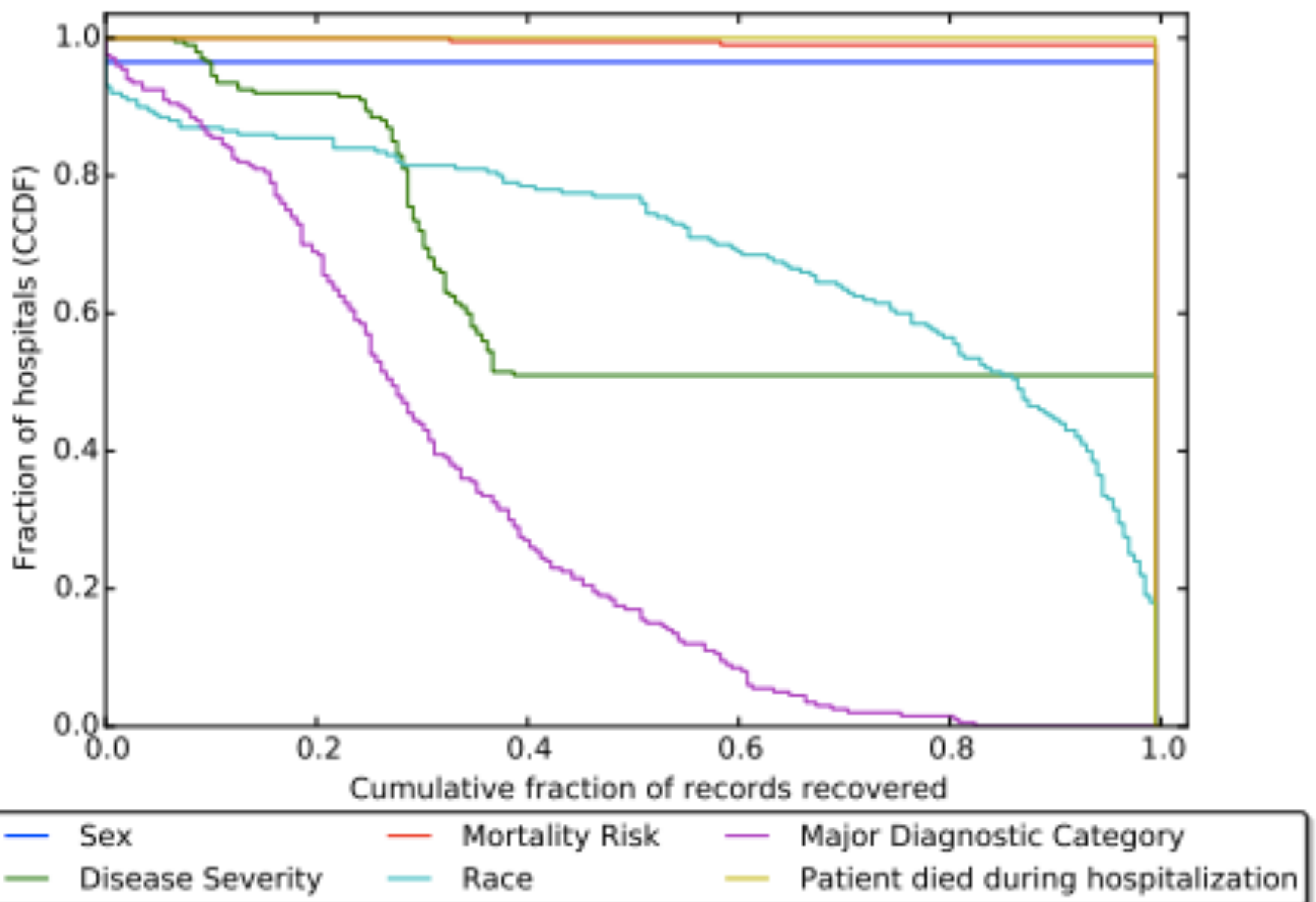
- Using **a large medical dataset**: per-patient data in 12 categories for 200 largest hospitals in the 2009 Nationwide Inpatient Sample (NIS), from the Healthcare Cost and Utilization Project (HCUP), run by the US Agency for Healthcare Research and Quality.

- DE encrypt data per hospital for each category.
- Use 2004 **aggregated** HCUP data as the auxiliary data.
- Run frequency analysis and measure percentage of data items correctly recovered **per hospital**.

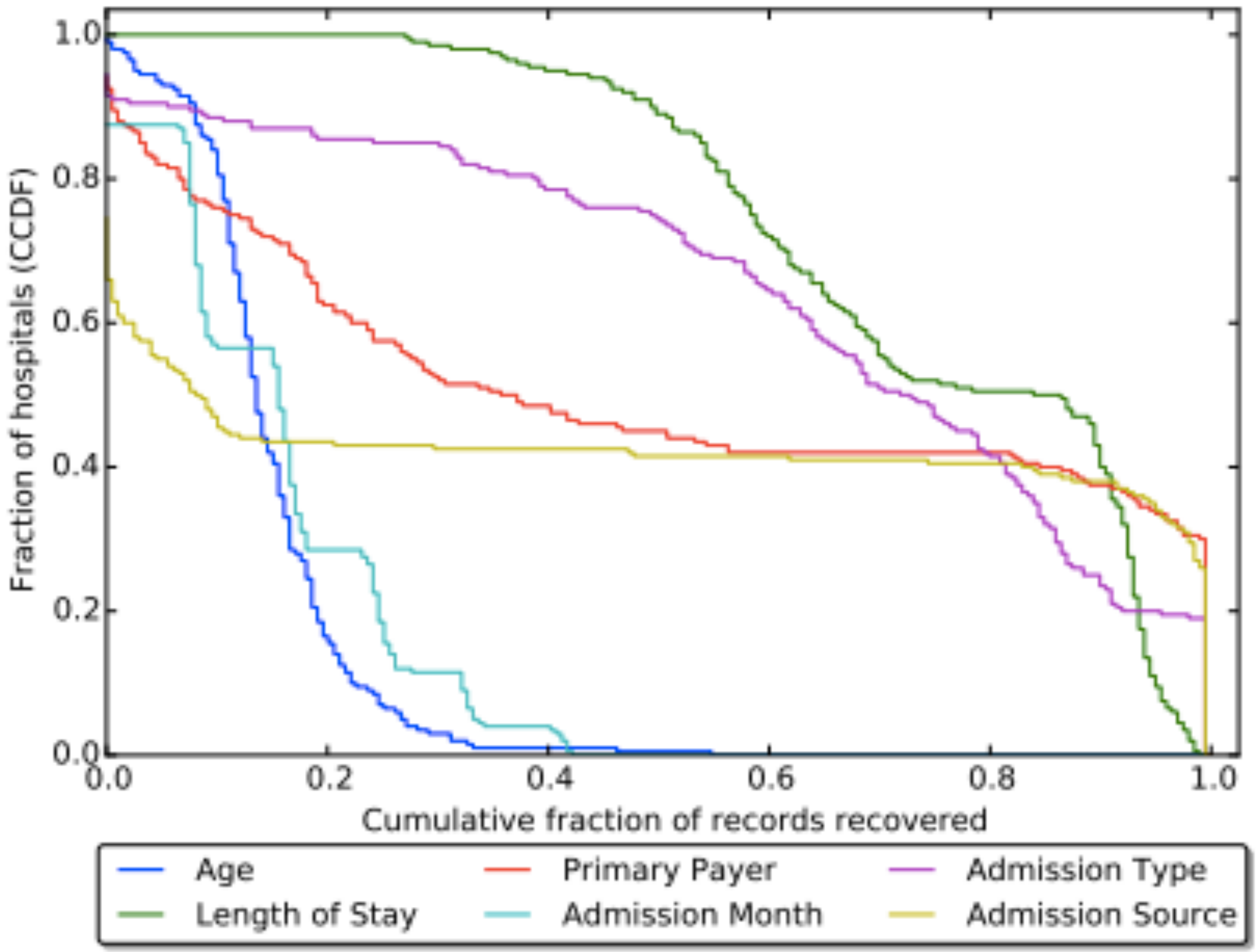
# Performance of Frequency Analysis Against DE

| Attribute | Num. values |
|---|---|
| Age (AGE) | 125 |
| Admission month (AMONTH) | 12 |
| Admission source (ASOURCE) | 5 |
| Admission type (ATYPE) | 6 |
| Patient died (DIED) | 2 |
| Sex (FEMALE) | 2 |
| Length of stay (LOS) | 365 |
| Major diagnostic category (MDC) | 25 |
| Primary payer (PAY1) | 6 |
| Ethnicity group (RACE) | 6 |
| Disease severity (APRDRG_Severity) | 4 |
| Mortality risk (APRDRG_Risk_Mortality) | 4 |

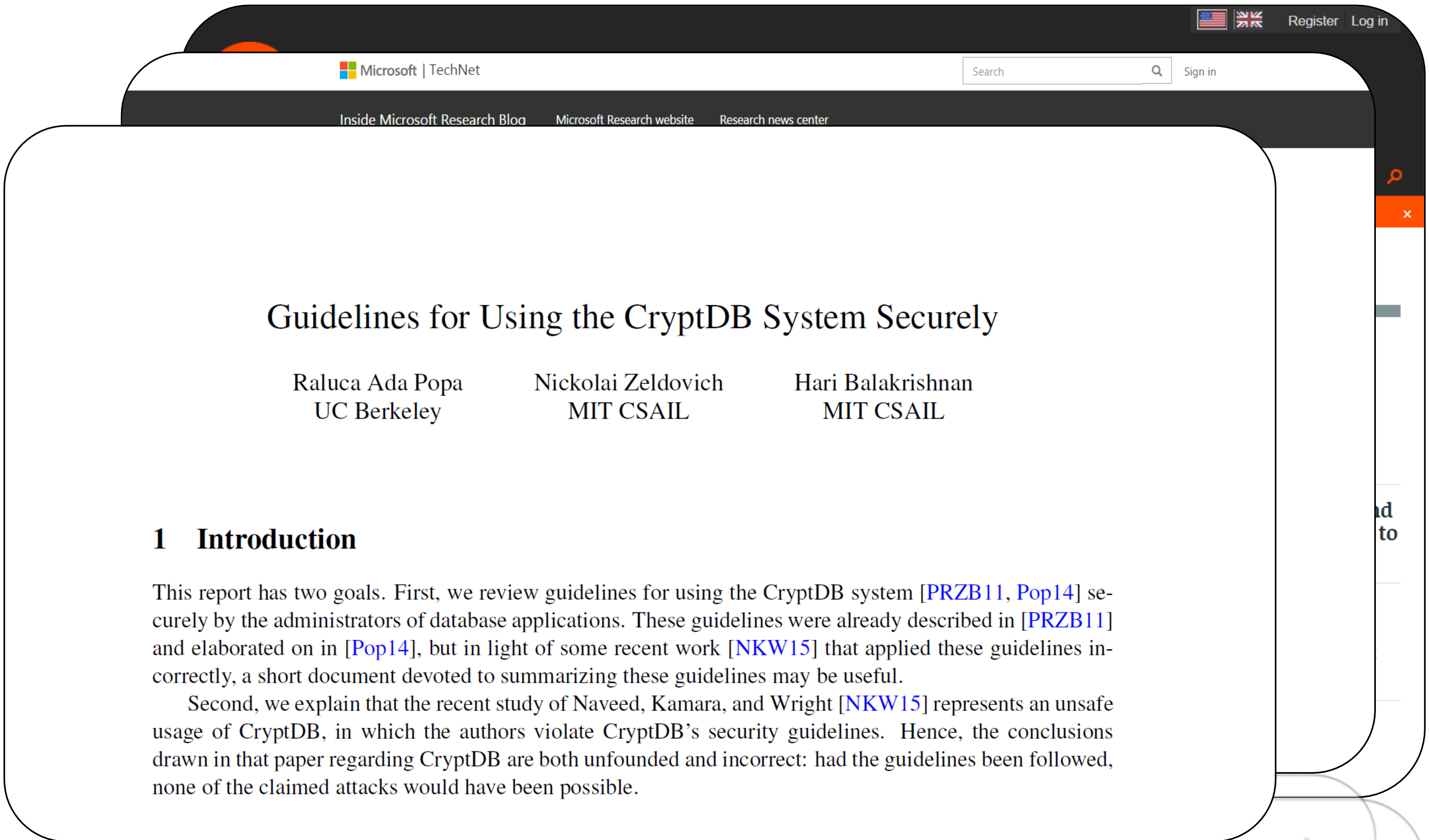
# Performance of Frequency Analysis Against DE



# Performance of Frequency Analysis Against DE



# Frequency Analysis Makes Headlines!




## Guidelines for Using the CryptDB System Securely

Raluca Ada Popa, UC Berkeley
Nickolai Zeldovich, MIT CSAIL
Hari Balakrishnan, MIT CSAIL

### 1 Introduction

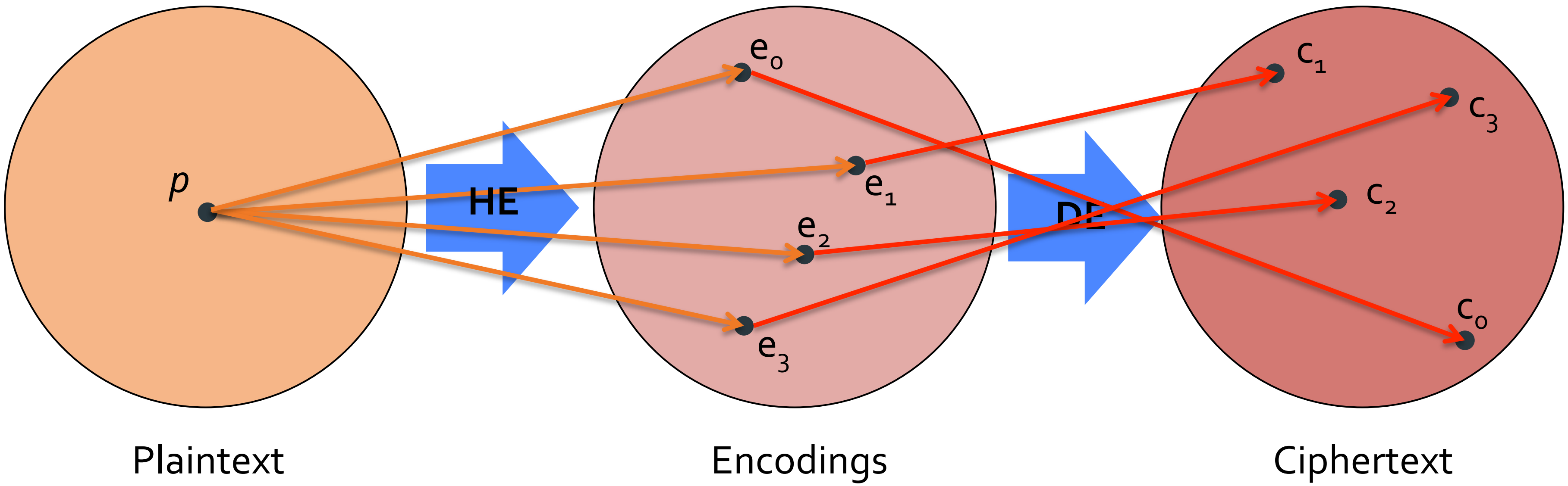
This report has two goals. First, we review guidelines for using the CryptDB system [PRZB11, Pop14] securely by the administrators of database applications. These guidelines were already described in [PRZB11] and elaborated on in [Pop14], but in light of some recent work [NKW15] that applied these guidelines incorrectly, a short document devoted to summarizing these guidelines may be useful.

Second, we explain that the recent study of Naveed, Kamara, and Wright [NKW15] represents an unsafe usage of CryptDB, in which the authors violate CryptDB's security guidelines. Hence, the conclusions drawn in that paper regarding CryptDB are both unfounded and incorrect: had the guidelines been followed, none of the claimed attacks would have been possible.

# Combatting Frequency Analysis

- We want to smooth out frequency distribution so that frequency analysis becomes ineffective.
  - Performing worse than random guessing of plaintext.
- We also want to preserve ability to efficiently perform search queries on a standard database.
  - Rules out fully randomised/IND-CPA secure encryption.
- What about adding a limited amount of randomness?
- Leads to idea of applying **homophonic encoding** to produce **Frequency Smoothing Encryption (FSE)** schemes (Lacharité-Paterson, forthcoming).

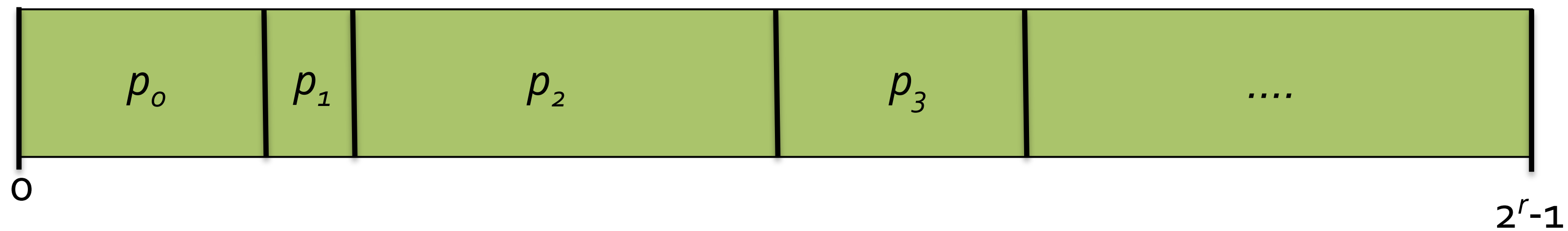
# Frequency Smoothing Encryption – Combatting Frequency Analysis




- Homophonic Encoding (HE) consumes **small** amount of randomness.
- Make number of encodings proportional to frequency of *p* for good frequency smoothing.
- DE = Deterministic Encryption.
- Match on $\{c_1, c_2, c_3, c_4\}$ instead of a single ciphertext.
- Query complexity blow-up by max. number of encodings in worst case.

# Interval-based Homophonic Encoding (IBHE)

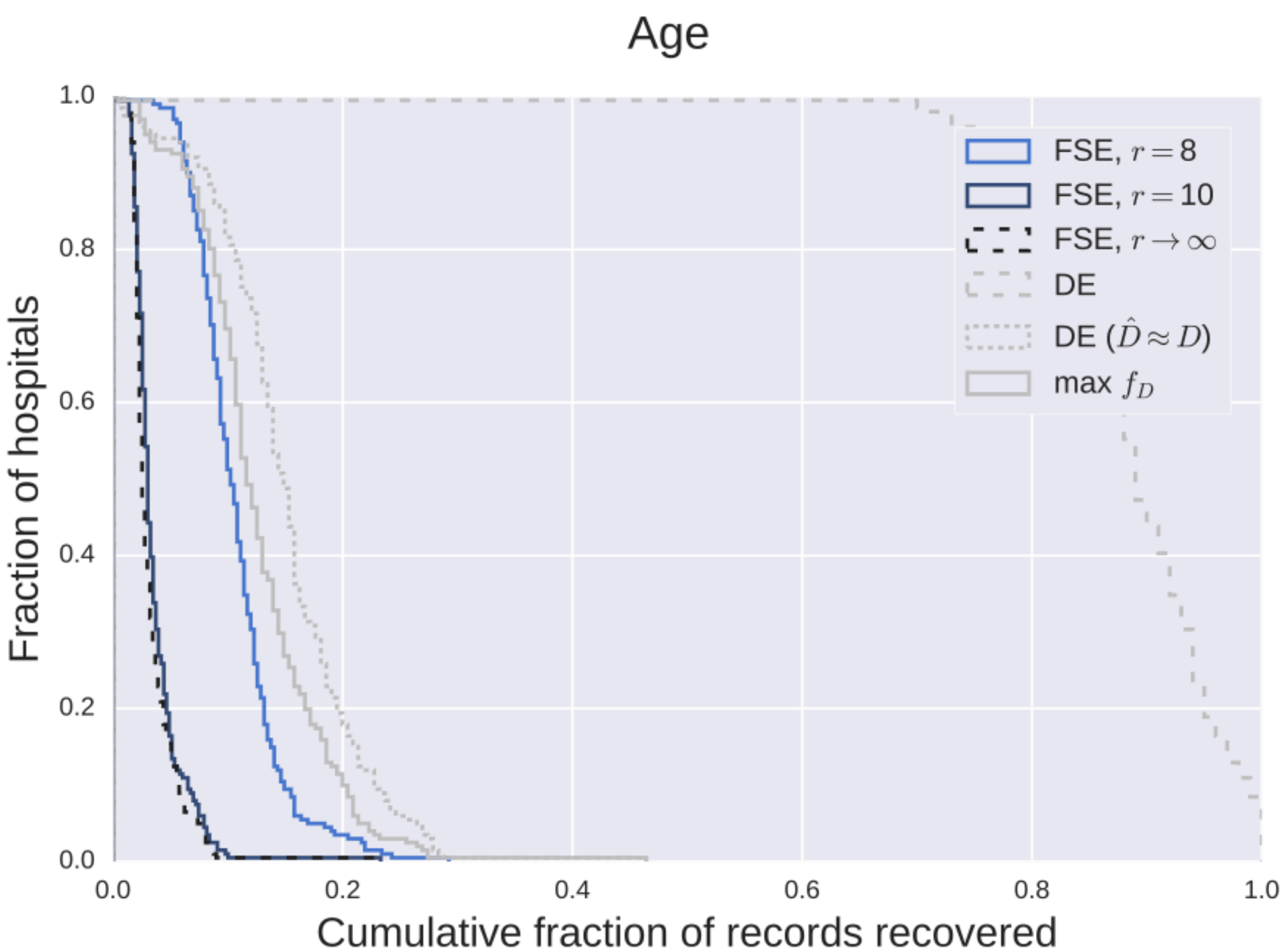
- Encoding space = $r$-bit strings / interval $[0, 2^r)$.
- Represent encodings of $p$ having frequency $f$ by an interval of size approximately $f \times 2^r$.
- Select uniformly at random from interval to encode $p$.
- Needs an encoding table to store an interval for each plaintext item; $|p| \times 2r$ bits.
- Also needs a decoding table mapping bits back to plaintexts.



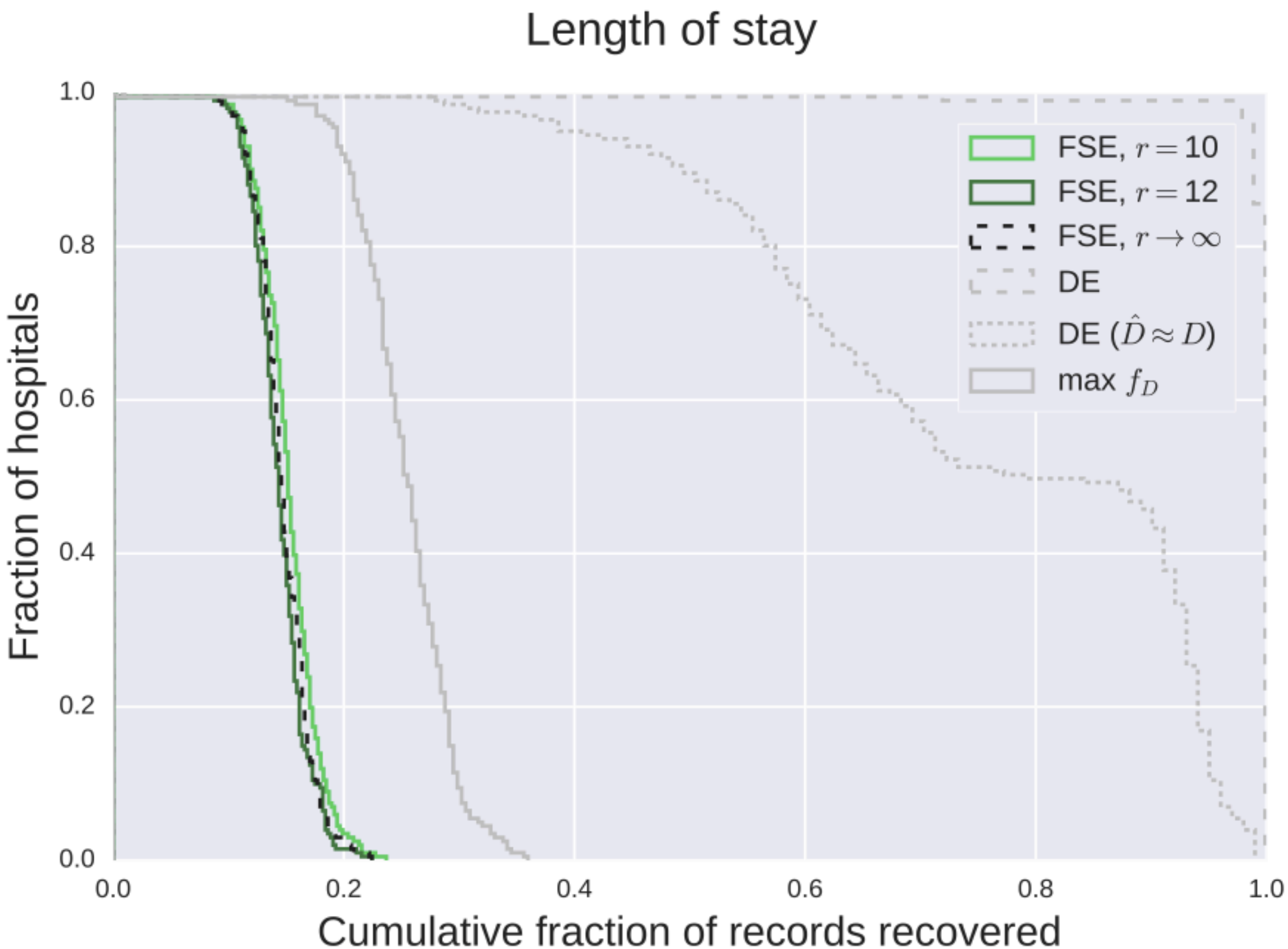
# Effectiveness of FSE from IBHE + DE

- Can prove that as r goes to $\infty$, no distinguisher can tell apart ciphertexts from uniformly random strings.
- But even for moderate *r*, IBHE + DE smooths well for all but very skewed data.
- Rapidly limits (generalised) frequency analysis to being worse than a pure guessing attack.
  - Such an attack is always possible for limited domain of plaintexts.
- We used same evaluation framework as Naveed-Kamara-Wright (CCS15).
  - Except that we gave the adversary the **exact, per-hospital distribution** as the auxiliary distribution!

# Effectiveness of FSE from IBHE + DE



# Effectiveness of FSE from IBHE + DE



# Effectiveness of FSE from IBHE + DE

- **Warning**: FSE only protects against a basic snapshot attacker.
- Recent work of Grubbs-Ristenpart-Shmatikov (HotOS17) questions legitimacy of snapshot attack model.
- Columns are treated in isolation.
- More powerful adversary could perform frequency analysis on the sets of **responses** to queries.
- Scheme does not protect against an active attacker who can inject his own queries.

# Analysis of OPE/ORE

# Order Preserving/Revealing Encryption

- **OPE: if $x < y$ then Enc($x$) < Enc ($y$).**

- **ORE**: there exists a (public) efficiently computable function "Order" such that:

$$x < y \quad \text{iff} \quad \mathbf{Order(Enc(}x\mathbf{), Enc(}y\mathbf{)) = 1}$$

- OPE/ORE allows range queries.

- Client who wishes to query on range [$a$,$b$] instead sends query for range [Enc($a$), Enc($b$)] to server.

# Order Preserving/Revealing Encryption

- Q: If DE leaks badly, does OPE/ORE leak even more?
- A: Often, yes.

- **Folklore**: if OPE scheme is deterministic and plaintext data is **dense** (every possible plaintext occurs) then a snapshot adversary can learn which plaintext is which.
- Simply order the ciphertexts and then read off the plaintexts.
- **Take-away**: beware of formal security models for OPE/ORE.

- This can sometimes be generalised to the non-dense case…

# The Scheme of Chenette-Lewi-Weis-Wu (FSE16)

- CLWW (FSE16) presented a clever and practical ORE scheme built using only PRFs.
- CLWW gave a precise characterisation of leakage in a simulation-based security model:
  - Given two ciphertexts Enc(x) and Enc(y), the scheme leaks exactly the first index at which bits of *x* and *y* differ (and which is bigger).
- **Example**: given Enc($x = 1101_2$) and Enc($y = 1001_2$), the scheme would leak that the two plaintexts are equal in MSB (bit 0) but that the first one has 1 in bit 1 and the other 0 in bit 1.
- Leakage is **greater** than in an ideal OPE scheme, which would leak only order.

# An Attack on the CLWW Scheme

- In the dense case, the folklore analysis applies.
- What about the non-dense case?

- **Assumption 1:** snapshot attacker.
- **Assumption 2:** $N$ plaintexts, close to uniformly random on the $s$ MSBs, where $N > s2^s$.

**Then, with high probability, the attacker can learn the *s* most significant bits of every plaintext.**

# An Attack on the CLWW Scheme

- **Assumption 1:** snapshot attacker.
- **Assumption 2:** $N$ plaintexts, close to uniformly random on the $s$ MSBs, where $N > s2^s$.

- Second assumption implies that, with high probability, every possible $s$-bit prefix occurs in at least one plaintext:

$$\text{Prob} \approx 1 - 2^{-s/2\wedge(s+1)}.$$

- Use the CLWW scheme's leakage to order the $N$ ciphertexts on the $2^s$ distinct $s$-bit prefixes.
- Now read off the $s$ most significant bits of each plaintext.

# Implications of the Attack

- Suppose a company has 10,000 employees with salaries that are 20-bit numbers (between \$0 and $\$2^{20}-1$).
- We can set $s = 10$ ($10 \times 2^{10} \approx 10{,}000$).
- Attack yields 10 MSBs of *every* salary.
- This is enough to identify each salary up to accuracy of \$1k.

- Example generalises to, say, 32-bit salaries that are all zero in the first 12 bit positions.
- Sufficient that data be dense in some positions (and constant in leading positions).

# Further Research on OPE/ORE Leakage

Several attack recent papers examine the real-world implications of the leakage of OPE/ORE schemes for snapshot attackers:

- **Durak-DuBuisson-Cash (CCS16):** attacks on correlated columns of OPE/ORE-encrypted data, especially longitude/ latitude data.
- **Grubbs-Sekniqi-Bindschaedler-Naveed-Ristenpart (S&P17):** revisit Naveed-Kamara-Wright for OPE/ORE; recast ptxt/ctxt matching problem as min-weight, non-crossing bipartite matching problem, solve it efficiently for many types of data, relies on auxiliary distributions.

# Access Pattern Leakage for Range Queries

# Analysis of Access Pattern Leakage for Range Queries

**Kellaris-Kollios-Nissim-O'Neill (CCS16):** analysis of access pattern leakage for SE; applicable to OPE/ORE schemes too.

- Honest-but-curious attack setting, stronger than snapshot adversary.
- **Assumption**: adversary can see which database rows are returned in response to any range query.
- For $N$-valued database, complete reconstruction in $O(N^4)$ queries.
- For dense case: $O(N^2 \log N)$ queries suffice.

# Analysis of Access Pattern Leakage for Range Queries

- Adversary in KKNO (CCS16) does not need to directly see the actual ranges queried.
  - In OPE/ORE, adversary would see only ciphertexts Enc(x), Enc(y) corresponding to range endpoints.
  - But in OPE/ORE setting, and in some SE schemes*, the **rank** also leaks.
  - The **rank** of a ciphertext is its position in an ordered list of all the ciphertexts.

*e.g. Arx scheme of Poddar-Boelter-Popa and FH-OPE scheme of Kerschbaum.

# Exploiting Rank in Analysis of Access Pattern Leakge

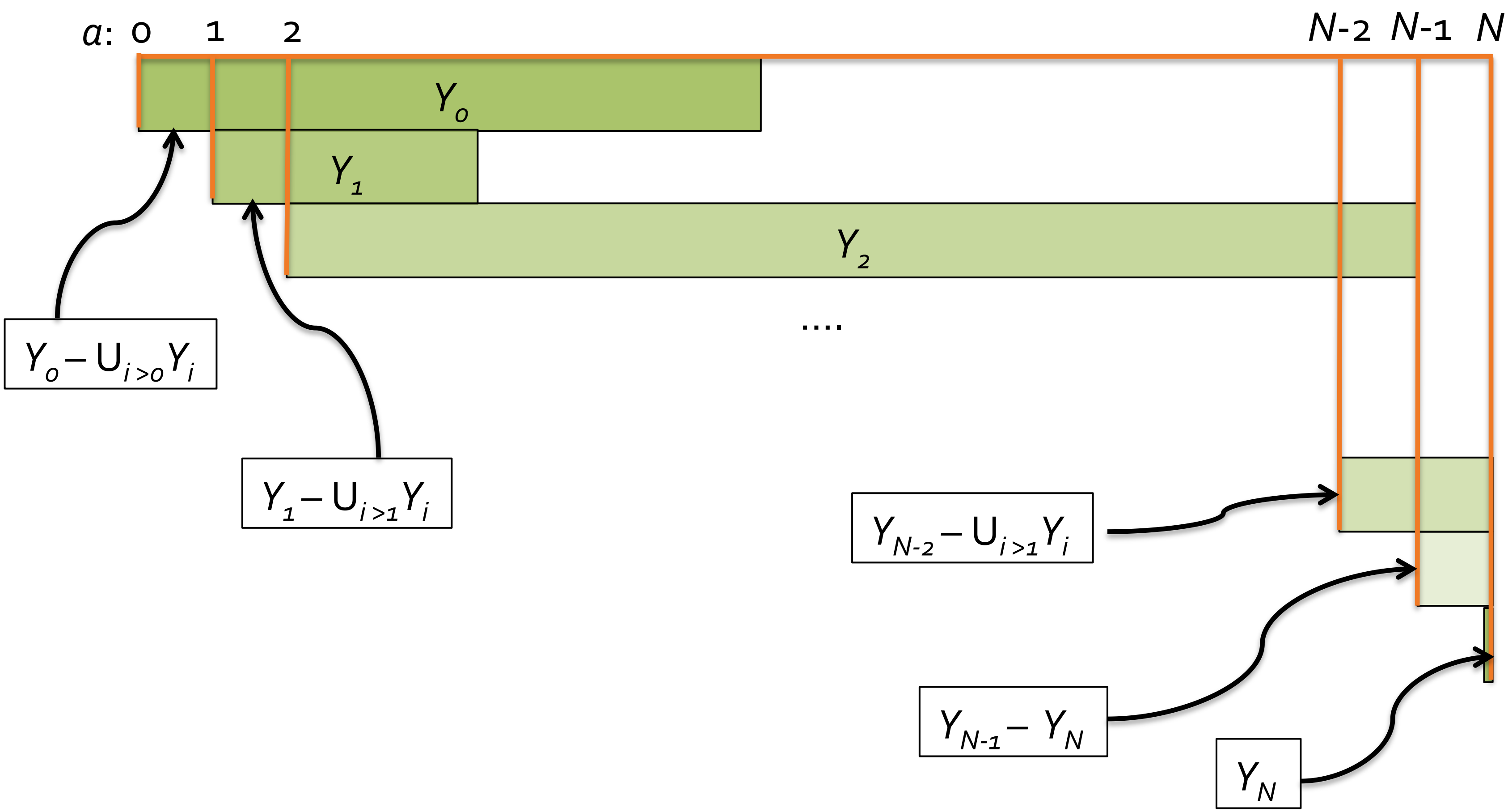
- Can we use the rank leakage to improve attack complexity?
- Lacharité-Minaud-Paterson (forthcoming):
  - Yes!
  - And much more besides…

# Exploiting Rank in Analysis of Access Pattern Leakge

**Simple motivating example: consider range queries [$a$,$b$] in which $a$ is uniformly random.**

- Then with probability $1 - 1/N$, after $2N\log N$ queries, all $N$ possible values for $a$ will have arisen.
  - Follows from standard analysis of the coupon collector problem.
- Easy to identify and order different $a$ values based on rank leakage.
- All values of $a$ in queries are now known.
- Pick out $N$ queries with distinct values of $a$; each such query produces a set of responses $Y_a$ (records in database).
- Then the set of records with value $a$ is: $Y_a - \bigcup_{i>a} Y_i$.

# Exploiting Rank in Analysis of Access Pattern Leakge



# Improved Analysis of Access Pattern Leakge

- Our simple example appears to show that rank leakage helps the adversary.
- In fact, we can **dispense with rank leakage** and obtain an $N\log N + O(N)$ attack in the general "dense" case!
  - Improving on KKNO's $O(N^2 \log N)$ attack.

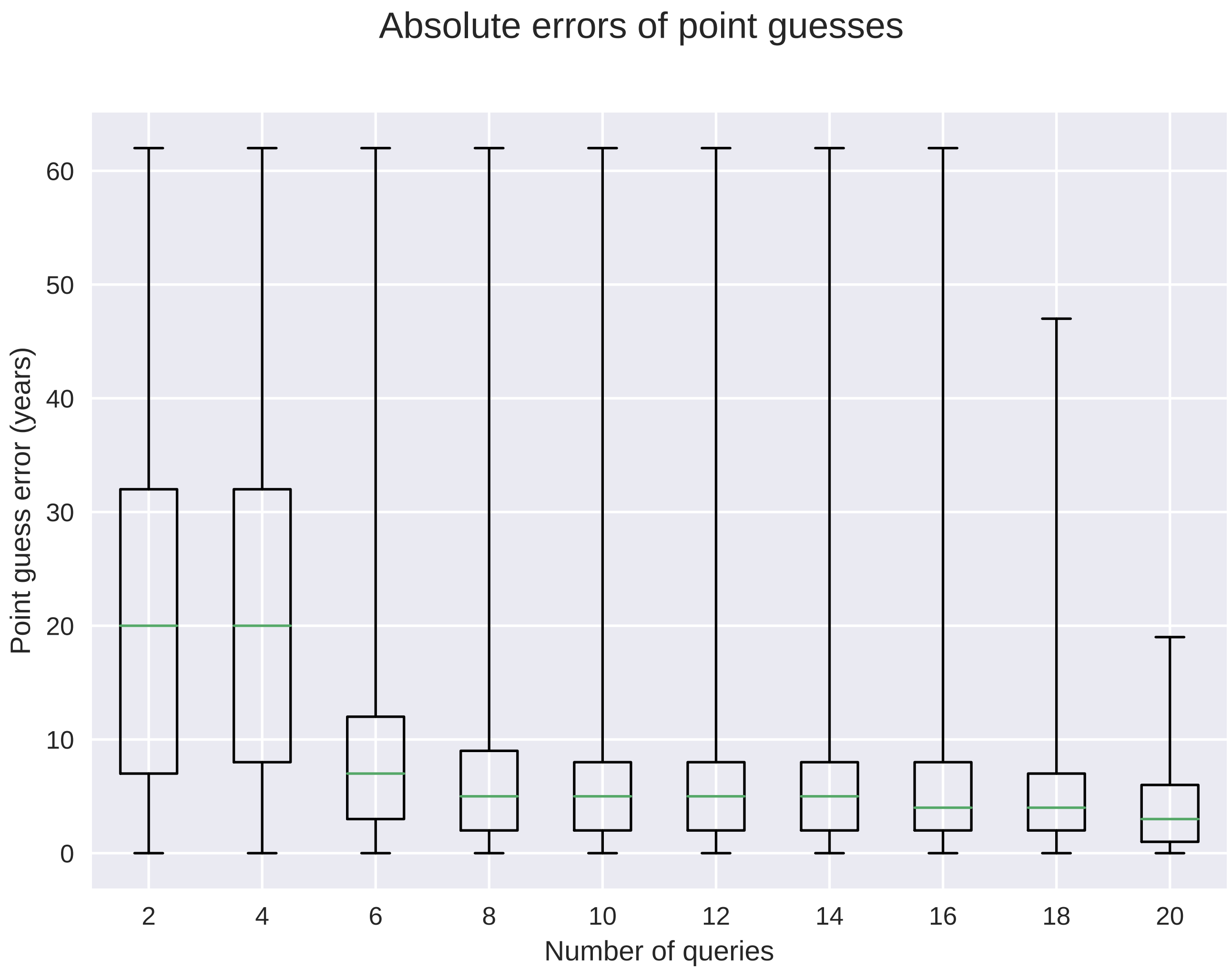
- We also consider the problem of **approximate reconstruction**.
- We can efficiently reconstruct values in records up to an absolute error of $\varepsilon N$ after seeing only $O(N)$ queries!
  - With a constant of $2\log(1/\varepsilon)$.

# Improved Analysis of Access Pattern Leakge

Finally, we study algorithms for **approximate reconstruction with the assistance of rank and an auxiliary distribution**.

- Significant reduction in number of queries required for accurate reconstruction.
- Perform set intersections and then map back to underlying data using rank + auxiliary distribution.
- Experiments with aggregated HCUP data…

# Approximate Reconstruction with Auxiliary Distribution



# Concluding Remarks

## Concluding Remarks

- Use DE/OPE/ORE with extreme care if at all.
- We are currently in a propose/break/patch cycle.
  - Despite the provision of security models and proofs.
- Just identifying and proving leakage is not enough; we need to also identify real-world implications of that leakage.
- Does PPE provide added security or a false sense of security?