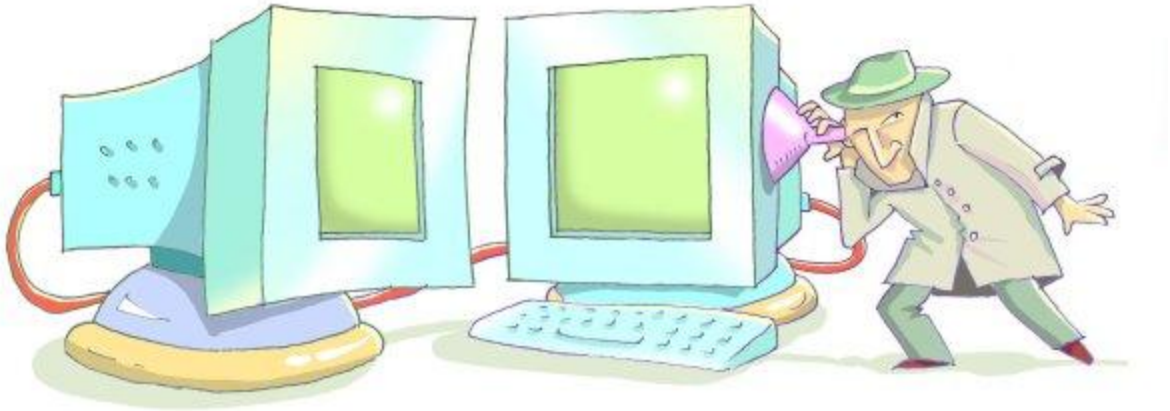


Implementing Differential Privacy & Side-channel attacks

CompSci 590.03

Instructor: Ashwin Machanavajjhala

Duke UNIVERSITY

Outline

- Differential Privacy Implementations
  - PINQ: Privacy Integrated Queries [McSherry SIGMOD '09]
  - Airavat: Privacy for MapReduce [Roy et al NDSS '10]
- Attacks on Differential Privacy Implementations
  - Privacy budget, state and timing attacks [Haeberlin et al SEC '11]
- Protecting against attacks
  - Fuzz [Haeberlin et al SEC '11]
  - Gupt [Mohan et al SIGMOD '12]

Duke UNIVERSITY

Differential Privacy

- Let A and B be two databases such that B = A – {t}.

- A mechanism M satisfies ε-differential privacy, if for all outputs O, and all such A, B

$$P(M(A) = O) \leq e^{\varepsilon} P(M(B) = O)$$

Differential Privacy

- Equivalently, let A and B be any two databases
- Let A Δ B = (A – B) U (B – A) … or the symmetric difference

- A mechanism M satisfies ε-differential privacy, if for all outputs O,

$$P(M(A) = O) \leq e^{\varepsilon \times |A \Delta B|} P(M(B) = O)$$

PINQ: Privacy Integrated Queries

[McSherry SIGMOD '09]

- Implementation is based on C#'s LINQ language

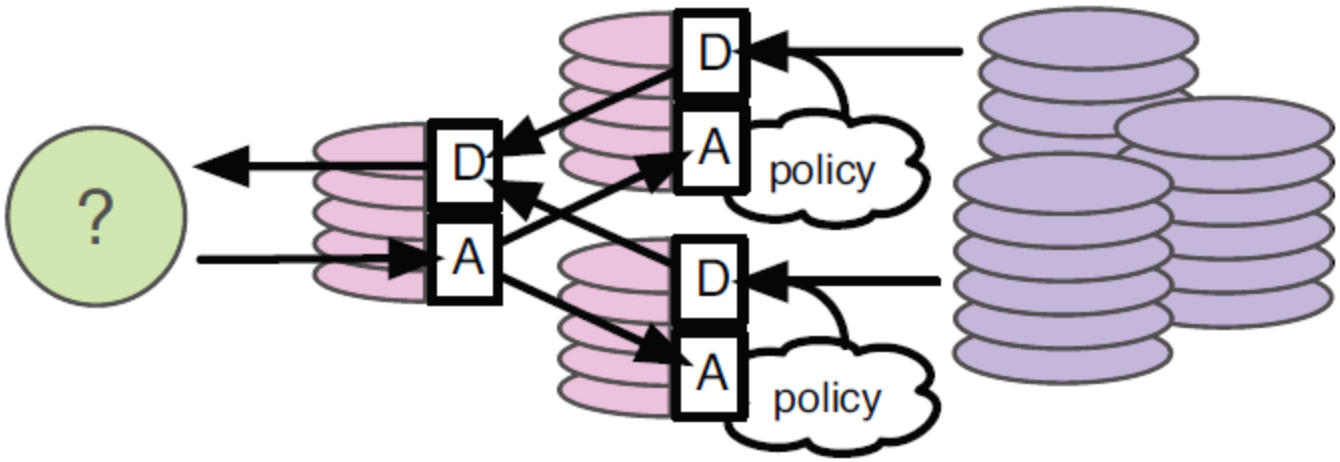
Example 1 Counting searches from distinct users in PINQ.

```
var data = new PINQueryable<SearchRecord>( ... ... );

var users = from record in data
            where record.Query == argv[0]
            groupby record.IPAddress

Console.WriteLine(argv[0] + ": " + users.NoisyCount(0.1));
```

PINQ



- An analyst initiates a PINQueryable object, which in turn recursively calls other objects (either sequentially or in parallel).
- A PINQAgent ensures that the privacy *budget* is not exceeded.

PINQAgent: Keeps track of privacy budget

Example 2 Implementing a fixed budget in a PINQAgent.

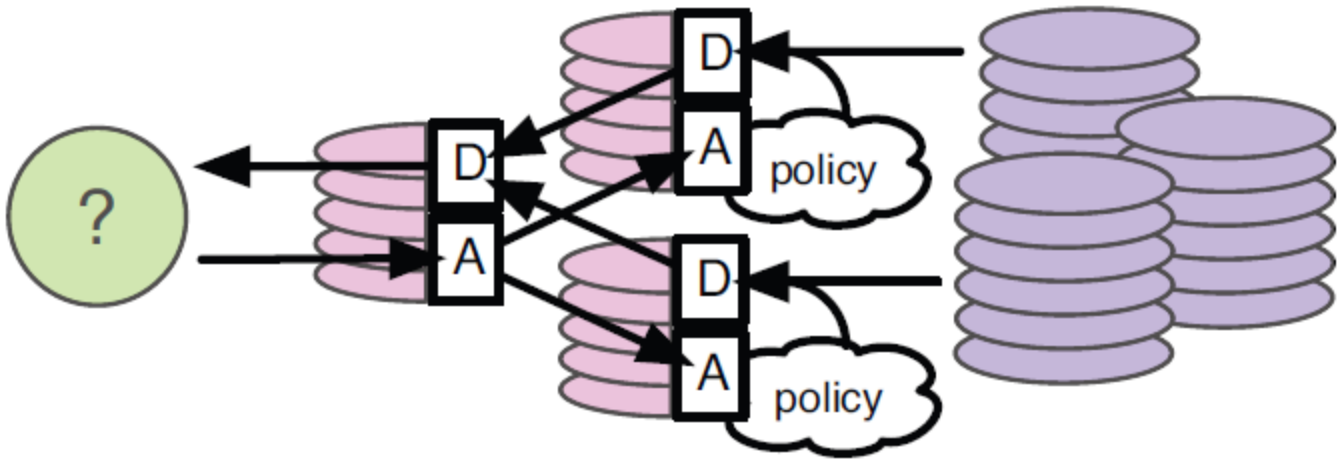
```
public class PINQAgentBudget : PINQAgent
{
    private double budget;

    public override bool Alert(double epsilon)
    {
        if (budget < epsilon)
            return false;

        budget = budget - epsilon;
        return true;
    }

    public PINQAgentBudget(double b)  { budget = b; }
}
```

PINQ: Composition



- When a set of operations O1, O2, … are performed sequentially, then the budget of the entire sequence is the sum of the ε for each operation.

- When the operations are run in parallel on disjoint subsets of the data, the privacy budget for the all the operations is the max ε.

Aggregation Operators

Example 3 [Abbreviated] Implementation of NoisyCount.

```
double NoisyCount(double epsilon)
{
    if (myagent.Alert(epsilon))
        return mysource.Count() + Laplace(1.0/epsilon);
    else
        throw new Exception("Access is denied");
}
```

Aggregation operators

Laplace Mechanism

- NoisyCount
- NoisySum

Exponential Mechanism

- NoisyMedian
- NoisyAverage

PINQ: Transformation

Sometimes aggregates are computed on transformations on the data

- *Where*: takes as input a predicate (arbitrary C# function), and outputs a subset of the data satisfying the predicate
- *Select*: Maps each input record into a different record using a C# function
- *GroupBy*: Groups records by key values
- *Join*: Takes two datasets, and key values for each and returns groups of pairs of records for each key.

PINQ: Transformations

Sensitivity can change once transformations have been applied.

- GroupBy: Removing a record from an input dataset A, can change one group in the output T(A). Hence, $|T(A) \Delta T(B)| = 2 |A \Delta B|$
- Hence, the implementation of GroupBy multiplies ε by 2 before recursively invoking the aggregation operation on each group.
- Join can have a much larger (unbounded) sensitivity.

Duke UNIVERSITY

Example

Example 5 Measuring query frequencies in PINQ.

```
// prepare data with privacy budget
var agent = new PINQAgentBudget(1.0);
var data  = new PINQueryable<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));
```

Outline

- Differential Privacy Implementations
  - PINQ: Privacy Integrated Queries [McSherry SIGMOD '09]
  - Airavat: Privacy for MapReduce [Roy et al NDSS '10]
- Attacks on Differential Privacy Implementations
  - Privacy budget, state and timing attacks [Haeberlin et al SEC '11]
- Protecting against attacks
  - Fuzz [Haeberlin et al SEC '11]
  - Gupt [Mohan et al SIGMOD '12]

Duke UNIVERSITY

Covert Channel

- Key assumption in differential privacy implementations: **The querier can *only* observe the result of the query, and nothing else.**
  - This answer is guaranteed to be differentially private.

- In practice: The querier can observe other effects.
  - E.g, Time taken by the query to complete, power consumption, etc.

  - Suppose a system takes 1 minute to answer a query if Bob has cancer and 1 micro second otherwise, then based on query time the adversary may know that Bob has cancer.

Duke UNIVERSITY

Threat Model

- Assume the adversary (querier) does not have physical access to the machine.
  - Poses queries over a network connection.

- Given a query, the adversary can observe:
  - **Answer to their question**
  - **Time that the response arrives at their end of the connection**
  - **The system's decision to execute the query or deny (since the new query would exceed the privacy budget)**

Duke UNIVERSITY

Timing Attack

```
Function is_f(Record r){
    if(r.name = Bob && r. disease = Cancer)
        sleep(10 sec);          // or go into infinite loop, or throw exception
    return f(r);
}

Function countf(){
    var  fs = from record in data
                where (is_f(record))
    print fs.NoisyCount(0.1);
}
```

Timing Attack

```
Function is_f(Record r){
    if(r.name = Bob && r. disease = Cancer)
        sleep(10 sec);          // or go into infinite loop, or throw exception
    return f(r);
}
```

If Bob has Cancer, then the query takes > 10 seconds
If Bob does not have Cancer, then query takes less than a second.

Duke UNIVERSITY

Global Variable Attack

```
Boolean found = false;
Function f(Record r){
    if(found)       return 1;
    if(r.name = Bob && r.disease = Cancer){
            found = true; return 1;
    } else return 0;
}

Function countf(){
    var  fs = from record in data
                where (f(record))
    print fs.NoisyCount(0.1);
}
```

Global Variable Attack

```
Boolean found = false;
Function f(Record r){
    if(found)       return 1;
    if(r.name = Bob && r.disease = Cancer){
        found = true; return 1;
    } else return 0;
```

Typically, the Where transformation does not change the sensitivity of the aggregate (each record transformed into another value).
But, this transformation changes the sensitivity – if Bob has Cancer, then all subsequent records return 1.

Duke UNIVERSITY

Privacy Budget Attack

```
Function is_f(Record r){
    if(r.name = Bob && r.disease = Cancer){
        run a sub-query that uses a lot of the privacy budget;
    }
    return f(r);
}

Function countf(){
    var  fs = from record in data
                where (f(record))
    print fs.NoisyCount(0.1);
}
```

Privacy Budget Attack

```
Function is_f(Record r){
    if(r.name = Bob && r.disease = Cancer){
        run a sub-query that uses a lot of the privacy budget;
    }
    return f(r);
}
```

If Bob does not has Cancer, then privacy budget decreases by 0.1.
If Bob has Cancer, then privacy budget decreases by 0.1 + Δ.

Even if adversary can't query for the budget, he can detect the change in budget by counting how many more queries are allowed.

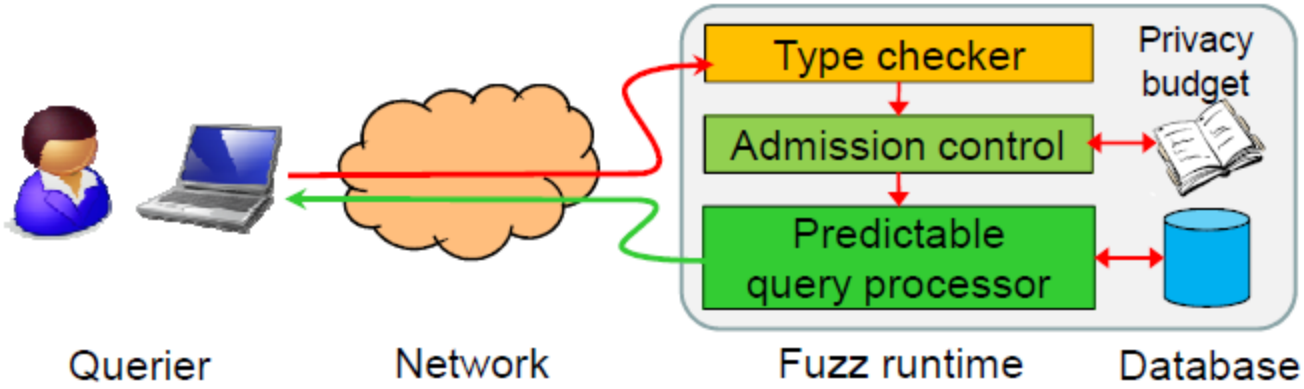
Duke UNIVERSITY

Outline

- Differential Privacy Implementations
  - PINQ: Privacy Integrated Queries [McSherry SIGMOD '09]
  - Airavat: Privacy for MapReduce [Roy et al NDSS '10]
- Attacks on Differential Privacy Implementations
  - Privacy budget, state and timing attacks [Haeberlin et al SEC '11]
- Protecting against attacks
  - Fuzz [Haeberlin et al SEC '11]
  - Gupt [Mohan et al SIGMOD '12]

Duke UNIVERSITY

Fuzz: System for avoiding covert-channel attacks



- Global variables are not supported in this language, thus ruling our *state attacks*.
- **Type checker** rules out *budget-based channels* by statically checking the sensitivity of a query *before* they are executed
- **Predictable query processor** ensures that each microquery takes the same amount of time, ruling out *timing attacks*.

Fuzz Type Checker

Primitive	Arguments	Return value
`map db f T d`	Database `db`, function `f`, timeout `T`, default value `d`	Database
`split db p T`	Database `db`, boolean predicate `p`, timeout `T`	Two databases
`count db`	Database `db`	Noised $\lvert db \rvert$
`sum db`	Database `db`	Noised $\sum_i db_i$

- A primitive is critical if it takes db as an input.
- Only four critical primitives are allowed in the language
  - No other code is allowed.
- A type system that can infer an upper bound on the sensitivity of any program (written using the above critical primitives). [Reed et al ICFP '10]

Duke UNIVERSITY

Handling timing attacks

- Each microquery takes exactly the same time T
- If it takes less time – delay the query
- If it takes more time – abort the query
  - **But this can leak information!**
  - **Wrong Solution**

Duke UNIVERSITY

Handling timing attacks

- Each microquery takes exactly the same time T
- If it takes less time – delay the query
- If it takes more time – **return a *default* value**

Fuzz Predictable Transaction

- P-TRANS (λ, a, T, d)
  - λ : function
  - a : set of arguments
  - T : Timeout
  - d : default value

- Implementing P-TRANS (λ, a, T, d) requires:
  - Isolation: Function λ(a) can be aborted without waiting for any other function
  - Preemptability: λ(a) can be aborted in bounded time
  - Bounded Deallocation: There is a bounded time needed to deallocate resources associated with λ(a)

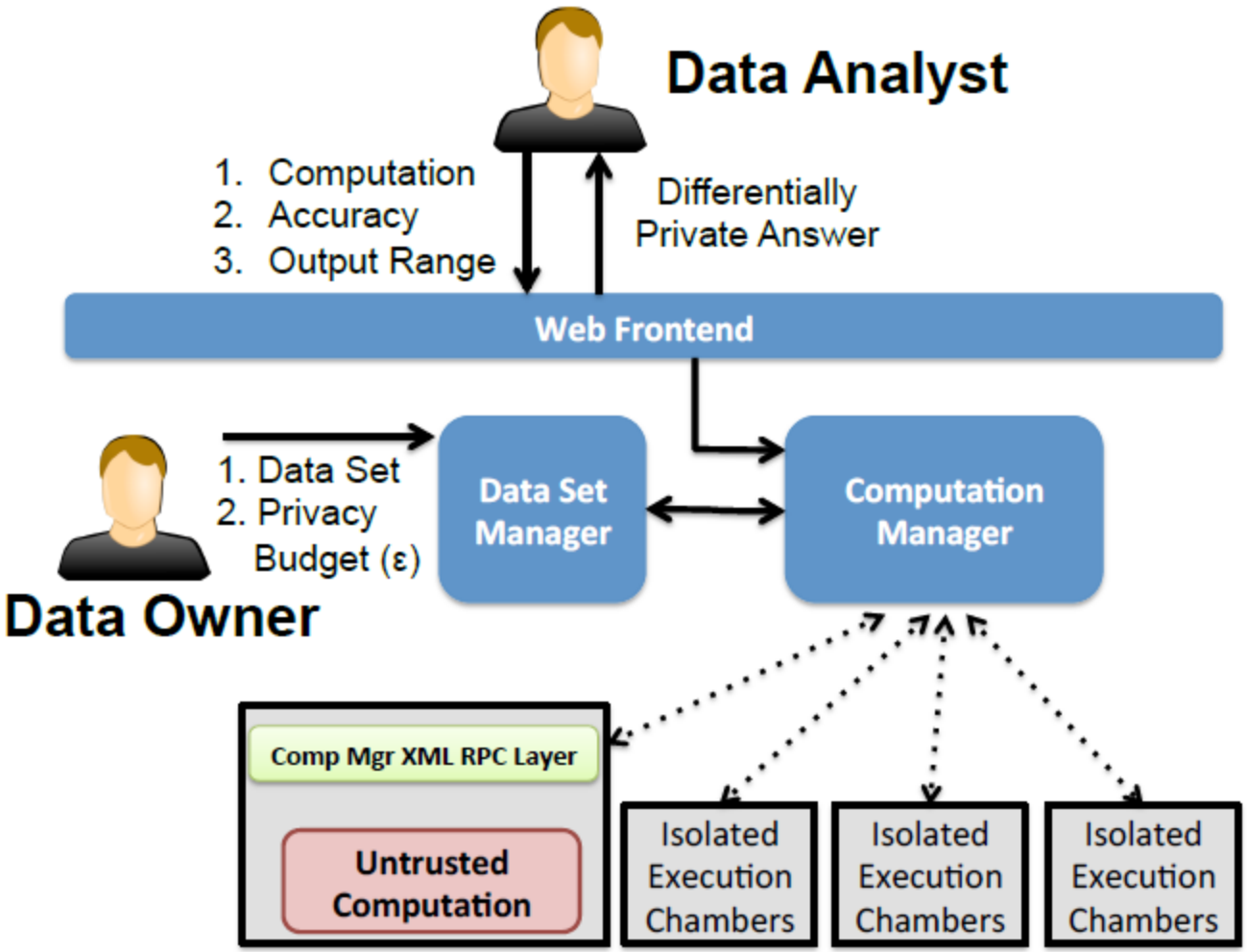
Duke UNIVERSITY

Outline

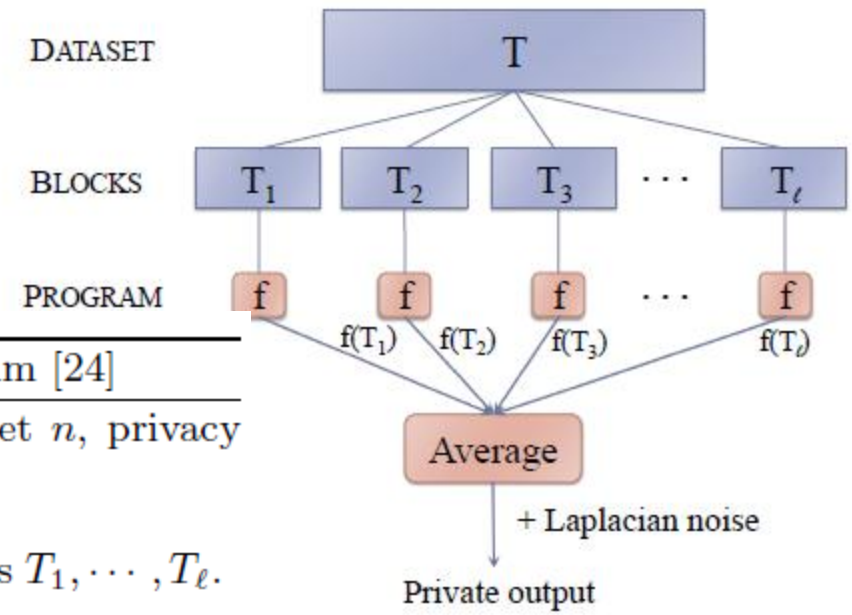
- Differential Privacy Implementations
  - PINQ: Privacy Integrated Queries [McSherry SIGMOD '09]
  - Airavat: Privacy for MapReduce [Roy et al NDSS '10]
- Attacks on Differential Privacy Implementations
  - Privacy budget, state and timing attacks [Haeberlin et al SEC '11]
- Protecting against attacks
  - Fuzz [Haeberlin et al SEC '11]
  - Gupt [Mohan et al SIGMOD '12]

Duke UNIVERSITY

GUPT


Data Analyst
1. Computation
2. Accuracy
3. Output Range
Differentially Private Answer
Web Frontend
1. Data Set
2. Privacy Budget (ε)
Data Owner
Data Set Manager
Computation Manager
Comp Mgr XML RPC Layer
Untrusted Computation
Isolated Execution Chambers
Isolated Execution Chambers
Isolated Execution Chambers

GUPT: Sample & Aggregate Framework

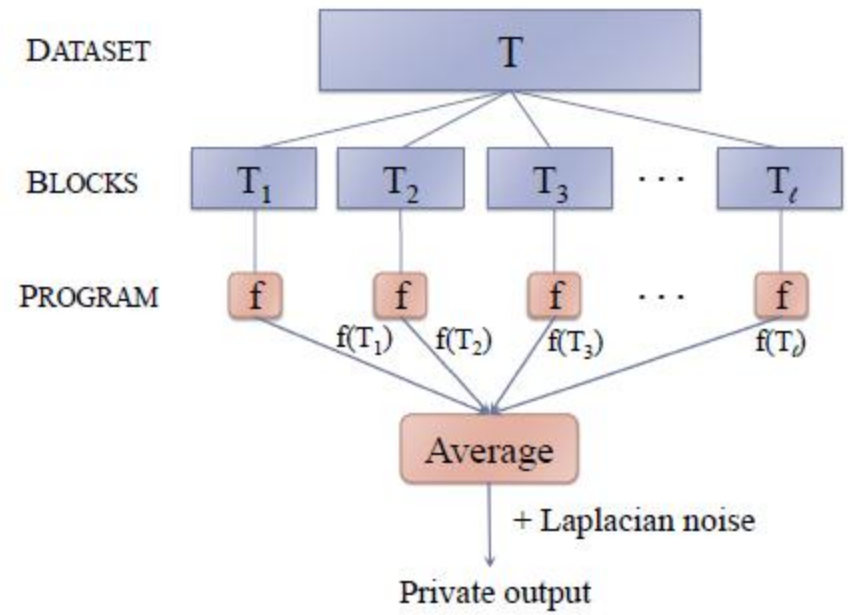


Algorithm 1 Sample and Aggregate Algorithm [24]

Input: Dataset $T \in \mathbb{R}^n$, length of the dataset n, privacy parameters ϵ, output range (min, max).

1: Let $\ell = n^{0.4}$
2: Randomly partition T into ℓ disjoint blocks $T_1, \cdots, T_\ell$.
3: **for** $i \in \{1, \cdots, \ell\}$ **do**
4: $\quad O_i \leftarrow$ Output of user application on dataset T_i.
5: $\quad$ If $O_i > \max$, then $O_i \leftarrow \max$.
6: $\quad$ If $O_i < \min$, then $O_i \leftarrow \min$.
7: **end for**
8: $A \leftarrow \frac{1}{\ell} \sum_{i=1}^{\ell} O_i + \mathsf{Lap}(\frac{|\max - \min|}{\ell \cdot \epsilon})$

Sample and Aggregate Framework



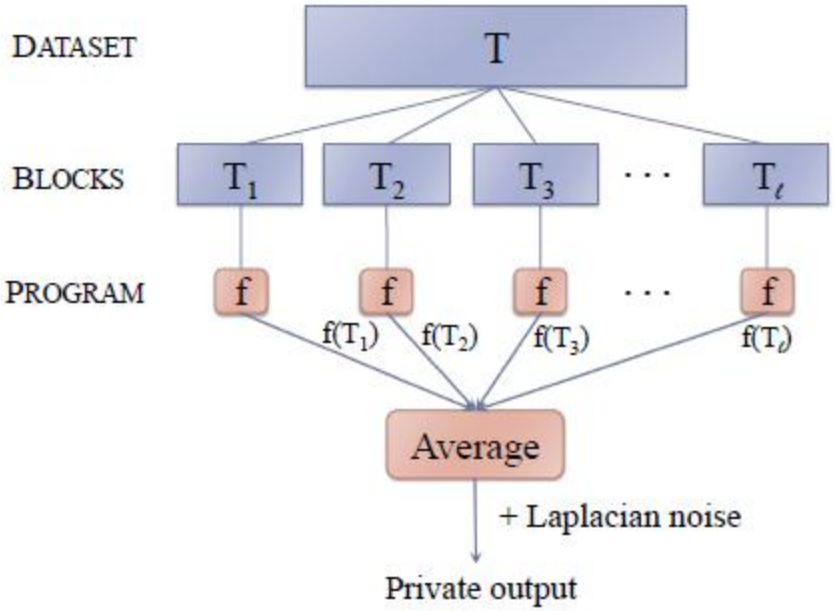
- S = range of the output
- L = number of blocks

Recall from previous lecture:

Theorem [Smith STOC '09]: Suppose database records are drawn i.i.d. from some probability distribution P, and the estimator (function *f*) is asymptotically normal at P. Then if L = o(√n), then **the average output by the Sample Aggregate framework converges to the true answer to *f*.**

Estimating the noise

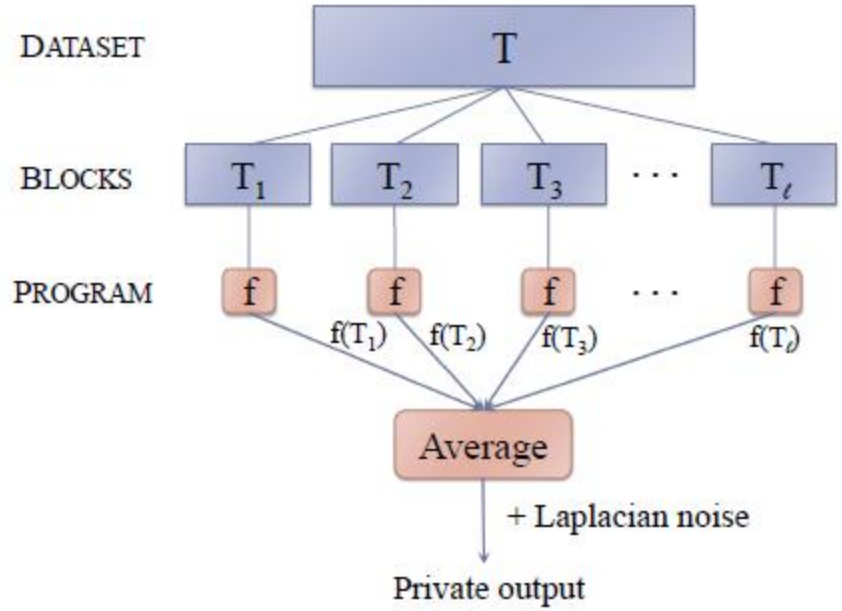
- Sensitivity of the aggregation function = S/L
  - S = range of the output
  - L = number of blocks



- Sensitivity is independent of the actual program *f*

- ***Therefore, GUPT avoids attacks using privacy budget as the covert channel.***

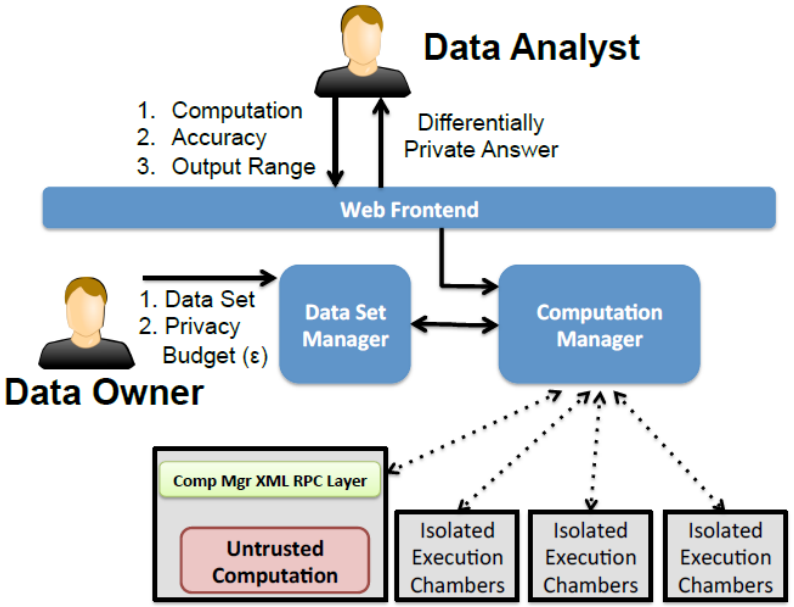
Estimating the noise



- Sensitivity of the aggregation function = S/L
  - S = range of the output
  - L = number of blocks
- Output range can be :
  - Specified by analyst, or
  - α^{th} and $(100 - \alpha)^{th}$ percentiles can be estimated using Exponential Mechanism, and a Windsorized mean can be used as the aggregation function.

Handling Global State attacks

- The function is computed on each block in an *isolated execution environment.*
  - Analyst sees only the final output, and cannot see any intermediate output or static variables.
  - Global variables can't inflate the sensitivity of the computation (like in the example we saw) … because the sensitivity only depends on S and L and not on the function itself.



Handling Timing Attacks

Same is in Fuzz …

- Fix some estimate T on the maximum time allowed for any computation (on a block)
- If computation finishes earlier, then wait till time T elapses
- If computation takes more time, stop and return a default value.

Comparing the two systems

GUPT	FUZZ
• Allows arbitrary computation. But, accuracy is guaranteed for certain estimators.	• Allows only certain critical operations.
• *Privacy-budget attack:* Sensitivity is controlled by S (output range) and L (number of blocks) that are statically estimated	• *Privacy-budget attack:* Sensitivity is statically computed.
• *State attack:* Adversary can't see any static variables.	• *State attack*: Global variables are disallowed
• *Timing attack*: Time taken across all blocks is predetermined.	• *Timing Attack*: Time taken across all records is predetermines

Duke UNIVERSITY

Summary

- PINQ (and Airavat) are frameworks for differential privacy that allow any programmer to incorporate privacy without needing to know how to do Laplace or Exponential mechanism.

- Implementation can disclose information through side-channels
  - Timings, Privacy-budget and State attacks

- Fuzz and GUPT are frameworks that disallow these attacks by
  - Ensuring each query takes a bounded time on all records or blocks
  - Sensitivity is statically estimated (rather than dynamically)
  - Global static variables are either inaccessible to adversary or disallowed

Open Questions

- Are these the only attacks that can be launched against a differential privacy implementation?

- Current implementations only simple algorithms for introducing privacy – Laplace and Exponential mechanisms. Optimizing error for batches of queries and advanced techniques (e.g., sparse vector) are not implemented. Can these lead to other attacks?

- Does differential privacy always protect against disclosure of sensitive information in all situations?
  - **NO … not when individuals in the data are correlated. More in the next class.**

Duke UNIVERSITY

References

F. McSherry, “PINQ: Privacy Integrated Queries”, SIGMOD 2009

I. Roy, S. Setty, A. Kilzer, V. Shmatikov, E. Witchel, “Airavat: Security and Privacy for MapReduce”, NDSS 2010

A. Haeberlin, B. Pierce, A. Narayan, “Differential Privacy Under Fire”, SEC 2011

J. Reed, B. Pierce, M. Gaboardi, “Distance makes types grow stronger: A calculus for differential privacy”, ICFP 2010

P. Mohan, A. Thakurta, E. Shi, D. Song, D. Culler, “Gupt: Privacy Preserving Data Analysis Made Easy”, SIGMOD 2012

A. Smith, "Privacy-preserving statistical estimation with optimal convergence rates", STOC 2011

Duke UNIVERSITY