

Security and Data Privacy

Instructor: Pratiksha Thaker

cs245.stanford.edu

Outline

Security requirements

Key concepts and tools

Differential privacy

Other security tools

Outline

Security requirements

Key concepts and tools

Differential privacy

Other security tools

Why Security & Privacy?

Why Security & Privacy?

Data is valuable & can cause harm if released

- » Example: medical records, purchase history, internal company documents, etc

Data releases can't usually be “undone”

Security policies can be complex

- » Each user can only see data from their friends
- » Analyst can only query aggregate data
- » Users can ask to delete their derived data

Why Security & Privacy?

It's the law! New regulations about user data:

US HIPAA: Health Insurance Portability & Accountability Act (1996)

- » Mandatory encryption, access control, training

EU GDPR, CA CCPA: (2018)

- » Users can ask to see & delete their data

PCI DSS: Payment Card Industry standard (2004)

- » Required in contracts with MasterCard, etc

Consequence

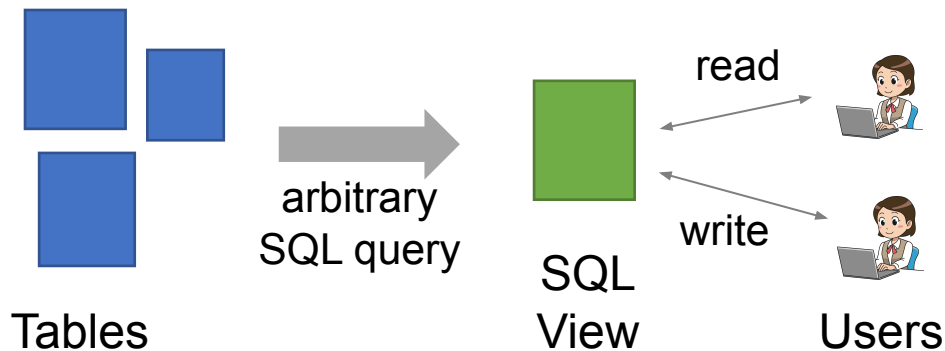
Security and privacy must be baked into the design of data-intensive systems

» Often a key differentiator for products!

The Good News

Declarative interface to many data-intensive systems can enable powerful security features
» One of the “big ideas” in our class!

Example: System R's access control on views



Outline

Security requirements

Key concepts and tools

Differential privacy

Other security tools

Some Security Goals

Access Control: only the “right” users can perform various operations; typically relies on:

- » **Authentication:** a way to verify user identity (e.g. password)
- » **Authorization:** a way to specify what users may take what actions (e.g. file permissions)

Auditing: system records an incorruptible audit trail of who did each action

Some Security Goals

Confidentiality: data is inaccessible to external parties (often via cryptography)

Integrity: data can't be modified by external parties

Privacy: only a limited amount of information about “individual” users can be learned

Clarifying These Goals

Say our goal was **access control**: only Matei can set CS 245 student grades on Axxess

What scenarios should Axxess protect against?

Clarifying These Goals

Say our goal was **access control**: only Matei can set CS 245 student grades on Axxess

What scenarios should Axxess protect against?

1. Bobby T. (an evil student) logging into Axxess as himself and being able to change grades

Clarifying These Goals

Say our goal was **access control**: only Matei can set CS 245 student grades on Axxess

What scenarios should Axxess protect against?

1. Bobby T. (an evil student) logging into Axxess as himself and being able to change grades
2. Bobby sending hand-crafted network packets to Axxess to change his grades

Clarifying These Goals

Say our goal was **access control**: only Matei can set CS 245 student grades on Axxess

What scenarios should Axxess protect against?

1. Bobby T. (an evil student) logging into Axxess as himself and being able to change grades
2. Bobby sending hand-crafted network packets to Axxess to change his grades
3. Bobby getting a job as a DB admin at Axxess

Clarifying These Goals

Say our goal was **access control**: only Matei can set CS 245 student grades on Axxess

What scenarios should Axxess protect against?

1. Bobby T. (an evil student) logging into Axxess as himself and being able to change grades
2. Bobby sending hand-crafted network packets to Axxess to change his grades
3. Bobby getting a job as a DB admin at Axxess
4. Bobby guessing Matei's password

Clarifying These Goals

Say our goal was **access control**: only Matei can set CS 245 student grades on Axxess

What scenarios should Axxess protect against?

1. Bobby T. (an evil student) logging into Axxess as himself and being able to change grades
2. Bobby sending hand-crafted network packets to Axxess to change his grades
3. Bobby getting a job as a DB admin at Axxess
4. Bobby guessing Matei's password
5. Bobby blackmailing Matei to change his grade

Clarifying These Goals

Say our goal was **access control**: only Matei can set CS 245 student grades on Axxess

What scenarios should Axxess protect against?

1. Bobby T. (an evil student) logging into Axxess as himself and being able to change grades
2. Bobby sending hand-crafted network packets to Axxess to change his grades
3. Bobby getting a job as a DB admin at Axxess
4. Bobby guessing Matei's password
5. Bobby blackmailing Matei to change his grade
6. Bobby discovering a flaw in AES to do #2

Threat Models

To meaningfully reason about security, need a **threat model**: what adversaries may do

- » Same idea as failure models!

For example, in our Axiom scenario, assume:

- » Adversaries only interact with Axiom through its public API
- » No crypto algorithm or software bugs
- » No password theft

Implementing complex security policies can be hard even with these assumptions!

Threat Models

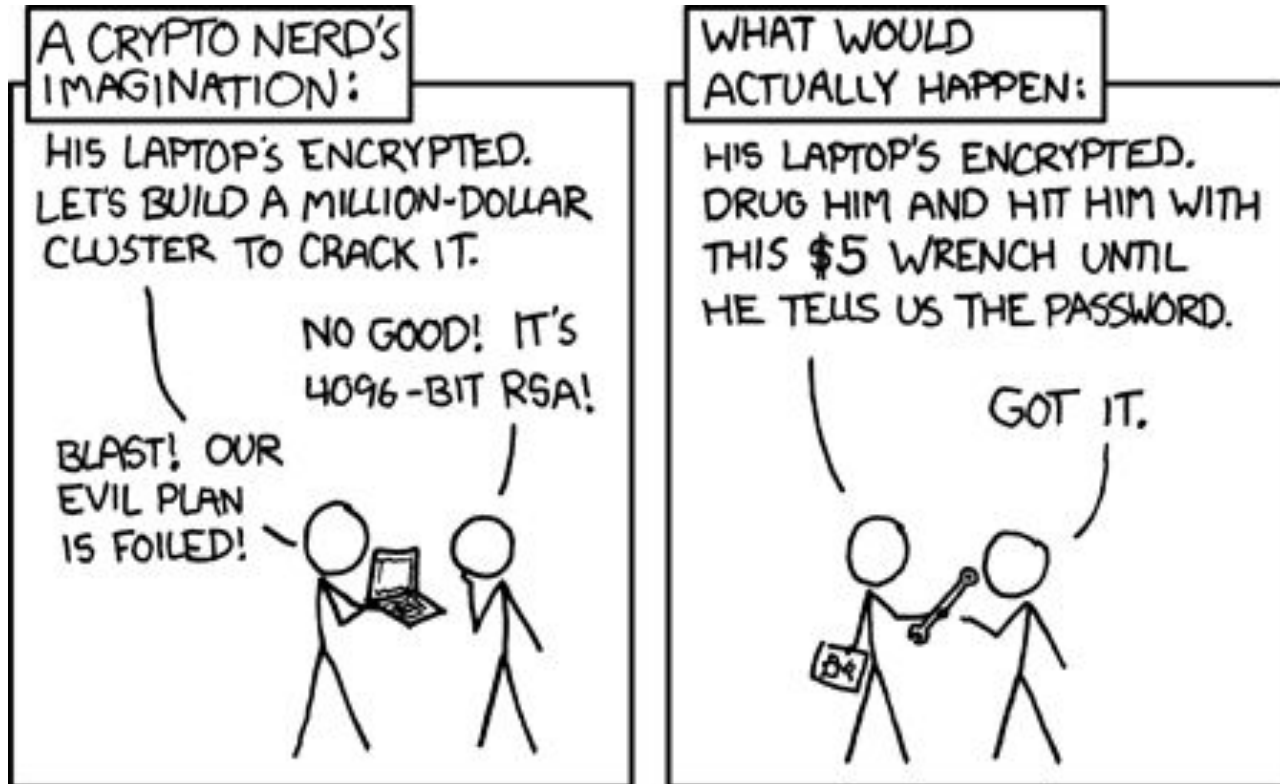
No useful threat model can cover everything

- » Goal is to cover the most feasible scenarios for adversaries to increase the **cost** of attacks

Threat models also let us divide security tasks across different components

- » E.g. auth system handles passwords, 2FA

Threat Models



Useful Building Blocks

Encryption: encode data so that only parties with a key can efficiently decrypt

Cryptographic hash functions: hard to find items with a given hash (or collisions)

Secure channels (e.g. TLS): confidential, authenticated communication for 2 parties

Security Tools from DBMSs

First-class concept of users + *access control*
» Views as in System R, tables, etc

Secure channels for network *communication*

Audit logs for analysis

Encrypt data on-disk (perhaps at OS level)

Modern Tools for Security

Privacy metrics and enforcement thereof
(e.g. differential privacy)

Computing on encrypted data (e.g. CryptDB)

Hardware-assisted security (e.g. enclaves)

Multi-party computation (e.g. secret sharing)

Outline

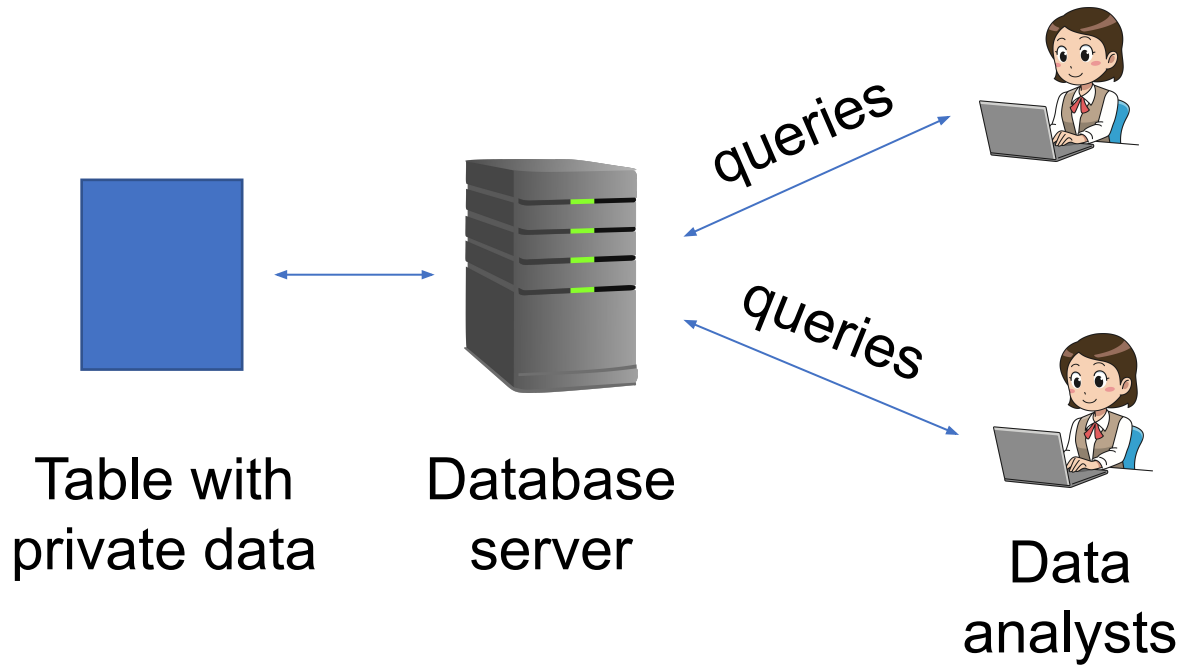
Security requirements

Key concepts and tools

Differential privacy

Other security tools

Threat Model



- Database software is working correctly
- Adversaries only access it through public API
- Adversaries have limited # of user accounts

Private statistics

```
SELECT AVG(income) FROM professors WHERE  
state="California"
```

Private statistics

Are aggregate statistics *more private* than individual data?

```
SELECT AVG(income) FROM professors WHERE  
state="California"
```

```
SELECT AVG(income) FROM professors WHERE  
name="Matei Zaharia"
```

Private statistics

Are aggregate statistics *more private* than individual data? **No!**

```
SELECT AVG(income) FROM professors WHERE  
state="California"
```

```
SELECT AVG(income) FROM professors WHERE  
name="Matei Zaharia"
```

Private statistics

Robust De-anonymization of Large Datasets (How to Break Anonymity of the Netflix Prize Dataset)

Arvind Narayanan and Vitaly Shmatikov

The University of Texas at Austin

February 5, 2008

Abstract

We present a new class of statistical de-anonymization attacks against high-dimensional micro-data, such as individual preferences, recommendations, transaction records and so on. Our techniques are robust to perturbation in the data and tolerate some mistakes in the adversary's background knowledge.

We apply our de-anonymization methodology to the Netflix Prize dataset, which contains anonymous movie ratings of 500,000 subscribers of Netflix, the world's largest online movie rental service. We demonstrate that an adversary who knows only a little bit about an individual subscriber can easily identify this subscriber's record in the dataset. Using the Internet Movie Database as the source of background knowledge, we successfully identified the Netflix records of known users, uncovering their apparent political preferences and other potentially sensitive information.

1 Introduction

Private statistics

Robust De-anonymization of Large Datasets

(How to)

OPEN ACCESS Freely available online

PLoS GENETICS

Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays

Nils Homer^{1,2}, Szabolcs Szelinger¹, Margot Redman¹, David Duggan¹, Waibhav Tembe¹, Jill Muehling¹, John V. Pearson¹, Dietrich A. Stephan¹, Stanley F. Nelson², David W. Craig^{1*}

¹ Translational Genomics Research Institute (TGen), Phoenix, Arizona, United States of America, ² University of California Los Angeles, Los Angeles, California, United States of America

We present a
such as individu
robust to perturb

We apply our
movie ratings of
demonstrate that
identify this sub
background know
apparent politica

1 Introduction

Abstract

We use high-density single nucleotide polymorphism (SNP) genotyping microarrays to demonstrate the ability to accurately and robustly determine whether individuals are in a complex genomic DNA mixture. We first develop a theoretical framework for detecting an individual's presence within a mixture, then show, through simulations, the limits associated with our method, and finally demonstrate experimentally the identification of the presence of genomic DNA of specific individuals within a series of highly complex genomic mixtures, including mixtures where an individual contributes less than 0.1% of the total genomic DNA. These findings shift the perceived utility of SNPs for identifying individual trace contributors within a forensics mixture, and suggest future research efforts into assessing the viability of previously sub-optimal DNA sources due to sample contamination. These findings also suggest that composite statistics across cohorts, such as allele frequency or genotype counts, do not mask identity within genome-wide association studies. The implications of these findings are discussed.

Citation: Homer N, Szelinger S, Redman M, Duggan D, Tembe W, et al. (2008) Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays. PLoS Genet 4(8): e1000167. doi:10.1371/journal.pgen.1000167

Editor: Peter M. Visscher, Queensland Institute of Medical Research, Australia

Received: December 7, 2007; **Accepted:** July 15, 2008; **Published:** August 20, 2008

Idea: differential privacy

A contract for algorithms that output statistics

Idea: differential privacy

A contract for algorithms that output statistics

Intuition: the function is differentially private if removing or changing a data point does not change the output "too much"

Idea: differential privacy

A contract for algorithms that output statistics

Intuition: the function is differentially private if removing or changing a data point does not change the output "too much"

Intuition: plausible deniability

Idea: differential privacy

A contract for algorithms that output statistics

For A and B that differ in **one element**,

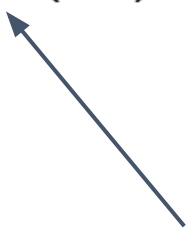
$$\Pr[M(A) \in S] \leq e^\epsilon \Pr[M(B) \in S]$$

Idea: differential privacy

A contract for algorithms that output statistics

For A and B that differ in **one element**,

$$\Pr[M(A) \in S] \leq e^\epsilon \Pr[M(B) \in S]$$



Randomized algorithm
that computes statistic

Idea: differential privacy

A contract for algorithms that output statistics

For A and B that differ in **one element**,

$$\Pr[M(A) \in S] \leq e^\epsilon \Pr[M(B) \in S]$$




Any subset of possible
outcomes

Idea: differential privacy

A contract for algorithms that output statistics

For A and B that differ in **one element**,

$$\Pr[M(A) \in S] \leq e^\epsilon \Pr[M(B) \in S]$$



Privacy parameter
Smaller $\epsilon \sim$ more
privacy, less accuracy

What Does It Mean?

Say an adversary runs some query and observes a result X

Adversary had some set of results, S , that lets them infer something about Matei if $X \in S$

Then:

$$\begin{aligned}\Pr[X \in S | \text{Matei} \in \text{DB}] &\leq e^\epsilon \Pr[X \in S | \text{Matei} \notin \text{DB}] \\ \Pr[X \notin S | \text{Matei} \in \text{DB}] &\leq e^\epsilon \Pr[X \notin S | \text{Matei} \notin \text{DB}]\end{aligned}$$

Note: An arrow points from the e^ϵ term in the first equation to the text $\approx 1+\epsilon$.

Similar outcomes whether or not Matei in DB

What Does It Mean?

Private information is noisy. Can we determine anything useful?

What Does It Mean?

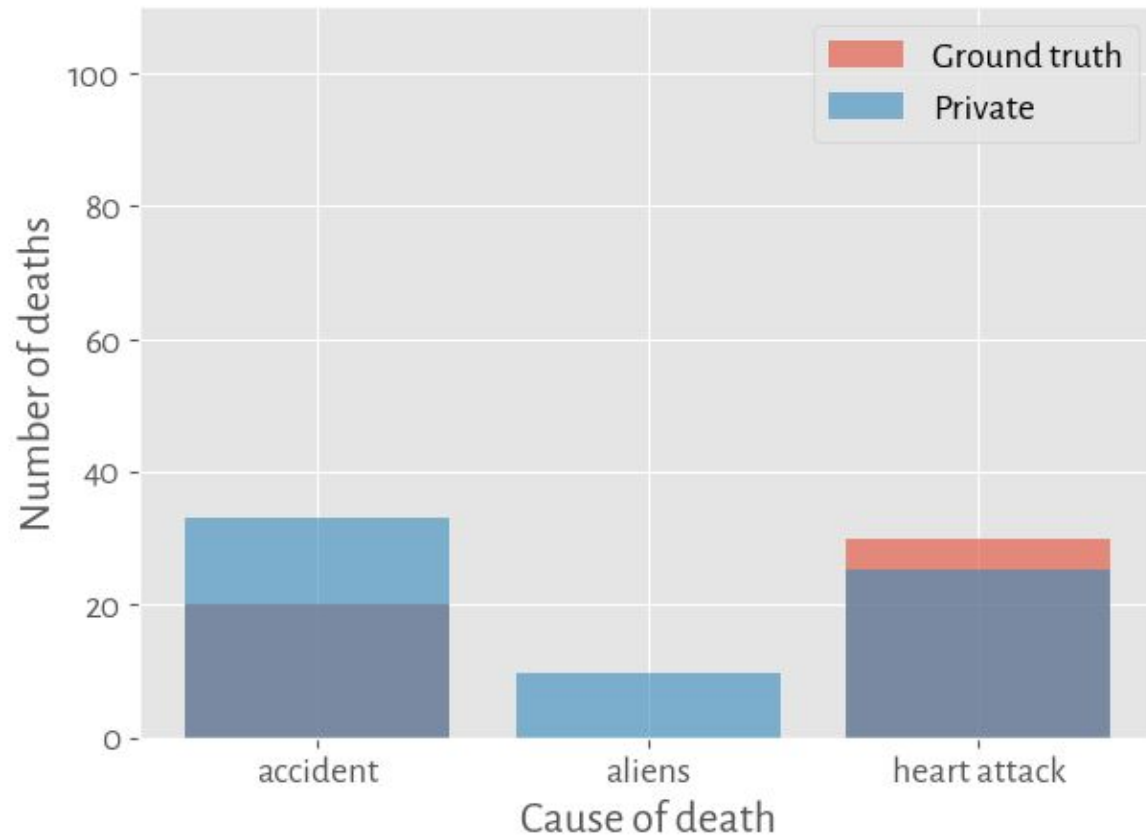
Private information is noisy. Can we determine anything useful?

Query:

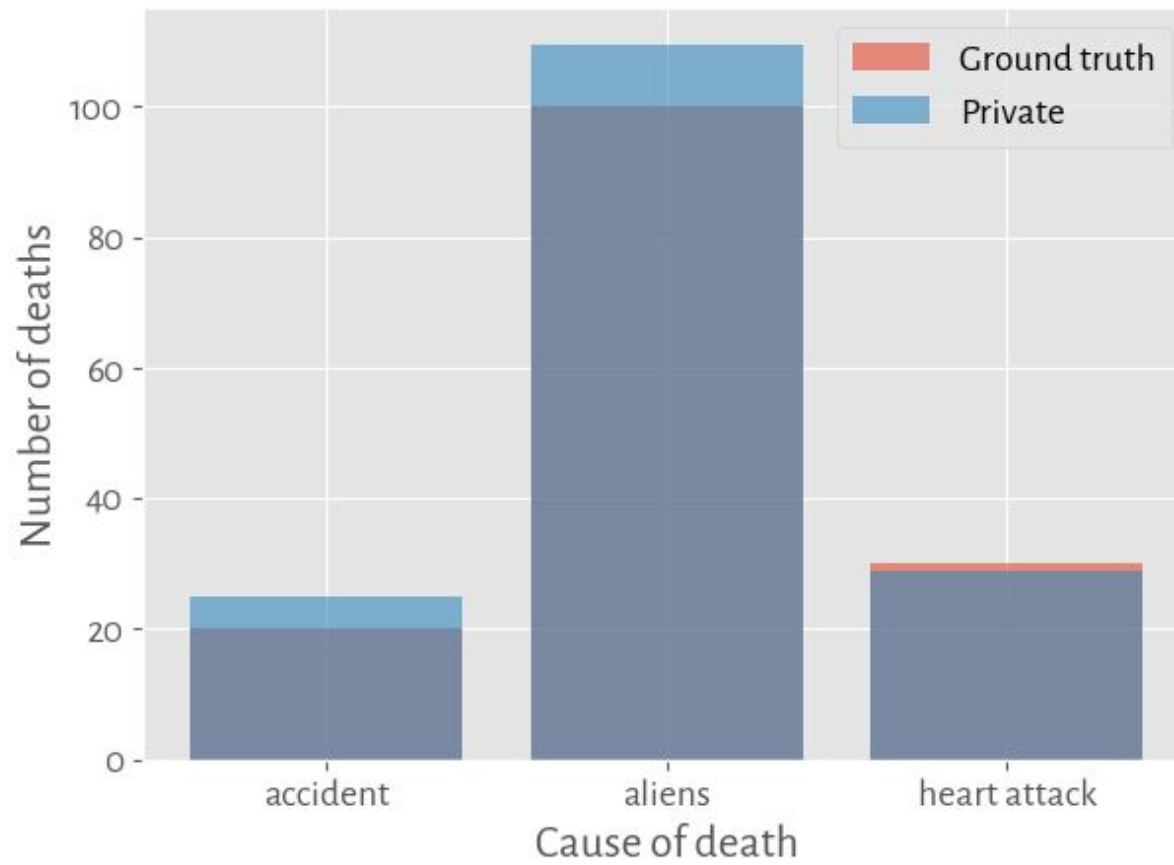
```
SELECT COUNT(*) FROM patients WHERE  
causeOfDeath = ...
```

Assume $\epsilon=0.1$

What Does It Mean?

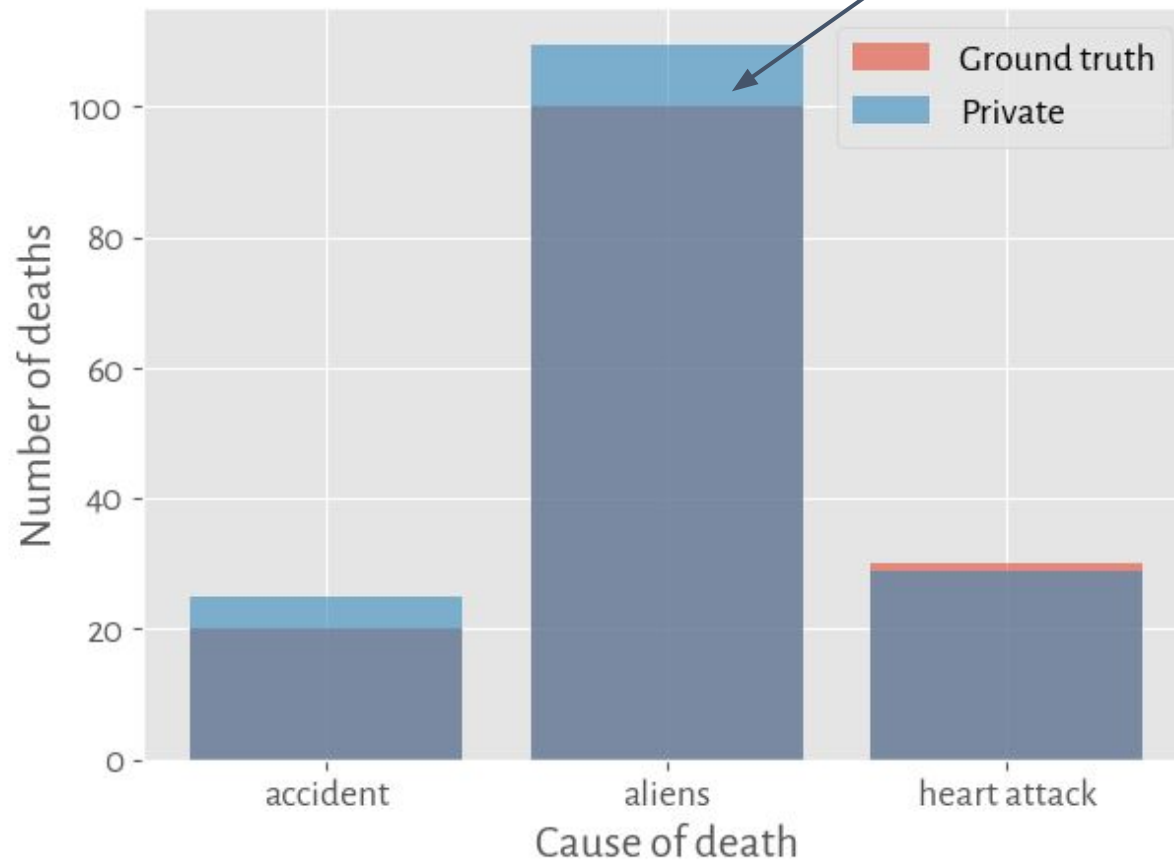


What Does It Mean?



What Does It Mean?

With enough base signal,
DP can still give useful
information!



Side Information

Consider the following query:

```
SELECT AVG(income) FROM professors WHERE  
state="CA"
```

Side Information

Consider the following query:

```
SELECT AVG(income) FROM professors WHERE  
state="CA"
```

What if adversary knows:

"Matei makes \$100k more than the average professor in CA"

Side Information

Consider the following query:

```
SELECT AVG(income) FROM professors WHERE  
state="CA"
```

What if adversary knows:

"Matei makes \$100k more than the average professor in CA"

Equally *bad* for Matei whether or not he personally participates in DB!

Some Nice Properties of Differential Privacy

Composition: can reason about the privacy effect of multiple (even dependent) queries

Let queries M_i each provide ϵ_i -differential privacy; then the set of queries $\{M_i\}$ provides $\sum_i \epsilon_i$ -differential privacy

Proof:

$$\Pr[\forall i M_i(A) = r_i] \leq e^{(\epsilon_1 + \dots + \epsilon_n) |A \oplus B|} \Pr[\forall i M_i(B) = r_i]$$

Adversary's ability to distinguish DBs A & B grows in a bounded way with each query

Some Nice Properties of Differential Privacy

Parallel composition: even better bounds if queries are on *disjoint subsets* (e.g., histograms)

Let M_i each provide ϵ -differential privacy and read disjoint subsets of the data D_i ; then the set of queries $\{M_i\}$ provides ϵ -differential privacy

Example: query both average patient age in CA and average patient age in NY

Some Nice Properties of Differential Privacy

Easy to compute: can use known results for various operators, then compose for a query
» Enables systems to **automatically** compute privacy bounds given declarative queries!

Disadvantages of Differential Privacy

Disadvantages of Differential Privacy

Each user can only make a limited number of queries (more precisely, limited total ϵ)

- » Their ϵ grows with each query and can't shrink
- » Have to specify how much ϵ to use

How to set ϵ in practice?

- » Apple system: 2? 4? 16?
- » NBER paper: 8?
- » Hard to tell what various values mean, though there is a nice Bayesian interpretation

Disadvantages of Differential Privacy

Each user can only make a limited number of queries (more precisely, limited total ϵ)



Computing Differential Privacy Bounds

Let's start with COUNT aggregates:
`SELECT COUNT(*) FROM A`

The randomized algorithm $M(A)$ that returns $|A| + \text{Laplace}(1/\epsilon)$ is ϵ -differentially private

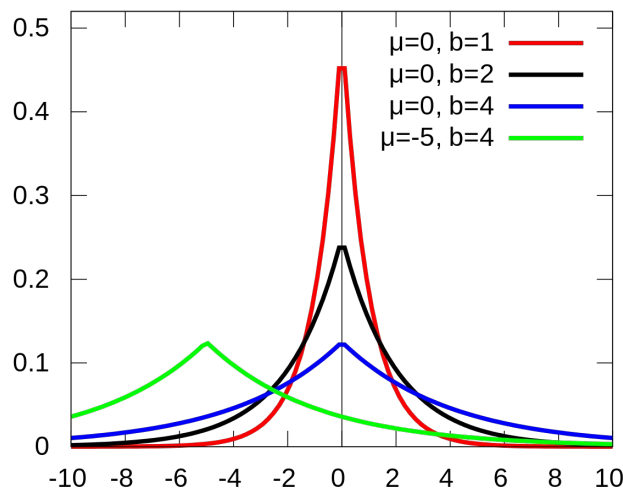


Image source: Wikipedia

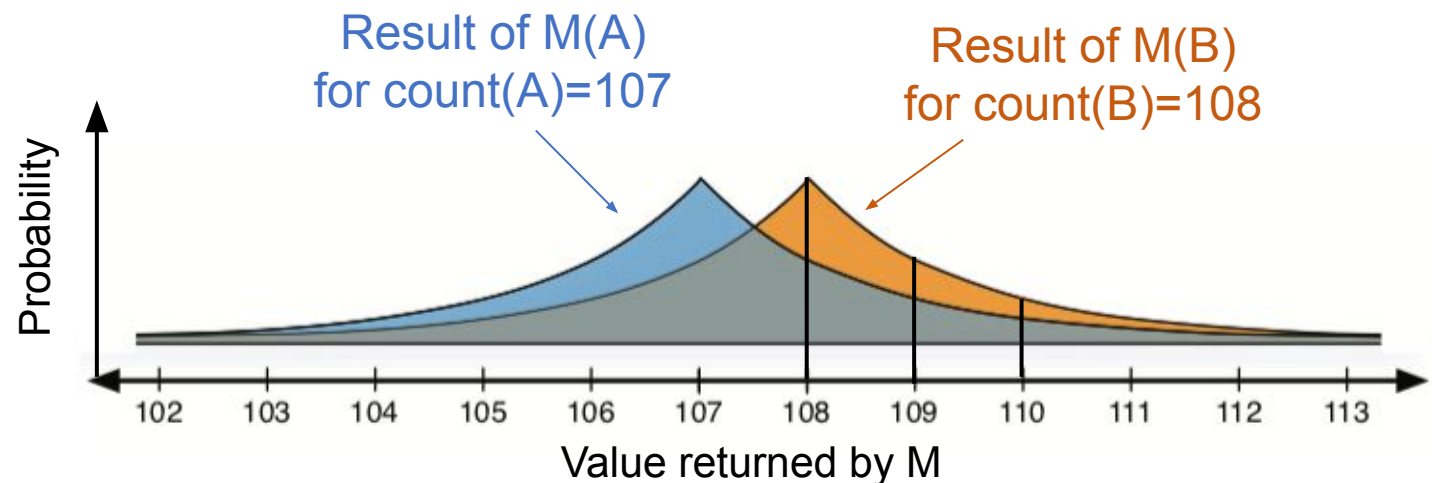
Laplace(b) distribution:
$$p(x) = 1/(2b) e^{-|x|/b}$$

Mean: 0
Variance: $2b^2$

Computing Differential Privacy Bounds

Let's start with COUNT aggregates:
`SELECT COUNT(*) FROM A`

The randomized algorithm $M(A)$ that returns $|A| + \text{Laplace}(1/\epsilon)$ is ϵ -differentially private



Computing Differential Privacy Bounds

What about AVERAGE aggregates:
`SELECT AVERAGE(x) FROM A`

Computing Differential Privacy Bounds

What about AVERAGE aggregates:
`SELECT AVERAGE(x) FROM A`

How much can one element of A affect result?

- » In general case, unboundedly much! No privacy
 - `SELECT AVG(wealth) WHERE city="Omaha, NE"`
- » If $x \in [0, m]$ for all x in A, then by at most m
 - Adding Laplace(m/ϵ) noise is ϵ -differentially private

Paper bounds AVG, SUM for values $x \in [-1, 1]$

Computing Differential Privacy Bounds

General notion to capture the impact of one element: **sensitivity**

Sensitivity of a function $f: U \rightarrow \mathbb{R}$ on sets is

$$\Delta f = \max_{A, B \in U \text{ differ in 1 element}} |f(A) - f(B)|$$

Sensitivity Examples

	Sensitivity
$f(A) = A $	1
$f(A) = \text{sum}(A), x \in [0, m] \quad \forall x \in A$	m
$f(A) = \text{avg}(A), x \in [0, m] \quad \forall x \in A$	m
$f(A) = \{x \in A \mid x \text{ is male}\} $	1
$f(A) = A \bowtie B $	unbounded
$f(A) = A \bowtie B $, each key has $\leq k$ matches	k

Multi-dimensional Sensitivity

Can also define sensitivity for functions that return multiple numerical results:

Sensitivity of a function $f: U \rightarrow \mathbb{R}^d$ on sets is

$$\Delta f = \max_{A, B \in U \text{ differ in 1 element}} \|f(A) - f(B)\|_1$$

Example: f fits a linear model to the data...

Computing Differential Privacy Bounds

How do we reason about functions involving **cardinality** (e.g. GROUP BY...)?

Computing Differential Privacy Bounds

How do we reason about functions involving **cardinality** (e.g. GROUP BY...)?

Concept introduced in PINQ: **stability**

Computing Differential Privacy Bounds

Concept introduced in PINQ: **stability**

A function T on sets is *c-stable* if for any two input sets A and B ,

$$|T(A) \oplus T(B)| \leq c |A \oplus B|$$

Number of records
that differ in A and B



PINQ's approach: let user do any # of set ops;
compute their stability; then let them do one
aggregate op and compute its sensitivity

Stability Examples

	Stability
$T(A) = \sigma_{\text{predicate}}(A)$ (“Where”)	1
$T(A) = \pi_{\text{exprs}}(A)$ (“Select”)	1
$T(A, B) = A \cup B$	1
$T(A) = \text{GroupBy}(A, \text{expr})$ (retruns 1 record/group)	2
$T(A) = A \bowtie B$ limited to at most 1 match per key	1

Partition Operator

`Partition(dataset, key_list)` returns a set of `IQueryables`: one for each key in your list

- » User provides the desired keys in advance (e.g. “CA” or “NY”); can’t use to discover keys
- » Lets PINQ use **parallel composition** rule since the sets returned are all disjoint

Stability = 1

Analyzing Queries in PINQ

User calls multiple set transformation ops and finally one aggregation/result op

- » Transformations are lazy; can't see result

PINQ computes stability of set ops and multiplies by sensitivity of each aggregate to get total sensitivity

User provides an ϵ to aggregate; PINQ adds noise proportional to sensitivity/ ϵ

Putting It All Together

Example 5 Measuring query frequencies in PINQ.

```
// prepare data with privacy budget
var agent = new PINQAgentBudget(1.0);
var data  = new PINQQueryable<string>(rawdata, agent);

// break out fields, filter by query, group by IP
var users = data.Select(line => line.Split(','))
                .Where(fields => fields[20] == args[0])
                .GroupBy(fields => fields[0]);

// output the count to the screen, or anywhere else
Console.WriteLine(args[0] + ": " + users.NoisyCount(0.1));
```

cricket: 127123.313

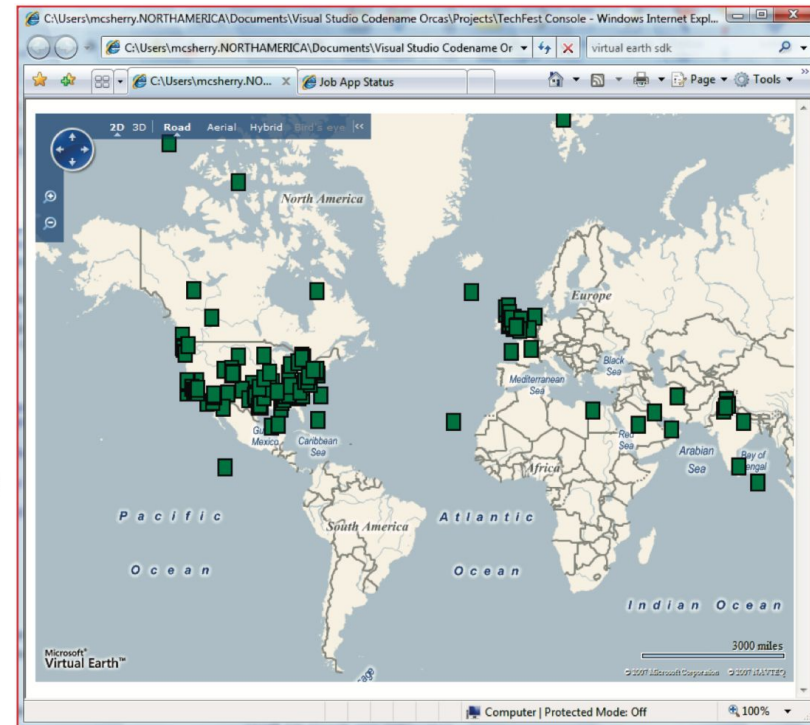
Putting It All Together

Example 7 Transforming IP addresses to coordinates.

```
// ... within the per-query loop, from before ...

// use the searches for query, group by IP address
var users = parts[query].GroupBy(fields => fields[0]);

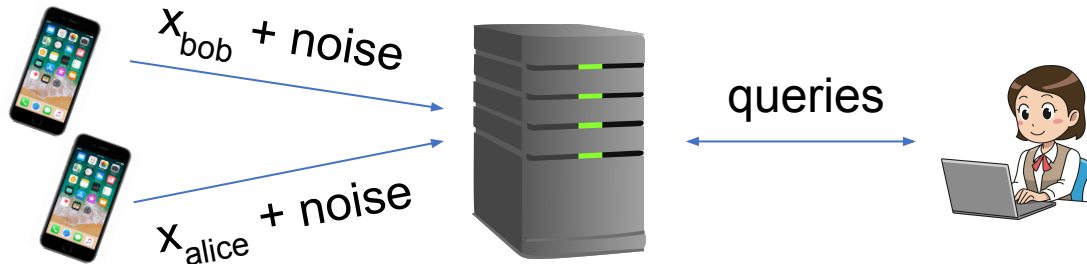
// extract IP address from each group, and match
var coords = users.Join(iplatlon,
    group => group.Key,
    entry => entry[0],
    (glist,elist) => elist.First());
```



Uses of Differential Privacy

Statistics collection about iOS features

“Randomized response”: clients add noise to data they send instead of relying on provider



Research systems that use DP to measure security (e.g. Vuvuzela messaging)

Outline

Security requirements

Key concepts and tools

Differential privacy

Other security tools

Computing on Encrypted Data

Threat model: adversary has access to the database server we run on (e.g. in cloud)

Idea: some encryption schemes allow computing on data without decrypting it:

$$f_{\text{enc}}(\text{Enc}(X)) = \text{Enc}(f(X))$$

Usually very expensive, but can be done efficiently for some functions f !

Example Systems

CryptDB, Mylar (MIT research projects)

Encrypted BigQuery (CryptDB on BigQuery)

Leverage properties of SQL to come up with efficient encryption schemes & query plans

Example Schemes

Equality checks with deterministic encryption

```
SELECT * FROM table WHERE state="CA"
```



Encrypt "state" column

```
SELECT * FROM table WHERE state="XAYDS9"
```

Example Schemes

Equality checks with deterministic encryption

```
SELECT * FROM table WHERE state="CA"
```



Encrypt "state" column

```
SELECT * FROM table WHERE state="XAYDS9"
```

Potential challenges with this scheme:

- » Adversary can see relative frequency of keys
- » Adversary sees which keys are accessed on each query (e.g. Matei logs in → CA key read)

Other Encryption Schemes

Additive homomorphic encryption:

$$\text{Enc}(A + B) = \text{Enc}(A) \odot \text{Enc}(B)$$

Fully homomorphic encryption:

$$\text{Enc}(f(A)) = f_{\text{enc}}(\text{Enc}(A))$$

Possible but very expensive
(10^8 or more overhead)

Order-preserving encryption:

$$\text{if } A < B \text{ then } \text{Enc}(A) < \text{Enc}(B)$$

Hardware Enclaves

Threat model: adversary has access to the database server we run on (e.g. in cloud) but can't tamper with hardware

Idea: CPU provides an “enclave” that can provably run some code isolated from the OS

- » Enclaves returns a certificate signed by CPU maker that it ran code C on argument A

Hardware Enclaves in Practice

Already present in all Intel CPUs (Intel SGX),
and many Apple custom chips (T2, etc)

Initial applications were digital rights mgmt.,
secure boot, secure login

» Protect even against a compromised OS

Some research systems explored using these
for data analytics: Opaque, ObliDB, others

Databases + Enclaves

1. Store data encrypted with an encryption scheme that leaks nothing (randomized)
2. With each query, user includes a public key k_q to encrypt the result with
3. Database runs a function f in the enclave that does query and encrypts result with k_q
4. User can verify f ran, DB can't see result!

Performance is fast too (normal CPU speed)!

Are Enclaves Enough to Secure Against Non-HW Adversaries?

Are Enclaves Enough to Secure Against Non-HW Adversaries?

Not quite! adversary can still learn info by observing **access patterns** to RAM or **timing**

- » Similar to some attacks on encrypted DBs

Oblivious algorithms can help prevent this but add more computational cost

- » Oblivious = same access pattern regardless of underlying data, query result, etc

Multi-Party Computation (MPC)

Threat model: participants p_1, \dots, p_n want to compute some joint function f of their data but don't trust each other

» E.g. patient stats across 2 hospitals

Idea: protocols that compute f without revealing anything else to participants

» Like with encryption, general computations are possible but expensive

Example: Secret Sharing

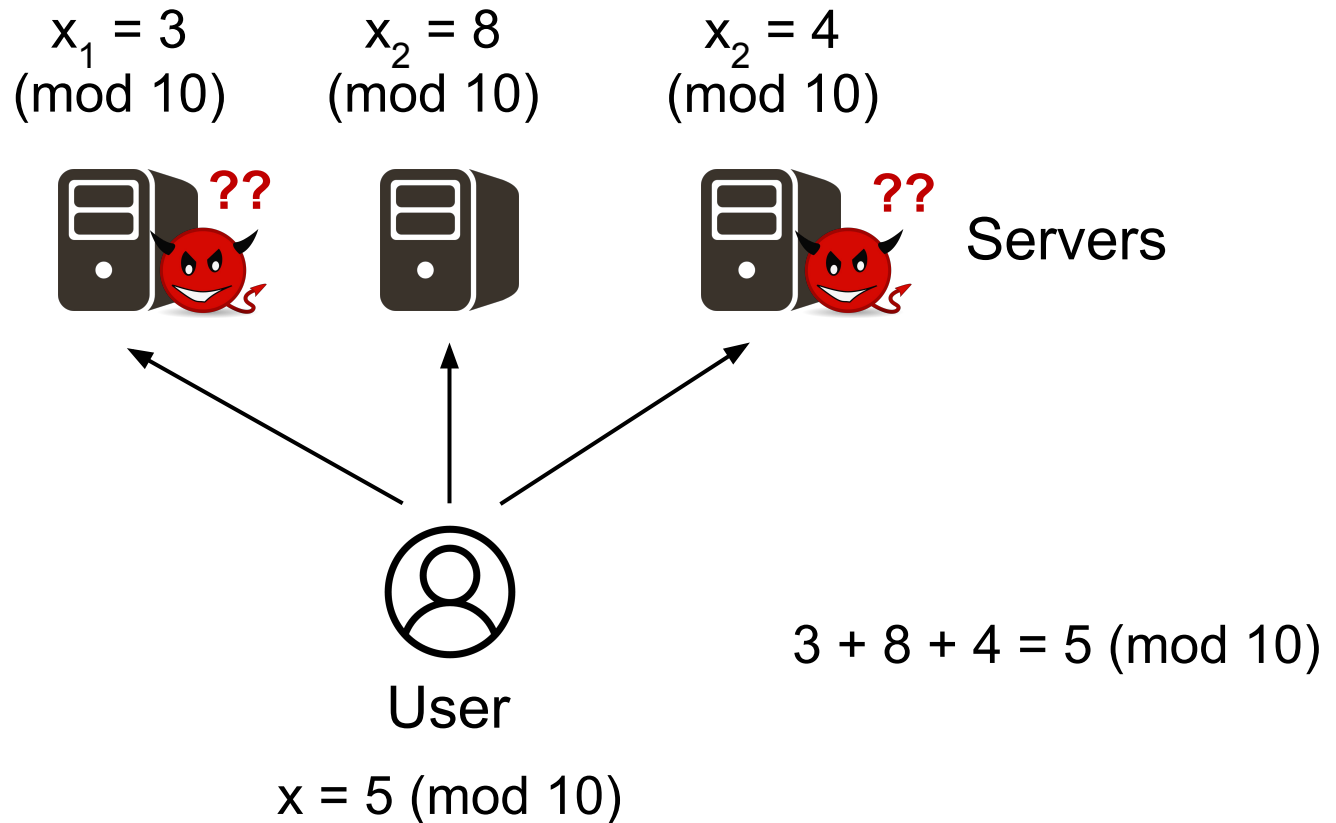
Users wants to store a secret value x among n servers, but doesn't fully trust them

» E.g. the servers are public clouds... what if one gets hacked?

Idea: split x into “shares” x_i so that all shares are needed to recover x

Additive secret sharing: $x = \text{integer mod } P$,
 x_i are random integers so $\sum x_i = x$

Secret Sharing Example

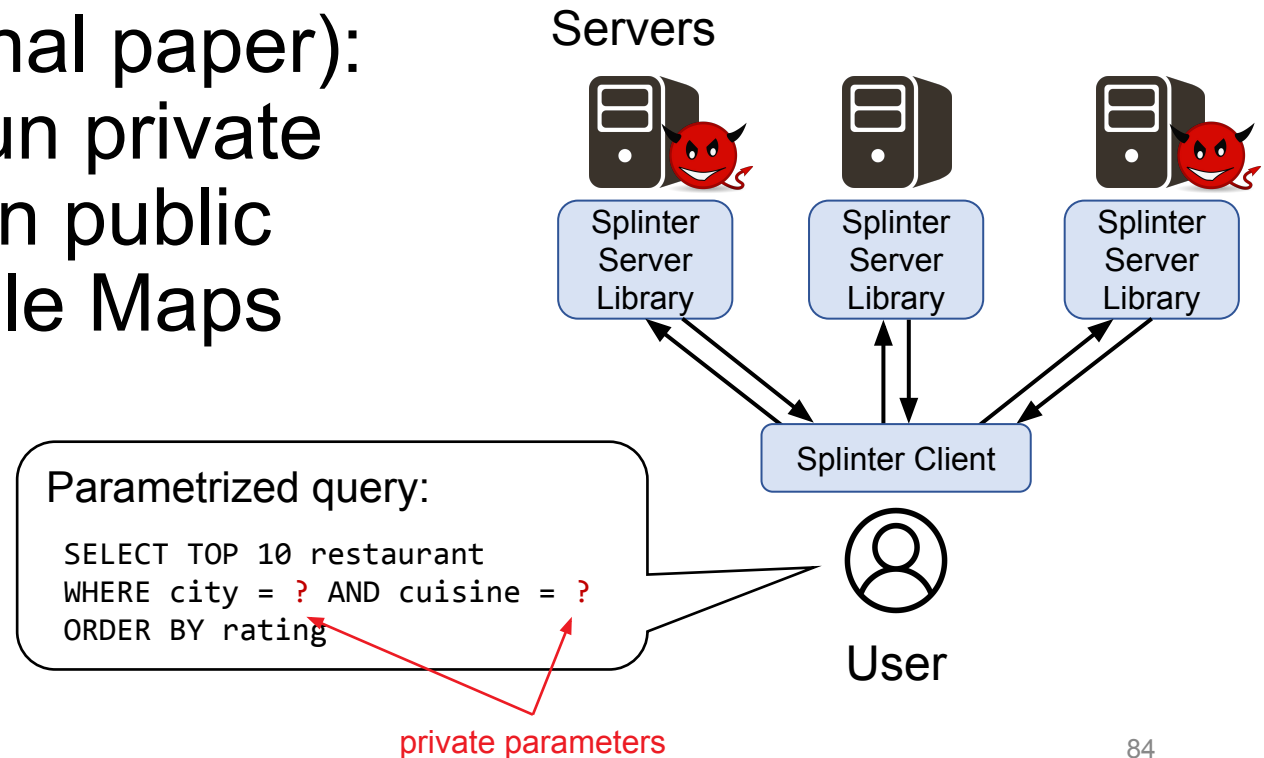


Note: performance is quite fast (just additions)

Function Secret Sharing

Recent result that allows sharing some functions too (keeping queries private)

Splinter (optional paper):
uses FSS to run private
SQL queries on public
data like Google Maps



Lineage Tracking and Retraction

Goal: keep track of which data records were derived from an individual input record

- » Facilitate removing a user's data in GDPR, verifying compliance, etc

Some real systems provide this already at low granularity, but could be baked into DB

Summary

Security and data privacy are essential concerns for data-intensive systems

Threat models are a systematic way to measure security and reason about designs

Many nice theoretical tools exist to reason about security needs of relational & math ops
» Build on declarative and relational APIs!