RUTGERS

Jana: Private Data as a Service

Anand D. Sarwate
Dept. of ECE / DIMACS
Rutgers, The State University of New Jersey

NSF DARPA NIH

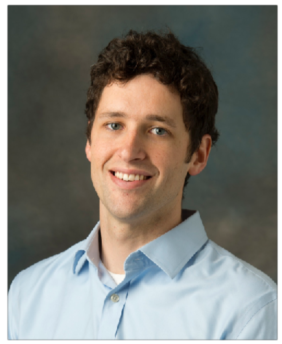
The Team

RUTGERS

Rebecca Wright

Anand Sarwate

THE UNIVERSITY OF CHICAGO



David Cash

GEORGE MASON UNIVERSITY

Dov Gordon

KATHOLIEKE UNIVERSITEIT LEUVEN

Nigel Smart

Portland State UNIVERSITY

Charles Wright

galois

Dave Archer
Principle Investigator

Data as a Service (PDaaS)

Data as a Service (DaaS) has proved very popular and useful.

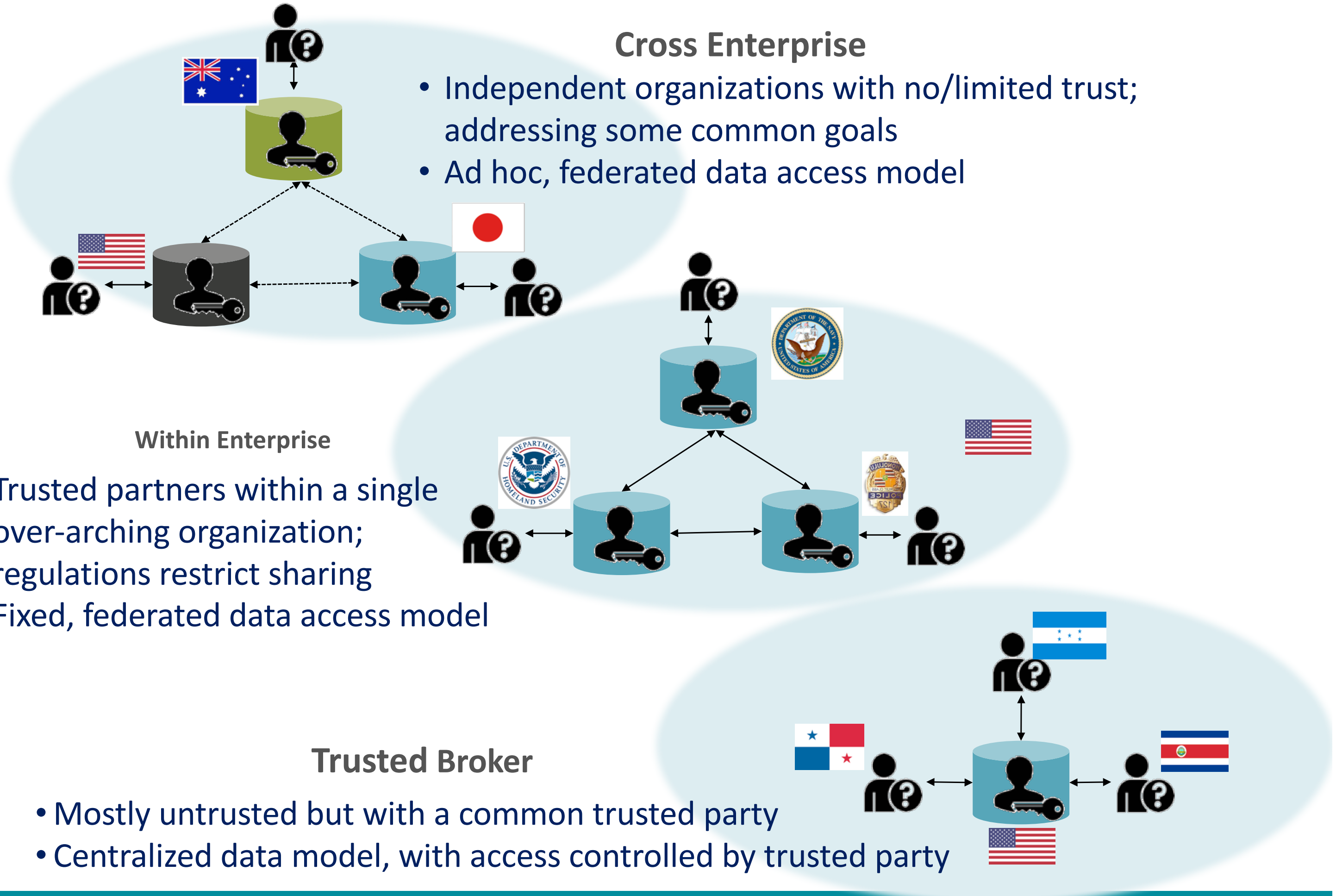
- Easy to use
- Standardized interfaces
- Fast
- Reliable: Atomicity, Consistency, Isolation, Durability (ACID)

What about using *private* data?

- Allows services to use data from multiple providers
- Creates challenges for modeling and guaranteeing privacy
- Either emulate centralized or decentralized/local model
  - See Salil's talk yesterday

Use Case 1: Sharing for coalitions

Enterprise Privacy Models



Cross Enterprise

- Independent organizations with no/limited trust; addressing some common goals
- Ad hoc, federated data access model

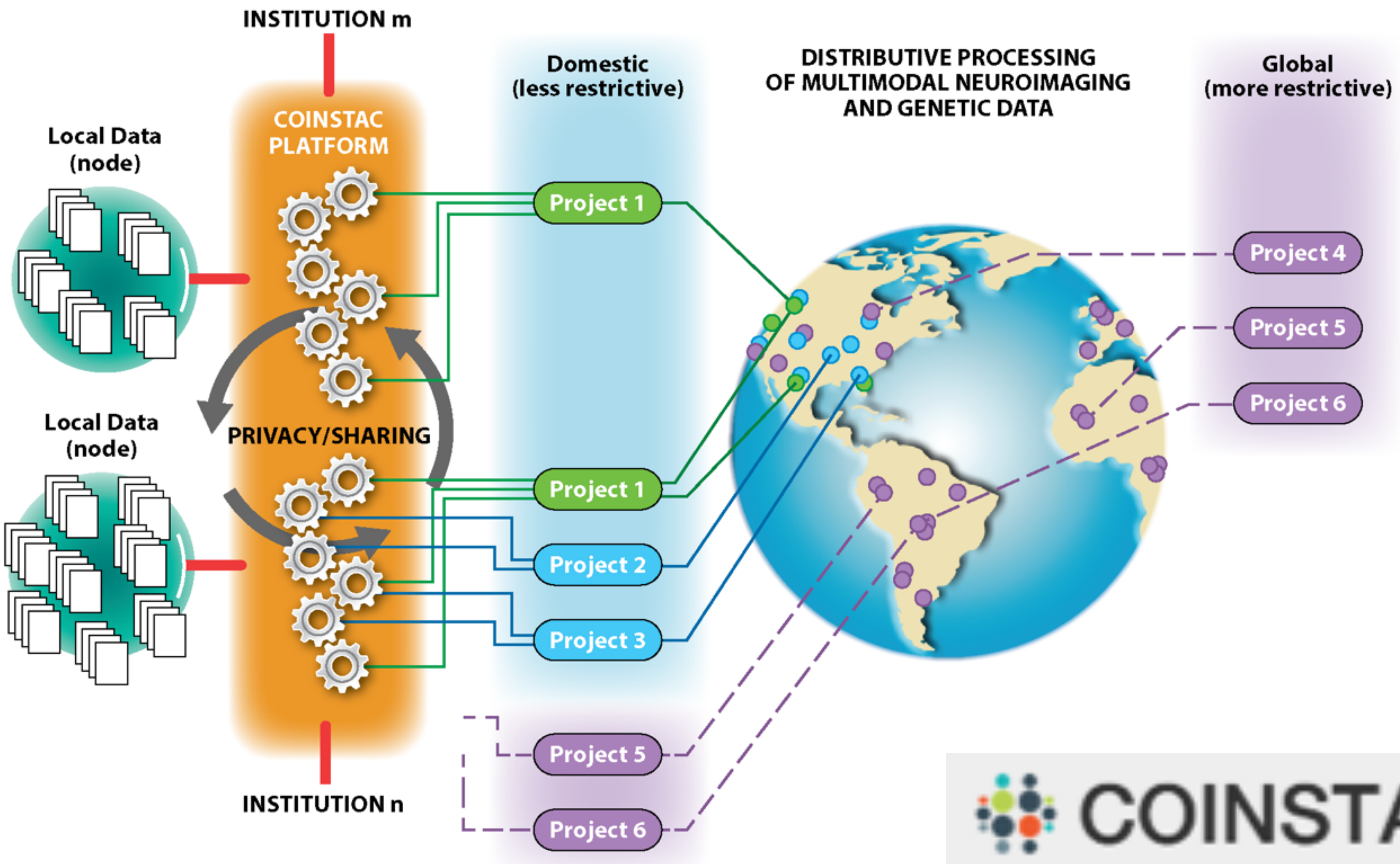
Within Enterprise

- Trusted partners within a single over-arching organization; regulations restrict sharing
- Fixed, federated data access model

Trusted Broker

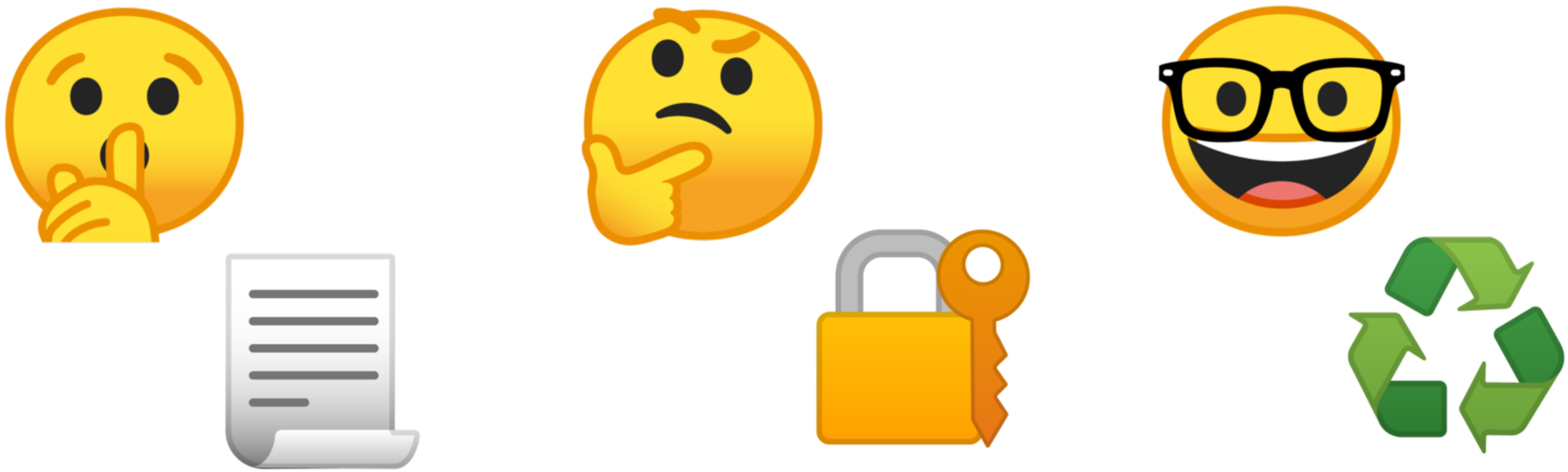
- Mostly untrusted but with a common trusted party
- Centralized data model, with access controlled by trusted party

Use Case 2: collaboration for health research



COINSTAC

Enabling Private DaaS (PDaaS)



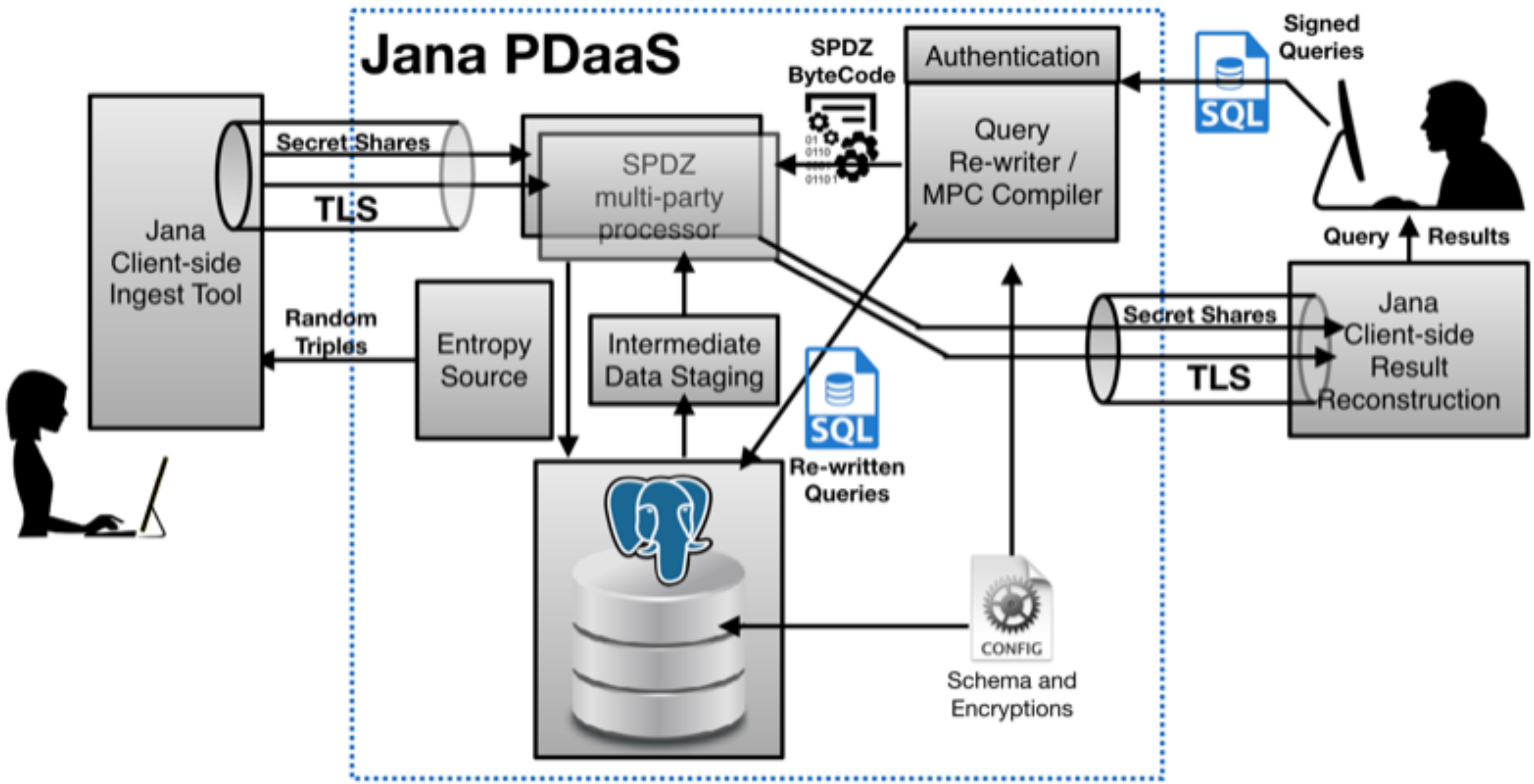
Goal: Data as a Service with a “privacy first” focus.

- **Data providers** can specify “privacy policies”
- **Data analysis** should use “privacy preserving” methods
- **Developers** should not have to reinvent the wheel

Key technologies to incorporate

- Secure data ingest
- Searchable encryption
- Secure multiparty computation
- Differential privacy
- Query processor to allow SQL-like query interface and enforce policies

Jana in a Picture (JiaP)



RUTGERS

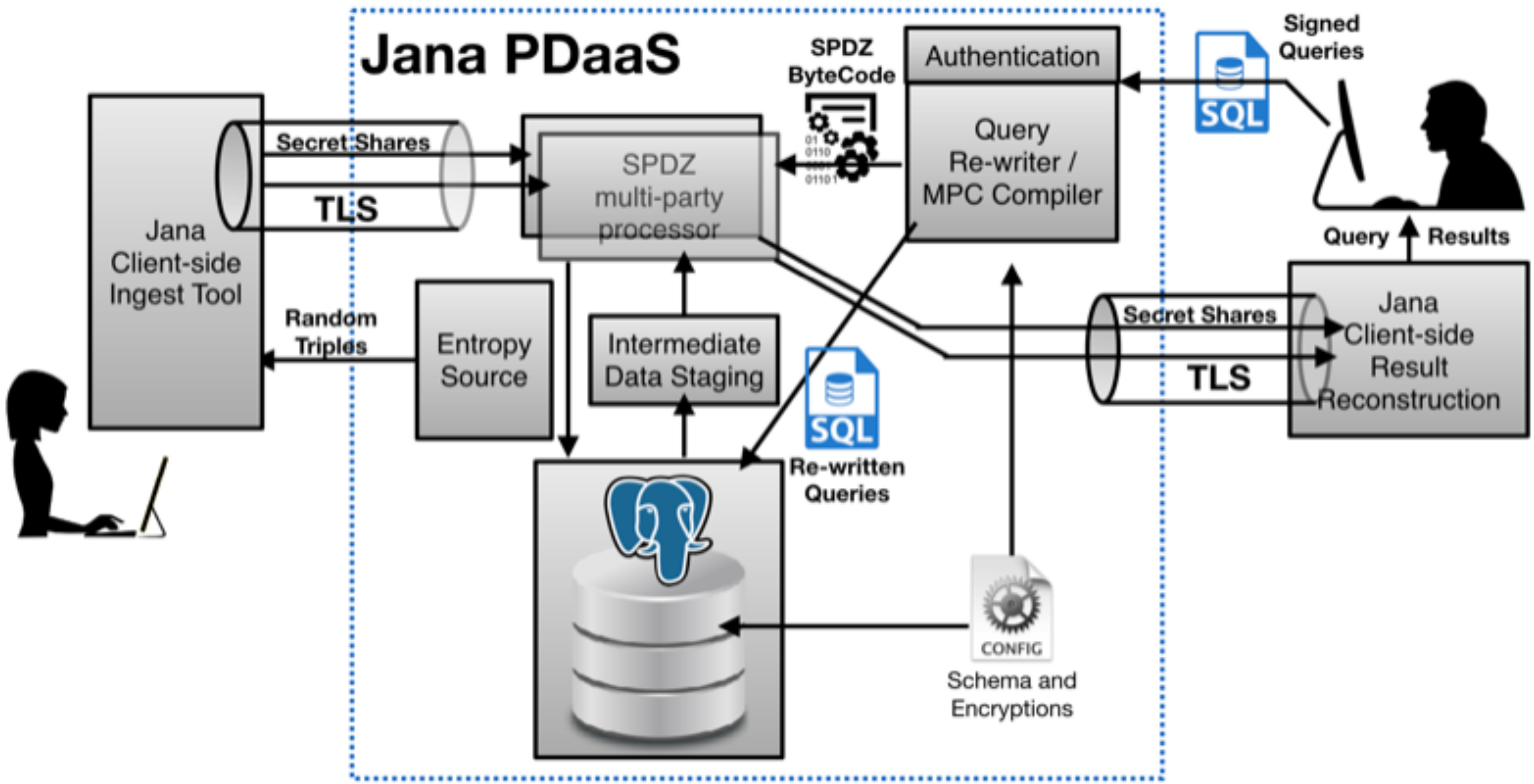
THE JANA SYSTEM

DIMACS

Jana capabilities

- Functionality
  - Generous subset of SQL
  - RDBMS ACID properties
- Privacy
  - Data-in-transit: public key cryptography
  - Data-at-rest: deterministic, random, searchable
  - Computation: MPC + RDBMS using deterministic & searchable encryption
  - Results: differential privacy applied (if needed) while in MPC
- Performance
  - 10Ks of records moving to 100Ks, queries in seconds to hours
- Deployment
  - Web service with RESTful API, Docker appliance

Jana in a Picture (JiaP)



Jana workflow

Data ingest:

- use public key encryptions of secret shares of data to protect the most sensitive provider data
- use searchable encryption schemes when data may be less sensitive

Query processing:

- analyst issues a query using standard SQL
- query re-writer breaks the queries into intermediate queries to the DB and a MPC program to operate on data shares
- apply privacy policies of data holders

MPC: from SPDZ to SCALE + MAMBA

SCALE = Secure Computation Algorithms from Leuven

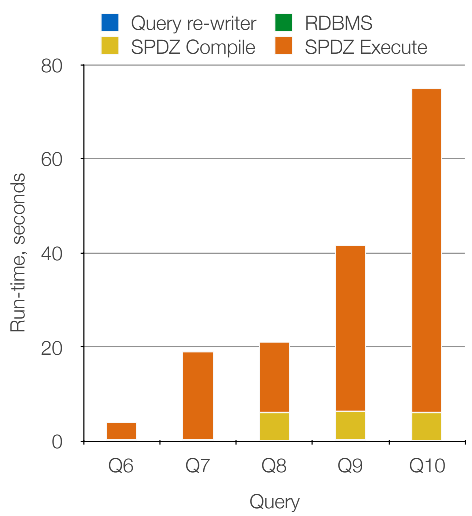
- Improvements in crypto over SPDZ
- Easier to use: full integration of offline and online phases

MAMBA = Multiparty AlgorithMs Basic Argot

- Python-like interface
- Greater functionality including more complex functions (e.g. trigonometric)

Targeting SQL-esque functionality

- Select, Project, Join, Union (SPJU) queries
- Aggregate operators (including versions with differential privacy)
- Subqueries (SQL IN statement)
- Group By
- Data types including Integer, String, Boolean, Fixed-point, Date




```
Q6:
SELECT lastname, firstname, diseasestate, birthdate, community_name, nation_name
FROM person
JOIN community ON community.community_id = person.residence
JOIN nation ON nation.nation_id = person.citizenship
JOIN person2diseasestate ON person2diseasestate.person_id = person.person_id
WHERE person2diseasestate.diseasestate = 'D' AND
person2diseasestate.transitionDate <= '04-03-2017'

Q7:
SELECT person2diseasestate.diseasestate
FROM person
JOIN community ON community.community_id = person.residence
JOIN person2diseasestate ON person2diseasestate.person_id = person.person_id
WHERE person2diseasestate.transitionDate <= '04-10-2017'

Q8:
SELECT person2diseasestate.diseasestate
FROM person
JOIN community ON community.community_id = person.residence
JOIN person2diseasestate ON person2diseasestate.person_id = person.person_id
WHERE person2diseasestate.diseasestate = 'D' AND
person2diseasestate.transitionDate < '04-02-2017'
ORDER BY lastname
```

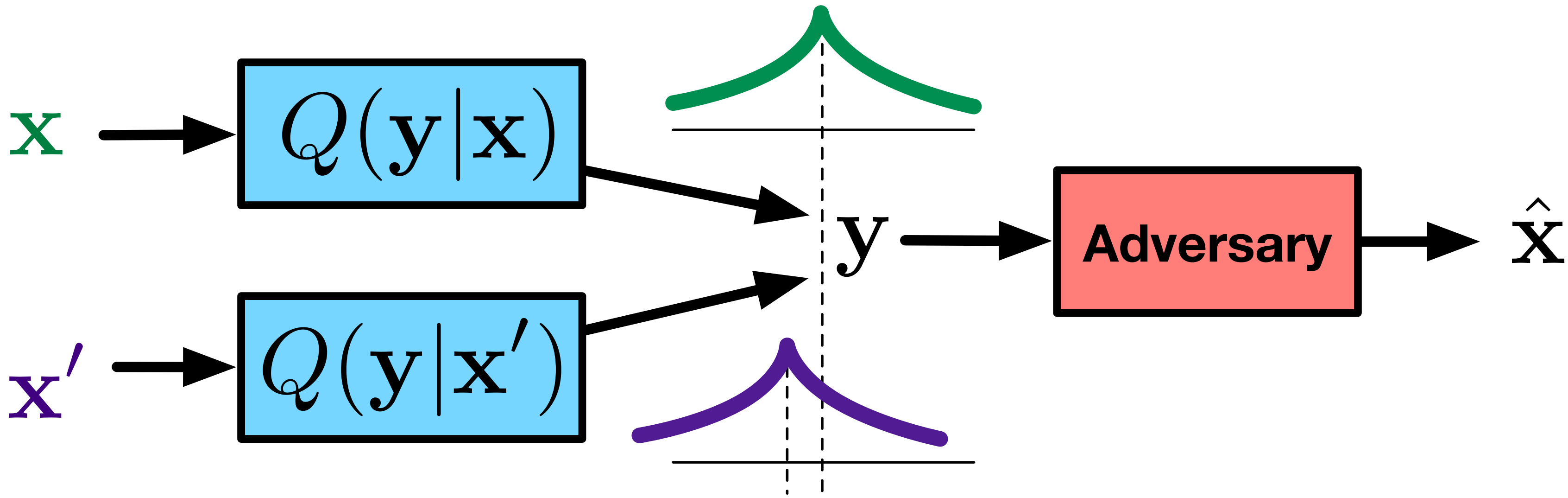
Q9: Q8, but less selective query —> more result rows

Q10: Q8, with even more result rows (but still not very many!)

Challenge:

- More complex queries can take extra time for execution.

Differential Privacy

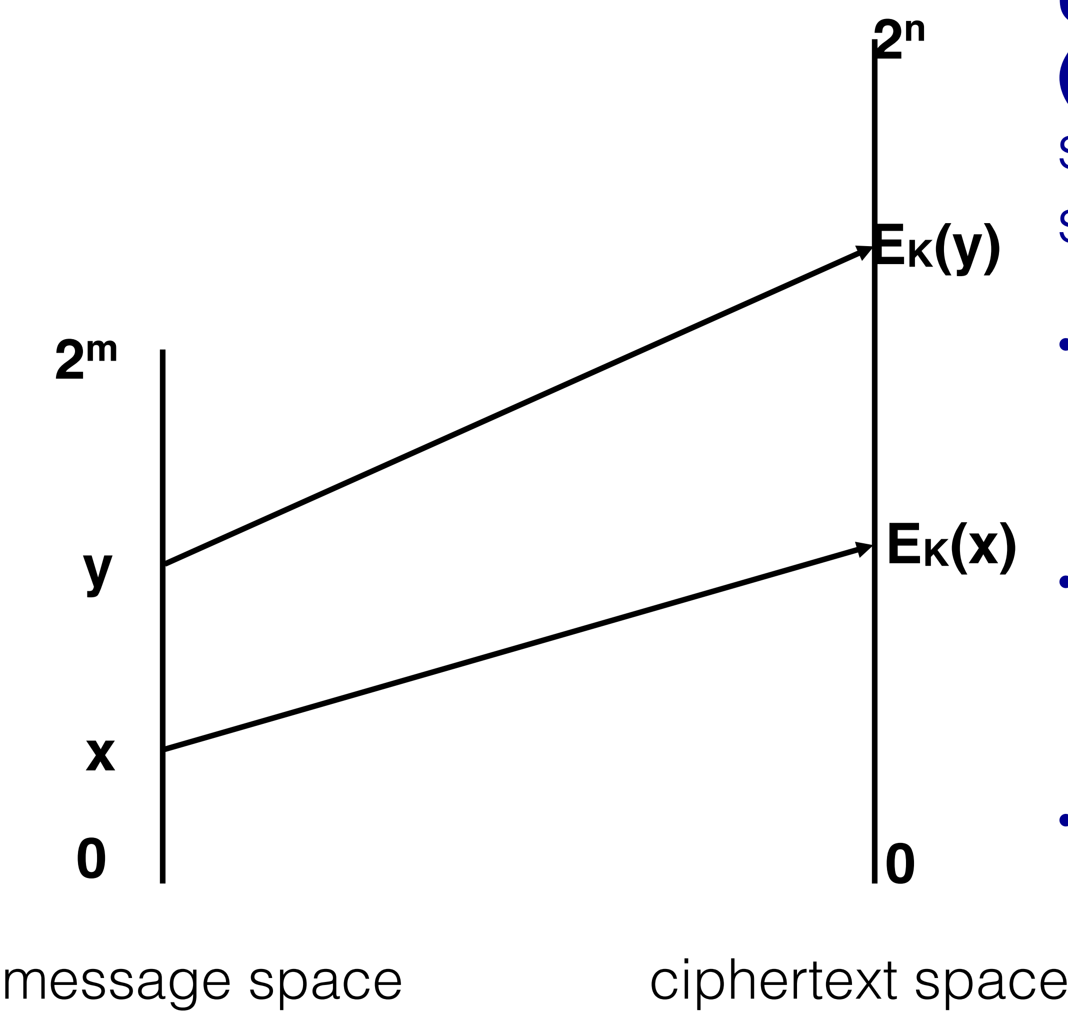


- Want to support core functions for SQL
- Need to generate noise that is friendly to MPC
- Extending query support to allow analyst to specify accuracy or confidence interval ("accuracy-first")

Some Research Questions

- **Problem:** We want symmetric encryption that can be efficiently computed "inside" the MPC.
  - Results: MPC-friendly symmetric encryption [GRRSS16]
- **Problem:** Want to better understand the privacy implications of using order-preserving encryption.
  - Results: How (in)secure is order-revealing encryption? [DDC16]
  - Ongoing work to try to fully characterize tradeoffs and develop best-possible solutions.
- **Problem:** The noise for differential privacy, as well as many functions we might want to compute make use of non-finite-field operations.
  - Goal: MPC-friendly differential privacy
  - For noise, currently using variant of [DKMMN06].
-

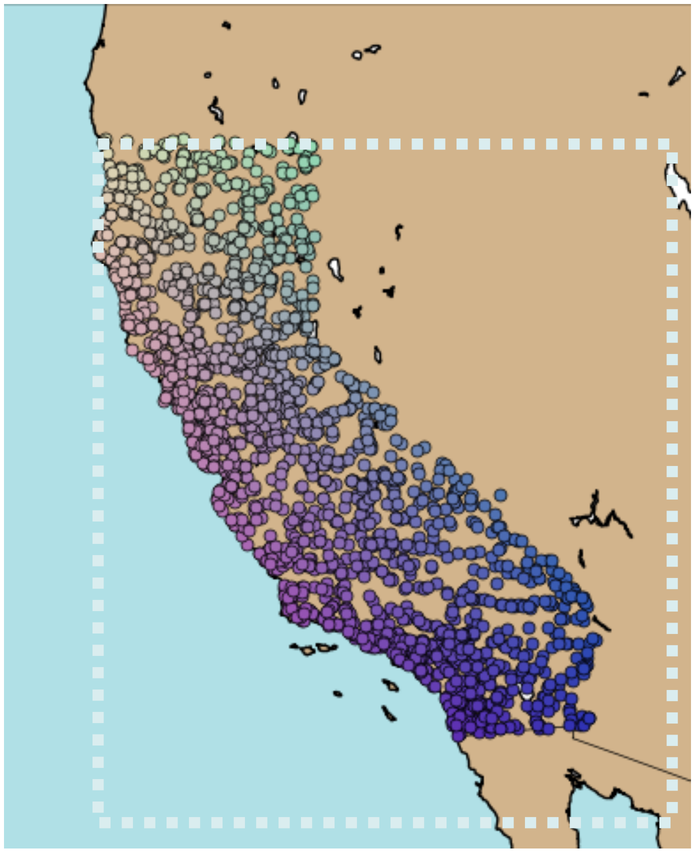
Order-Revealing Encryption (ORE) [AKSX'04,BCLO'09]



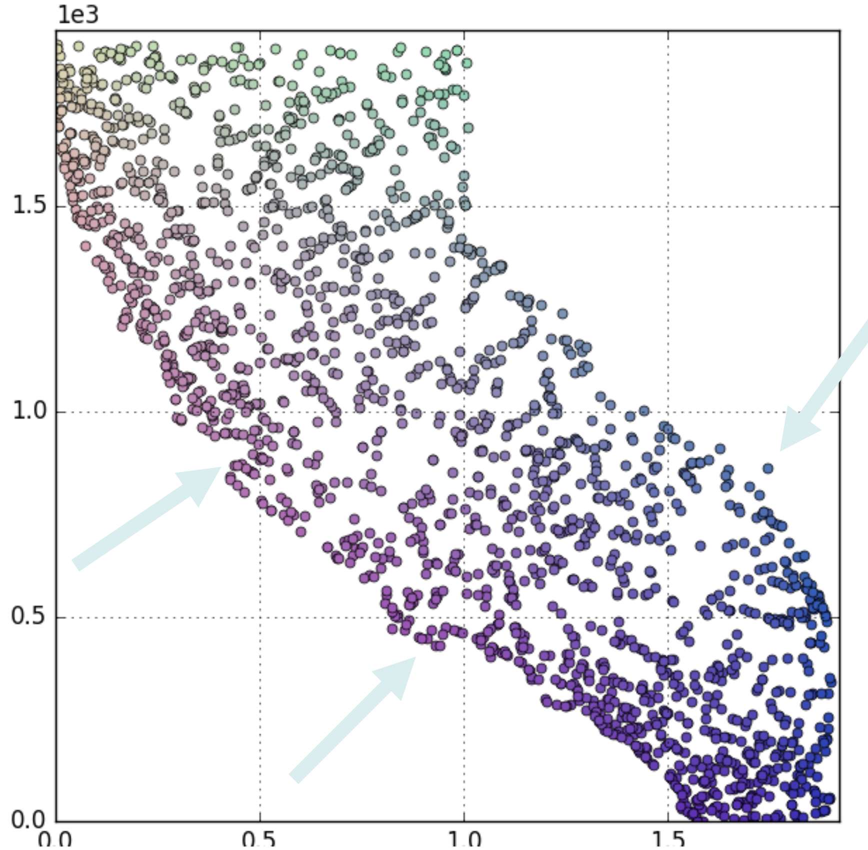
Order-Preserving Encryption (OPE): A symmetric encryption scheme that is deterministic and strictly increasing.

- **ORE** generalizes OPE. Both enable efficient computation of range queries on encrypted data.
- ORE/OPE are inherently less secure than standard encryption, subject to chosen-plaintext attacks.
- **Research approach:** Construct ORE schemes with best-possible security against passive attackers who only capture ciphertexts.

Plaintexts

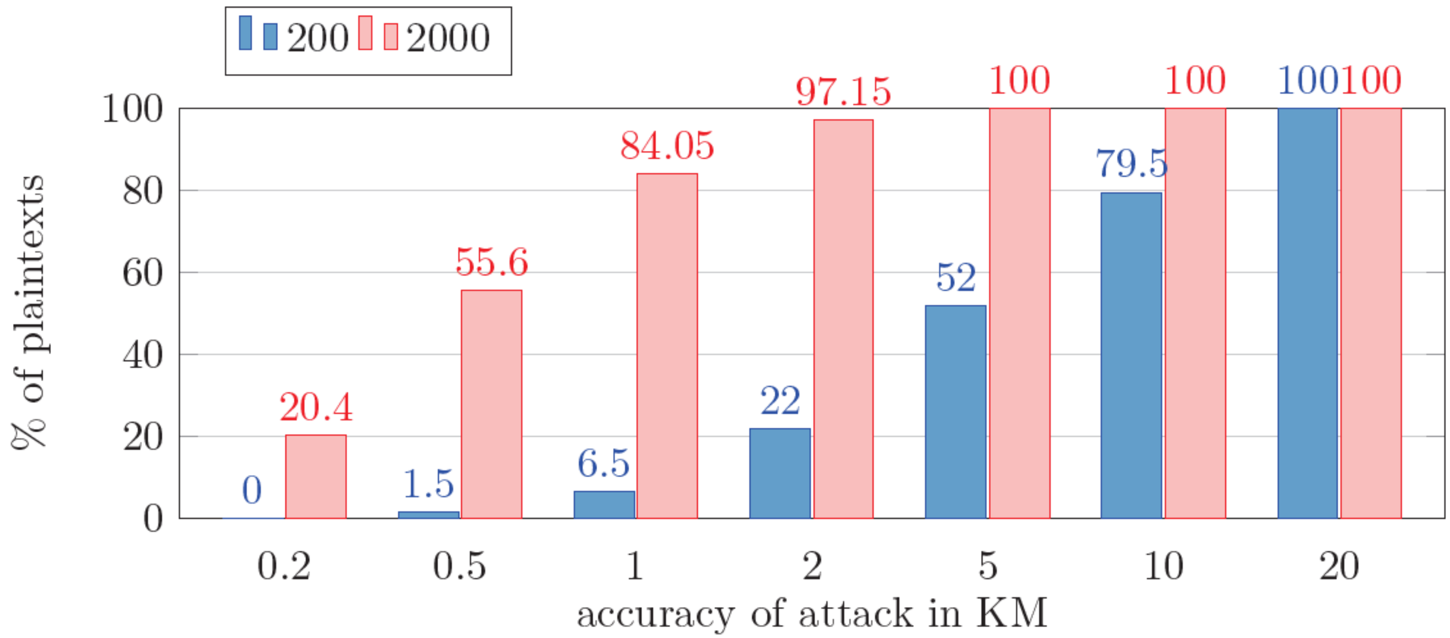


Ideal Leakage



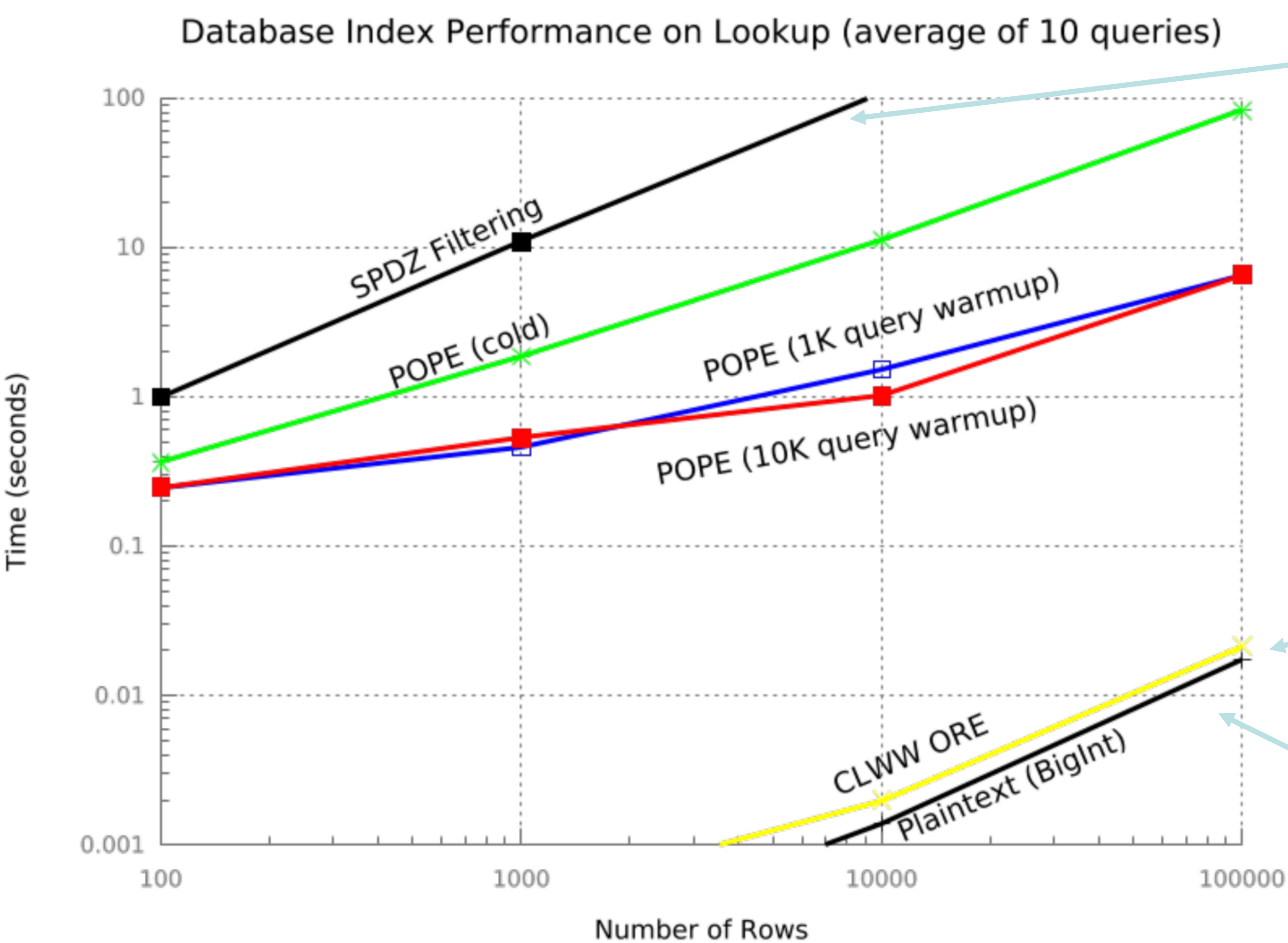
Data: Lat./long. for 21,000 road intersections (27 bits)
If bounding box is known: can guess 30% of points to within 50km

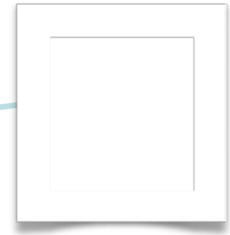
Problems with ORE [DDC16]

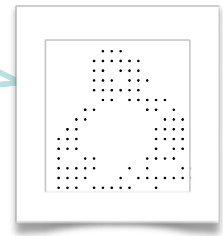


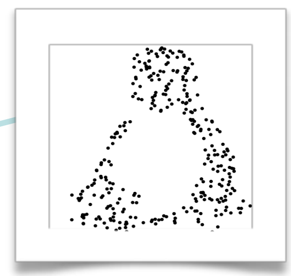
- Correlation causes information leakage, even for ideal ORE.
- Leaky ORE may be much leakier than previously thought.
- We should consider other primitives and different approaches for database protection (and cryptanalyze them).

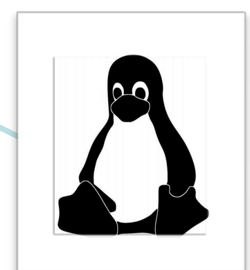
Searchable encryption using POPE Trees: leakage vs. performance











RUTGERS

DIFFERENTIAL PRIVACY IN JANA

DIMACS

Overview of differential privacy research

- Computed in SPDZ MPC engine in order to maintain privacy
- Aggregate query operators automatically replaced by DP variants by query re-writing as required by access control policies
- Supported operators
  - DP_COUNT (of fields matching where clause and sub-selects)
  - DP_HISTOGRAM with user-provided buckets
  - DP_SUM
  - AVG but our fixed point representation makes this mostly useless

Example: differentially private COUNT

- Two versions of each operator
  - Take a provided noise magnitude and implicit 95% confidence interval to compute the noise. Epsilon can be computed from this input
  - Take a provided epsilon and compute noise based on it and sensitivity

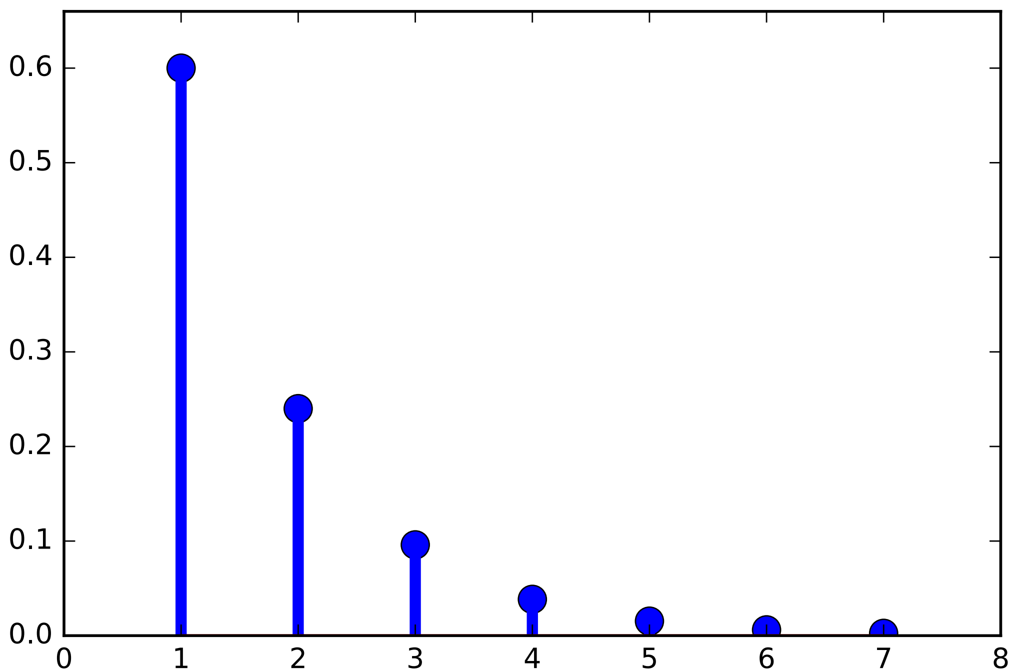
- COUNT has sensitivity 1
  - Add noise from a "discrete Laplace" distribution (2-sided geometric)
  - Similar to the approach in [DKMMN06]

- How do we do this in SPDZ or SCALE?
  - Need to use random number generation using biased coin flips

Discrete Laplace distributions

A geometric distribution (flip coins of bias *p*)

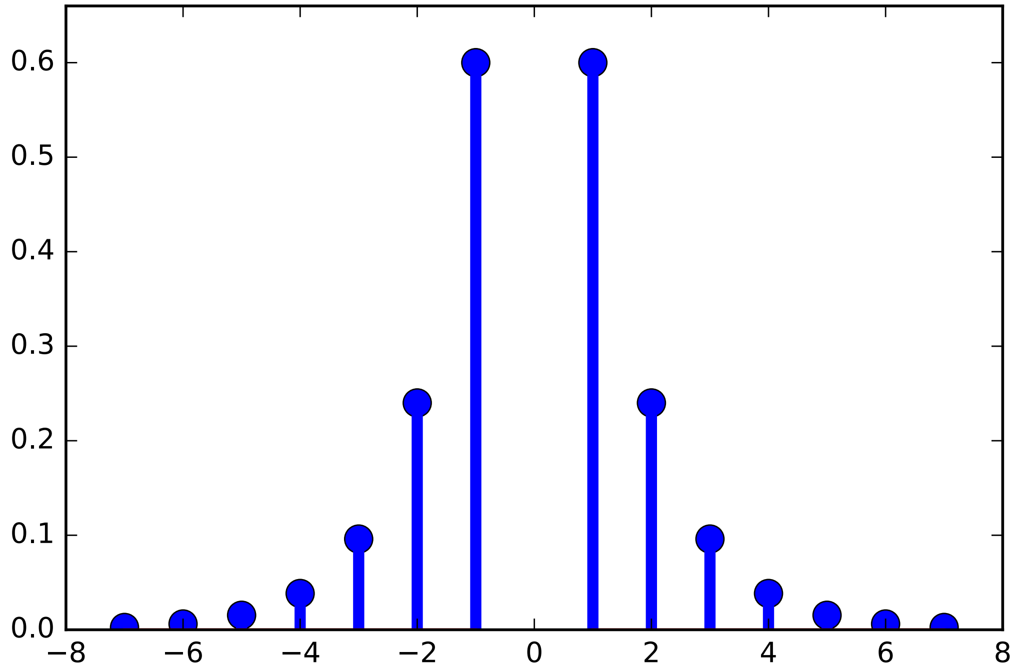
$$\mathbb{P}(G = k) = (1 - p)^{k-1} p$$



Discrete Laplace distributions

Two sided geometric distribution (add one coin of bias 0.5 to choose the sign:

$$\mathbb{P}(G_2 = k) = \frac{1}{2}(1-p)^{|k|-1}p$$



Discrete Laplace distributions

Add one more coin to of bias *a* to pick *k* = *0*:

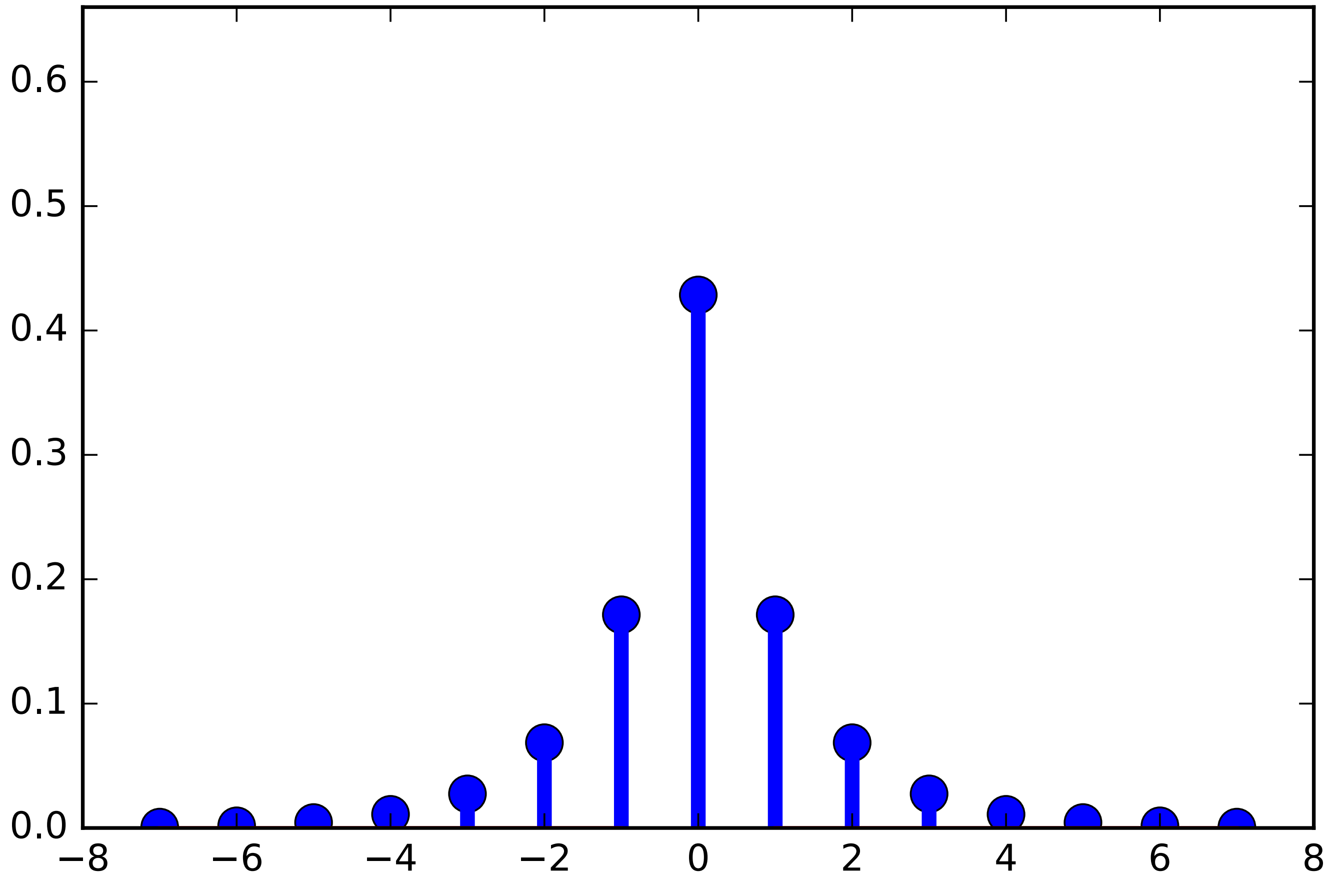
$$a = \frac{p}{2-p}$$

$$\mathbb{P}(Z = k) = \begin{cases} a & k = 0 \\ (1-a)\frac{1}{2}(1-p)^{|k|-1}p & k \neq 0 \end{cases}$$

$$\epsilon = \log \frac{\mathbb{P}(Z = k)}{\mathbb{P}(Z = k+1)} = \log \frac{1}{1-p}$$

Discrete Laplace distributions

1 biased + 1 fair + more biased coins for geometric distribution



Going from accuracy to privacy

Interpreting the DP privacy risk may be challenging:

- ε + sensitivity determine the noise distribution
- error + confidence level determine the noise distribution

$$\mathbb{P}(|Z| > \Delta) > 1 - q$$

- Numerically solve for required *p.*
- *Example:* want COUNT to be within ± 10 with 90% probability

Two issues

1. Geometric distribution has infinite support

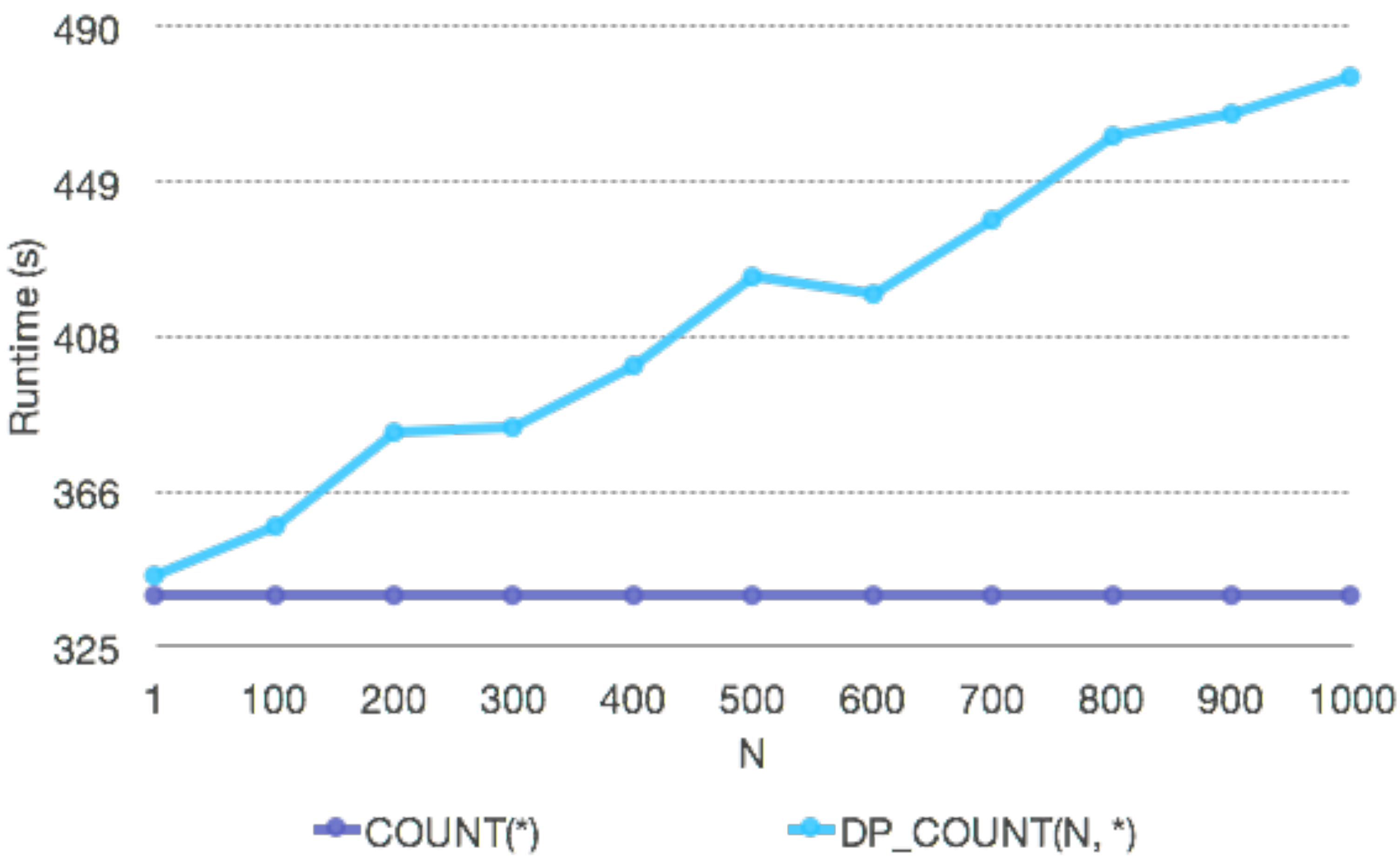
- Option 1: flip coins until success
  → run time depends on noise magnitude
  → side channels?

- Option 2: flip a fixed number of coins
  → can only simulate a finite-support distribution
  → relax from pure to approximate (ε, δ)-differential privacy

2. Does not scale as well to large domains

- Time to generate variables can become prohibitive
- Extensions to non-integer problems (e.g. SUM, AVERAGE) may be tricky

Privacy versus performance



ONGOING WORK

Extensions to other queries: SUM and AVG

Integer-valued data

- Makes sense for SUM but not AVERAGE.
- Same noise generation process works in theory.
- Large ranges may require too many coins in practice.

Real-valued data

- Currently restricted to fixed-precision calculations
- Rules for approximate averaging are unclear
- May need an alternative noise generation mechanism:
  - Lookup table via ”inverse CDF” method.
  - Something fancier?

Technical challenges with DP + MPC

Long term goal: machine learning algorithms running in Jana.

- Floating point vs. fixed point issue seems critical.
- Need multiplies to be as fast as adds.
- Should we use special procedures for linear algebra?
- What about large-scale iterative message-passing algorithms like SGD?

Privacy budgeting

Current state of privacy budgeting:

- Query returns privacy risk for each query
- Global budget using basic composition

Low-hanging fruit:

- Replace database budget with per-individual budget
- Replace basic composition with advanced composition [DRV10]

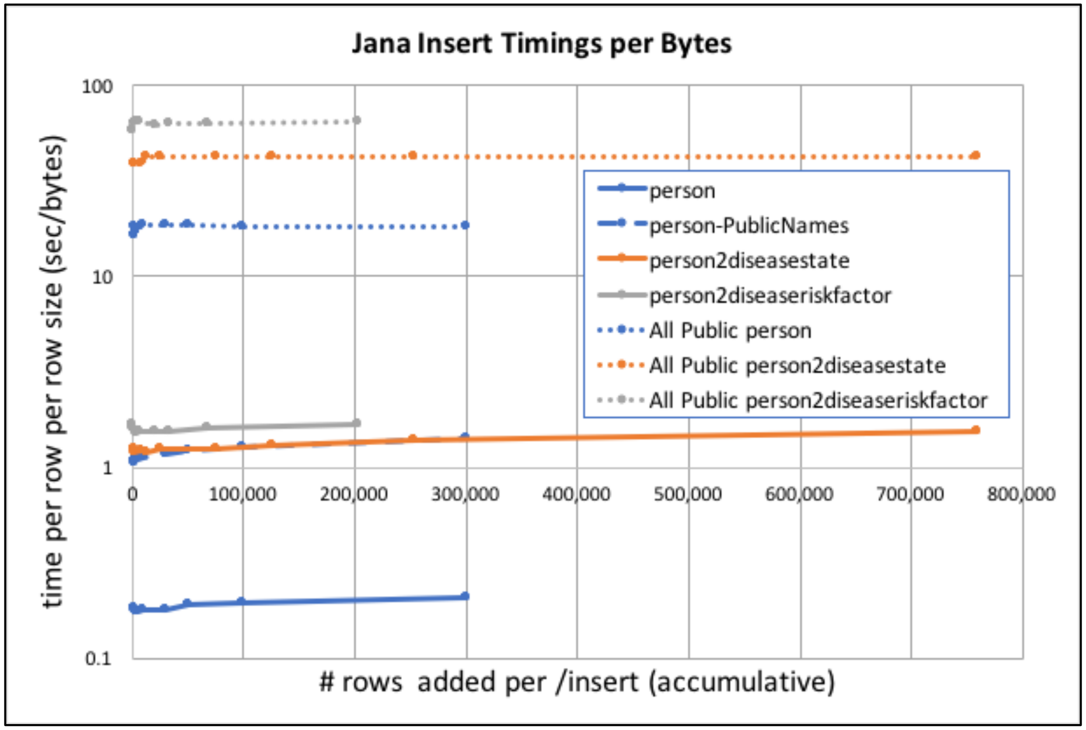
Getting fancier:

- Analyze privacy loss random variable more carefully
- Use (zero) concentrated-DP [DR16,BS16] or Rényi DP [M17] to track aggregate loss

Access control language

- Simple conditions for selecting controls, specified by system administrator
- Attributes of users vs. constants or attributes in data
  - Cardinality, Time window, Age of data
- Simple controls
  - Full detail, aggregates, counts
  - **Differentially private aggregates**
  - (Later) data masking
- Conflict-free rules, by language construction
- Natural language and JSON rule representations
- **Policy enforcement by query re-writing inside Jana**

Dealing with dynamic data



Data insertion in Jana can be fast!

How should we track and manage privacy loss in dynamic settings?

How can we make this more practical?

CONCLUSIONS

Recap

Jana is proving a useful platform for exploring the feasibility, scalability, flexibility, privacy, and limits of various privacy tools and methods.

- **MPC:** understanding the impact of standard database operations on speed and efficiency.
- **DP:** understanding how to adapt even simple mechanisms to practical constraints imposed by the MPC computation model.

Tons of work to do still!

Rutgers

Thanks!