Differential Privacy

Based on slides from: Ken Birman

A firm foundation for private data analysis

Cynthia Dwork

“A firm foundation”

1. Motivation: privacy with statistical databases.
2. Past attempts at privacy.
3. Creating privacy using noise.
4. Defining differential privacy.
5. Achieving differential privacy.
6. Properties of differentially private queries.

"A firm foundation"

1. **Motivation: privacy with statistical databases.**
2. Past attempts at privacy.
3. Creating privacy using noise.
4. Defining differential privacy.
5. Achieving differential privacy.
6. Properties of differentially private queries.

1. Motivation

- "Anonymous" datasets (with PII removed) have been connected to specific individuals.
  - AOL search histories.
  - Netflix prize.
  - Human genetic datasets.
- All of these cases involved *auxiliary information*.
- A technology is required to give incentives (or at least to remove disincentives) to contribute to these datasets.
  - "First, do no harm."

Statistical databases

- The purpose of a statistical database is to inform.
- By definition, these databases will be exposed to users.
- A user can, even with good intentions, become an attacker.

The goal is to reveal information, so the right definition of privacy is not obvious.

Dalenius's Desideratum

An attempt at a workable definition of privacy in this setting:

"Anything that can be learned about a respondent from the statistical database should be learnable without access to the database."

Naturally relates to **semantic security** in cryptosystems.

Dalenius's Desideratum

- Terry Gross is two inches shorter than the average Lithuanian woman `DB allows computing average height of a Lithuanian woman `This DB breaks Terry Gross's privacy according to this definition… even if her record is not in the database!
- This has been extended to a general proof of the inadequacy of this definition.
- This definition fails as it punishes the database for revealing information, which is the purpose of a statistical database.

"A firm foundation"

1. Motivation: privacy with statistical databases.
2. **Past attempts at privacy.**
3. Creating privacy using noise.
4. Defining differential privacy.
5. Achieving differential privacy.
6. Properties of differentially private queries.

2. Past attempts

- ***Large query sets*:** Forbid queries about specific individuals.
  - Non-specific queries can still reveal information.
- ***Query auditing*:** Determine by analysis if a set of queries will reveal information about individuals.
  - Computationally infeasible in general [J. Kleinberg et al.].
  - Rejecting a query leaks information.
- ***Subset sampling*:** Release only a subset of the dataset.
  - Punishes individuals in the subsample

2. Past attempts

- ***Input perturbation*:** Randomize the data at collection time. "Randomize once."
  - Does not work with complex data.
- ***Output perturbation*:** Add random noise to query responses.
  - If done naively, easy to defeat.

"A firm foundation"

1. Motivation: privacy with statistical databases.
2. Past attempts at privacy.
3. **Creating privacy using noise.**
4. Defining differential privacy.
5. Achieving differential privacy.
6. Properties of differentially private queries.

Two problems

- We do not have a good definition of privacy in this setting.
  - How can we make progress without one?
- Past attempts to achieve some kind of privacy did not work.

Blatant non-privacy

A system is blatantly non-private if an adversary can construct a replica database that matches the real database in 99% of its entries.

The adversary gets at most 1% wrong.

Of past approaches, adding random noise did not work if done naively, but perhaps we can fix it.

How much noise must be added by our database mechanism to avoid blatant non-privacy?

Theorem 1: Let M be a mechanism that adds noise bounded by E. Then there exists an adversary that can reconstruct the database to within $4E$ positions.

PROOF: Let d be the true database. The adversary can attack in two phases:

1. **Estimate the number of 1's in all possible sets:** Query M on all subsets $S \subseteq [n]$.

2. **Rule out "distant" database:** For every candidate database $c \in \{0,1\}^n$, if, for any $S \subseteq [n]$, $|\sum_{i \in S} c_i - M(S)| > E$, then rule out c. If c is not ruled out, then output c and halt.

$M(S)$ never errs by more than E, so the real database will not be ruled out and thus this algorithm must return some database. Call its output c.

Let I_0 be the indices in which $d_i = 0$. Given the second step of the algorithm, it must be that $|M(I_0) - \sum_{i \in I_0} c_i| \leq E$. By assumption $|M(I_0) - \sum_{i \in I_0} d_i| \leq E$. It follows from the triangle inequality that c and d differ in at most $2E$ positions in I_0.

Let I_1 be the indices in which $d_i = 1$. The same argument holds, so that c and d differ in at most $2E$ positions in I_1.

Thus, c and d agree on all but at most $4E$ positions.

To avoid blatant non-privacy, we must add noise bounded by $n/400$. A bound of $n/401$ or lower is provably non-private.

Noise and the "noisy table"

- Our analysis demonstrates that we cannot release a static "noisy table" that can be used to get very accurate statistics about the real data.
  - A table that can give accurate statistics can be attacked.
  - A table that cannot be attacked cannot give accurate statistics.
- This yields a key conclusion: we can only achieve both privacy and accurate statistics with an *interactive* database mechanism.

Privacy goal

- We considered already Dalenius's definition of privacy and saw that it is inadequate.
- Because statistical databases are intended to reveal information and because we want to ensure participation in these databases, we require a special definition of privacy.

Our goal is to minimize the increased risk to an individual incurred by joining or leaving the database.

This goal leads us directly to *differential privacy*.

"A firm foundation"

1. Motivation: privacy with statistical databases.
2. Past attempts at privacy.
3. Creating privacy using noise.
4. **Defining differential privacy.**
5. Achieving differential privacy.
6. Properties of differentially private queries.

Differential privacy

- It should not harm you or help you as an individual to enter or to leave the dataset.
- To ensure this property, we need a mechanism whose output is nearly unchanged by the presence or absence of a single respondent in the database.
- In constructing a formal approach, we concentrate on pairs of databases (*D, D'*) differing on only one row, with one a subset of the other and the larger database containing a single additional row.

Differential privacy

Definition 2. A randomized function K gives ϵ-differential privacy if for all data sets D and D' differing on at most one row, and all $S \subseteq Range(K)$,

$$\Pr[K(D) \in S] \leq \exp(\epsilon) \times \Pr[K(D') \in S],$$

where the probability space in each case is over the coin flips of K.

Differential privacy

- An equivalent expression of this idea is given as a ratio bounded by R:

$$\frac{Pr[K(D) \in S]}{Pr[K(D') \in S]} \leq exp(\epsilon) = R$$

- The closer R is to 1, or ε to 0, the more difficult it will be for an attacker to determine an individual's data.

- ε is a publicly known characteristic of our database. It defines the level of privacy maintained and it informs users of the amount of error to expect in the responses it yields.

Differential privacy

- An important property of this definition is that any output with zero probability is invalid for all databases.
  - An output with a probability of zero in a given database must have a probability of zero in both neighboring databases and by induction, in any other database as well.
- It immediately follows that sub-sampling fails to implement differential privacy.
  - A row cannot be present in a sub-sample if that person has previously left the dataset.
- With this definition in hand, we can attempt an implementation.

"A firm foundation"

1. Motivation: privacy with statistical databases.
2. Past attempts at privacy.
3. Creating privacy using noise.
4. Defining differential privacy.
5. **Achieving differential privacy.**
6. Properties of differentially private queries.

Noise properties

- We know we can add noise to query responses to disguise the true contents of the database.
- We know the level of disguise required for differential privacy.
- What distribution should we employ to generate this noise?

Simple case

"How many rows in the database satisfy P?"

Adding or removing a row can only change the answer by 1.

To build a differentially private mechanism for answering this query, we add to the response random noise drawn from a distribution with the property:

$$\forall z, z' s.t. |z - z'| = 1 : Pr[z] \leq e^{\epsilon} Pr[z']$$

We make this requirement so that the noise itself does not leak information beyond our chosen ε.

$$e^{-\epsilon} \leq \frac{Pr[noise = r - m]}{Pr[noise = r - m + 1]} \leq e^{\epsilon}$$

General case

- We must be able to handle vector-valued queries.
- To do this, we must consider the *sensitivity* of the function that will generate the response.
  - In the simple case, the sensitivity was 1.

DEFINITION 3. For $f : D \rightarrow \mathbf{R}^d$, the L_1 sensitivity of f is

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_1$$
$$= \max_{D,D'} \sum_{i=1}^{d} |f(D)_i - f(D')_i|$$

for all D, D' differing in at most one row.

- The sensitivity defines the difference that the noise must hide.

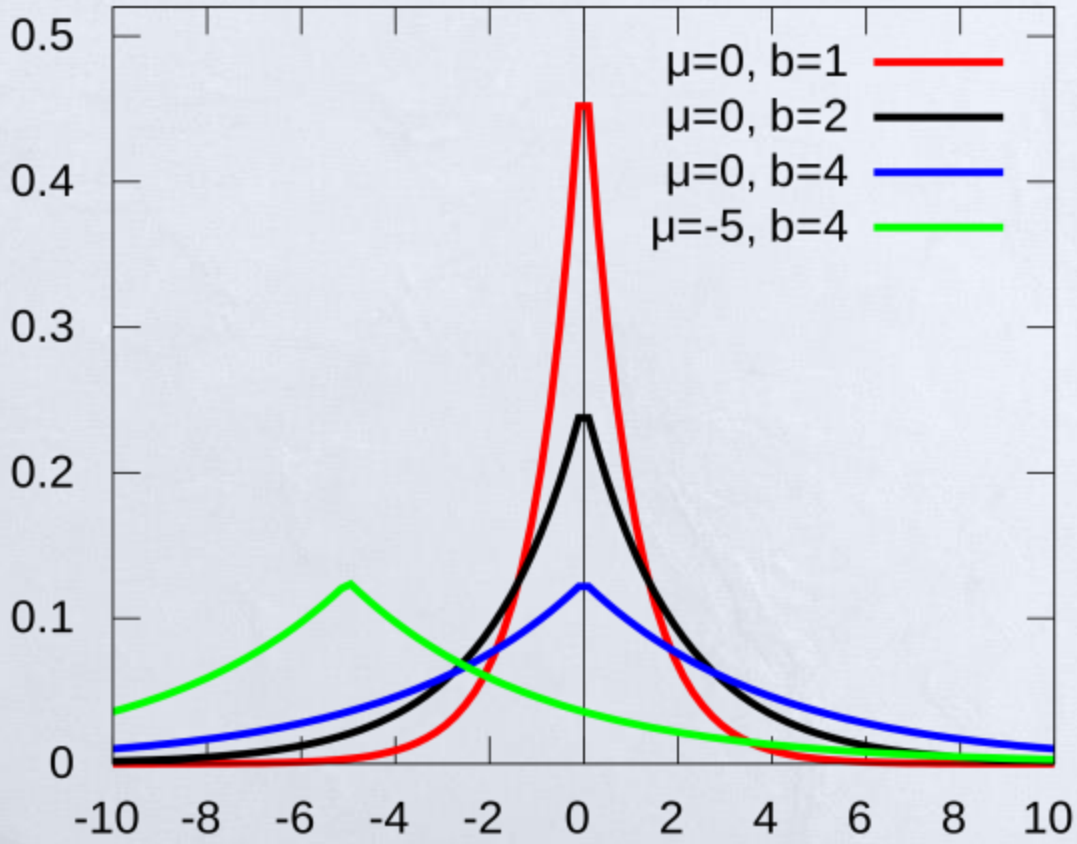
Laplace distribution

- We generate noise using the Laplace distribution.
- The Laplace distribution, denoted Lap(b), is defined with parameter b and has density function:

$$P(z|b) = \frac{1}{2b} e^{\frac{-|z|}{b}}$$

- Taking $b = 1/\varepsilon$ we have immediately that the density is proportional to $e^{-\varepsilon|z|}$.
- This distribution has its highest density at 0.
- **For any z, z' such that $|z - z'| \leq 1$, the density at z is at most e^{ε} times the density at z', satisfying the condition we outlined in the simple case.**
- The distribution is symmetric about 0.
- The distribution flattens as ε decreases. More likely to deviate from the true value.

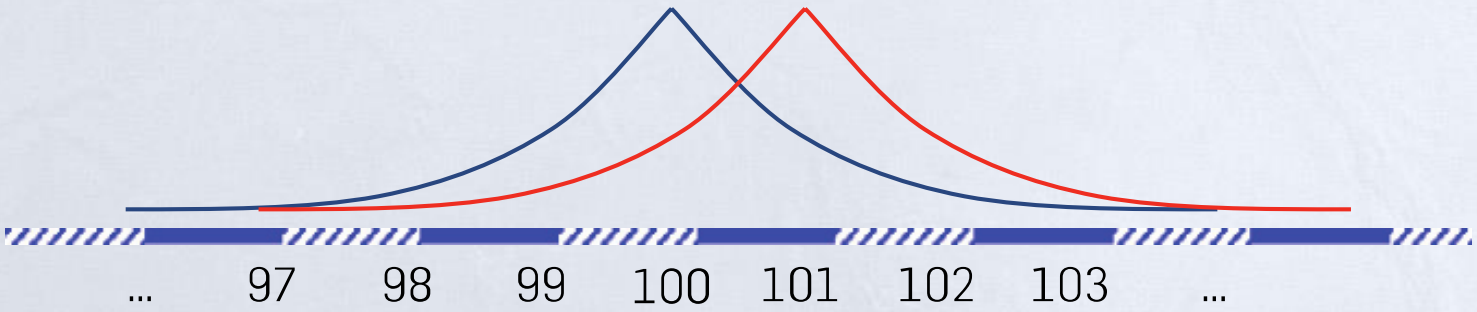
Laplace distribution



Wikipedia, Creative Commons license

Final theorem

Theorem 4. For $f : D \rightarrow \mathbf{R}^d$, the mechanism K that adds independently generated noise with distribution $Lap(\Delta f/\epsilon)$ to each of the d output terms enjoys ϵ-differential privacy.



In this figure, the distribution on the outputs, show in gray, is centered at the true answer of 100, where $\Delta f = 1$ and $\varepsilon = \ln 2$. In orange is the same distribution where the true answer is 101.

PROOF. Consider any subset $S \subseteq Range(K)$, and let D, D' be any pair of databases differing in at most one row. When the database is D, the probability density at any $r \in S$ is proportional to $e^{-||f(D)-r||_1(\epsilon/\Delta f)}$. Similarly, when the database is D', the probability density at any $r \in Range(K)$ is proportional to $e^{-||f(D')-r||_1(\epsilon/\Delta f)}$.

$$\begin{aligned}
\frac{e^{-||f(D)-r||(\epsilon/\Delta f)}}{e^{-||f(D')-r||(\epsilon/\Delta f)}} &= \frac{e^{||f(D')-r||(\epsilon/\Delta f)}}{e^{||f(D)-r||(\epsilon/\Delta f)}} \\
&= \frac{e^{||f(D')-r||(\epsilon/\Delta f)}}{e^{||f(D)-r||(\epsilon/\Delta f)}} \\
&= e^{(||f(D')-r||-||f(D)-r||)(\epsilon/\Delta f)} \\
&\leq e^{(||f(D')-f(D)||)(\epsilon/\Delta f)}
\end{aligned}$$

where the inequality follows from the triangle inequality. By the definition of sensitivity, $||f(D') - f(D)||_1 \leq \Delta f$, and so the ratio is bounded by e^{ϵ}. Integrating over S yields ϵ-differential privacy.

"A firm foundation"

1. Motivation: privacy with statistical databases.
2. Past attempts at privacy.
3. Creating privacy using noise.
4. Defining differential privacy.
5. Achieving differential privacy.
6. **Properties of differentially private queries.**

Sequence of queries

Given any query sequence $f_1, ..., f_m$, ϵ-differential privacy can be achieved by running K with noise distribution $\text{Lap}(\Sigma_{i=1}^{m} \Delta f_i / \epsilon)$ on each query.

- We allow the quality of each answer to deteriorate with the sum of sensitivities of the queries, maintaining ε-differential privacy.
- **A complex query need only be penalized by its aggregate sensitivity. This may be surprisingly small.**
  - **Example: the number of 2-bit rows whose entries are both 1 has sensitivity 1 despite involving 2 bits per row.**

Histogram queries

- Histogram queries are an example of the aforementioned principle.
- The addition or removal of a row can only change the count in a bucket by 1.
- This means we need only perturb the count in each bucket according to $\mathrm{Lap}(\Delta f / \varepsilon) = \mathrm{Lap}(1 / \varepsilon)$.
- **The cost in noise of a query has the desired property: it increases when the query threatens individual privacy and shrinks when the query concerns an aggregate value.**

Histogram queries

"This algorithm is a primitive that can be applied to any 'histogram' query – that is, one asking how many people fall into each of several mutually exclusive categories, such as first names"

"When [Adam] Smith told Dwork about this insight in the early days of differential privacy research, 'something inside me went, "Wow!"' Dwork said. **'I realized that we could exploit the structure of a query or computation to get much greater accuracy than I had realized.'**"

- Scientific American

Sensitivity analysis

- To employ this technology for practical data analysis, we rely on the observation that standard data mining queries can be implemented as noisy sums, allowing them to be broken into a series of steps with known sensitivity, giving an aggregate sensitivity for the overall query.
- Even queries that are challenging to describe by sensitivity can be performed with ε-differential privacy maintained.
- Much of the work done in differential privacy is in the creation of algorithms with low noise cost.

Objections

1. Do enough queries and we lose privacy.
   Does this make sense? We just stop using the data?
2. This work presumes a static dataset.
   What happens to the analysis if the data is changing?
3. Are there static parameters of the database that should be hidden as well?
4. Using differential privacy does not stop many other forms of information leakage.
5. Whoever does the calculation has the data.
6. How limited is the query model?
   The standard database model does not fit with differential privacy.

Query model

- If it is natural to put things in terms of a histogram, then it is natural to query.
- Absurd example from the *Scientific American* article:

"If you wanted to generate a list of the top 100 baby names for 2012, for example, you could ask a series of questions of the form, "How many babies were given names that start with A?" (or Aa, Ab or Ac), and work your way through the possibilities."

Query model

"'One of the early results in machine learning is that almost everything that is possible in principle to learn can be learned through counting queries,' [Aaron] Roth said. 'Counting queries are not isolated toy problems, but a fundamental primitive' — that is, a building block from which many more complex algorithms can be built."

- This does not seem like a query model most people can use naturally.
- Perhaps Dwork defined away this problem by focusing on *statistical databases*.

Mr. Burns

- We can learn about expected power consumption over a population.
- But we cannot learn about an employee's name.

Suggested Reading

- (a few practical options)

- CryptDB: Protecting Confidentiality with Encrypted Query Processing. Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Proceedings of the 23rd ACM SOSP, Cascais, Portugal, October 2011.
- Fully Homomorphic Encryption. Wikipedia page maintained by Craig Genty and others.
- Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In PKC 2010, LNCS volume 6056, pages 420-443. Springer, 2010.

CryptDB?

CryptDB

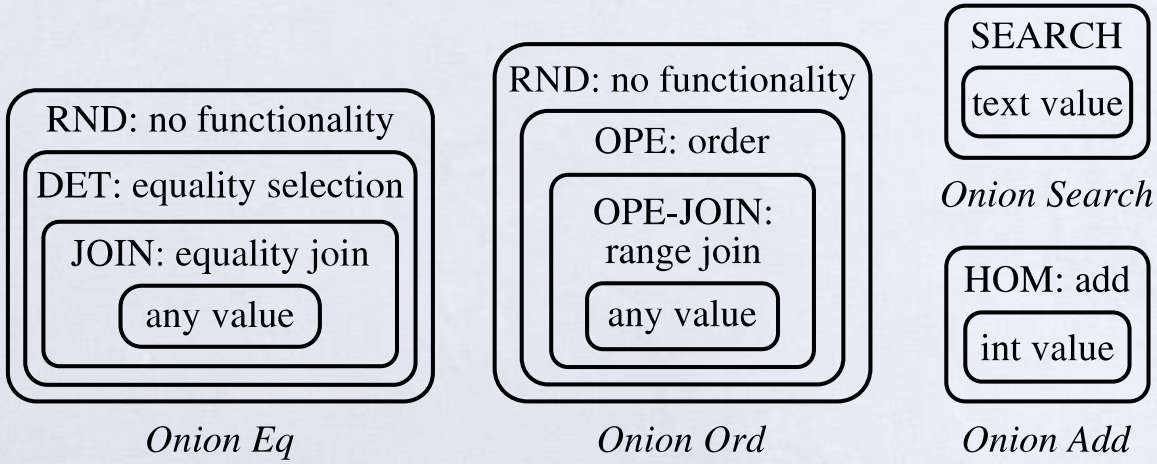


Figure 2: Onion encryption layers and the classes of computation they allow. Onion names stand for the operations they allow at some of their layers (Equality, Order, Search, and Addition). In practice, some onions or onion layers may be omitted, depending on column types or schema annotations provided by application developers (§3.5.2). DET and JOIN are often merged into a single onion layer, since JOIN is a concatenation of DET and JOIN-ADJ (§3.4). A random IV for RND (§3.1), shared by the RND layers in *Eq* and *Ord*, is also stored for each data item.

CryptDB

Application	Total cols.	Consider for enc.	Needs plaintext	Needs HOM	Needs SEARCH	Non-plaintext cols. with MinEnc: RND	SEARCH	DET	OPE	Most sensitive cols. at HIGH
phpBB	563	23	0	1	0	21	0	1	1	6 / 6
HotCRP	204	22	0	2	1	18	1	1	2	18 / 18
grad-apply	706	103	0	0	2	95	0	6	2	94 / 94
OpenEMR	1,297	566	7	0	3	526	2	12	19	525 / 540
MIT 6.02	15	13	0	0	0	7	0	4	2	1 / 1
PHP-calendar	25	12	2	0	2	3	2	4	1	3 / 4
TPC-C	92	92	0	8	0	65	0	19	8	—
Trace from `sql.mit.edu`	128,840	128,840	1,094	1,019	1,125	80,053	350	34,212	13,131	—
... with in-proxy processing	128,840	128,840	571	1,016	1,135	84,008	398	35,350	8,513	—
... col. name contains *pass*	2,029	2,029	2	0	0	1,936	0	91	0	—
... col. name contains *content*	2,521	2,521	0	0	52	2,215	52	251	3	—
... col. name contains *priv*	173	173	0	4	0	159	0	12	2	—

Figure 9: Steady-state onion levels for database columns required by a range of applications and traces. "Needs plaintext" indicates that CryptDB cannot execute the application's queries over encrypted data for that column. For the applications in the top group of rows, sensitive columns were determined manually, and only these columns were considered for encryption. For the bottom group of rows, all database columns were automatically considered for encryption. The rightmost column considers the application's most sensitive database columns, and reports the number of them that have MinEnc in HIGH (both terms are defined in §8.3).

CryptDB performance

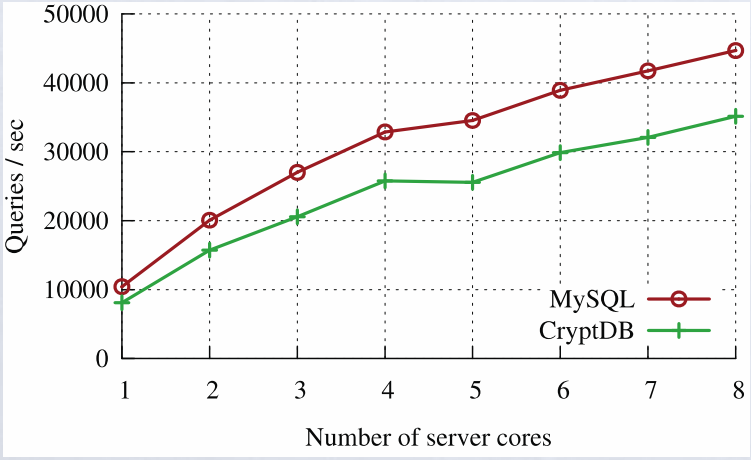


Figure 10: Throughput for TPC-C queries, for a varying number of cores on the underlying MySQL DBMS server.

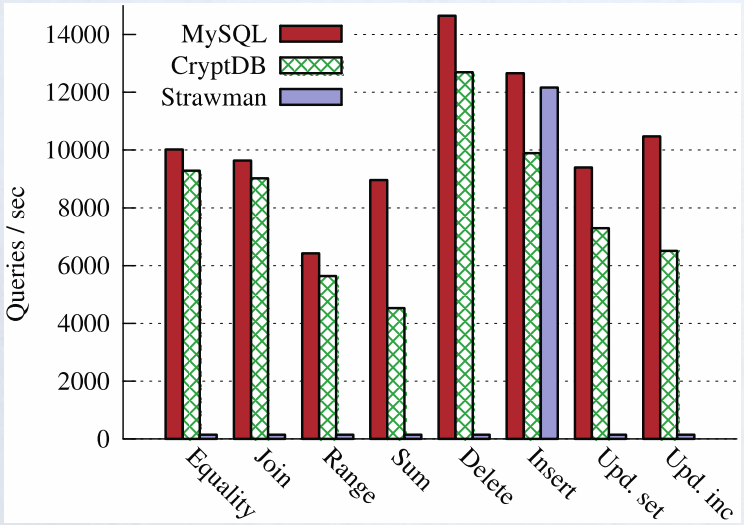


Figure 11: Throughput of different types of SQL queries from the TPC-C query mix running under MySQL, CryptDB, and the strawman design. "Upd. inc" stands for `UPDATE` that increments a column, and "Upd. set" stands for `UPDATE` which sets columns to a constant.

Homomorphic encryption

- Computation done on encrypted values is reflected in the unencrypted values.
- Known since the 1970s.
- Key breakthrough: an encryption system that is homomorphic under both addition and multiplication and is secure.
  - Due to Craig Gentry, announced in 2009.
    - 2014 MacArthur Fellow, has a J.D., and worked as a lawyer!
  - This allows for homomorphic computation of any Boolean circuit.

Homomorphic encryption

"**Unfortunately -- you knew that was coming, right? -- Gentry's scheme is completely impractical.** It uses something called an ideal lattice as the basis for the encryption scheme, and both the size of the ciphertext and the complexity of the encryption and decryption operations grow enormously with the number of operations you need to perform on the ciphertext -- and that number needs to be fixed in advance. And converting a computer program, even a simple one, into a Boolean circuit requires an enormous number of operations. **These aren't impracticalities that can be solved with some clever optimization techniques and a few turns of Moore's Law; this is an inherent limitation in the algorithm.**"

- Bruce Schneier