Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
    - The Minimality Attack & Simulatable Auditing
    - Privacy Social Networks
    - Active Attacks in Social Networks
  - Strong adversaries
  - Bridging the Gap
- A Success Story: OnTheMap

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
    - The Minimality Attack & Simulatable Auditing
    - Privacy Social Networks
    - Active Attacks in Social Networks
  - Strong adversaries
  - Bridging the Gap
- A Success Story: OnTheMap

Minimality Attack on Generalization

[Wong et al, VLDB 2007]

- K-Anonymity, L-Diversity, t-closeness try to maximize utility
  - They minimize number of generalization steps
- What is the impact of this?
- Example:
  - Dataset with one quasi-identifier with two values, q1 and q2
  - q1 and q2 generalize to Q
  - Simplified notion of 2-diversity (at least two different values of sensitive attribute)

Example

Possible Input dataset

4 occurences of q1

QID	Cancer
q1	Yes
q1	Yes
q1	No
q1	No
q2	No
q2	No

Already a 2-diverse generalization!

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Example

Possible Input dataset

3 occurences of q1

QID	Cancer
q1	Yes
q1	Yes
q1	No
q2	No
q2	No
q2	No

QID	Cancer
q1	Yes
q1	Yes
q2	No
q1	No
q2	No
q2	No

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Example

Possible Input dataset

3 occurences of q1

QID	Cancer
q1	Yes
Q	Yes
Q	No
q1	No
q2	No
q2	No

This is the best generalization!

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Example

Possible Input dataset

1 occurence of q1

QID	Cancer
q1	Yes
q2	Yes
q2	No
q2	No
q2	No
q2	No

QID	Cancer
q2	Yes
q2	Yes
q2	No
q1	No
q2	No
q2	No

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Example

Possible Input dataset

1 occurence of q1

QID	Cancer
q2	Yes
Q	Yes
Q	No
q2	No
q2	No
q2	No

This is the best generalization!

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Example

Possible Input dataset

1 occurence of q1

QID	Cancer
q2	Yes
Q	Yes
Q	No
q2	No
q2	No
q2	No

This is the best generalization!

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

There must be exactly 2 tuples with q1

Example

Possible Input dataset

2 occurences of q1

QID	Cancer
q1	Yes
q1	Yes
q2	No
q2	No
q2	No
q2	No

QID	Cancer
q2	Yes
q2	Yes
q1	No
q1	No
q2	No
q2	No

QID	Cancer
q1	Yes
q2	Yes
q1	No
q2	No
q2	No
q2	No

Already 2 diverse

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Example

Possible Input dataset

2 occurences of q1

QID	Cancer
q1	Yes
q1	Yes
q2	No
q2	No
q2	No
q2	No

QID	Cancer
q2	Yes
q2	Yes
q1	No
q1	No
q2	No
q2	No

If learning NO Cancer is OK, Then this is private

Output dataset

{q1,q2} → Q

("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Example

Possible Input dataset

2 occurences of q1

QID	Cancer
q1	Yes
q1	Yes
q2	No
q2	No
q2	No
q2	No

This is the ONLY generalization!

Output dataset

{q1,q2} → Q

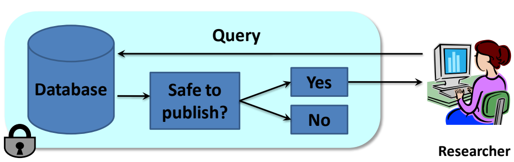
("2-diverse")

QID	Cancer
Q	Yes
Q	Yes
Q	No
Q	No
q2	No
q2	No

Minimality Attack

- The decisions made by the algorithm are used to attack the generalization algorithm.
- This is not specific to generalization.

Query Auditing



Database has numeric values (say salaries of employees).

Subset-AGG queries: MIN, MAX, SUM queries over subsets of the database.

Question: When to allow/deny queries?

Value-Based Auditing

- Let $a_1, a_2, \ldots, a_k$ be the answers to previous queries $Q_1, Q_2, \ldots, Q_k$.
- Let a_{k+1} be the answer to Q_{k+1}.

$$\mathbf{a_i = f(c_{i1}x_1, c_{i2}x_2, \ldots, c_{in}x_n), \quad i = 1 \ldots k+1}$$

$c_{im} = 1$ if Q_i depends on x_m

Check if any x_j has a unique solution.

Value-based Auditing

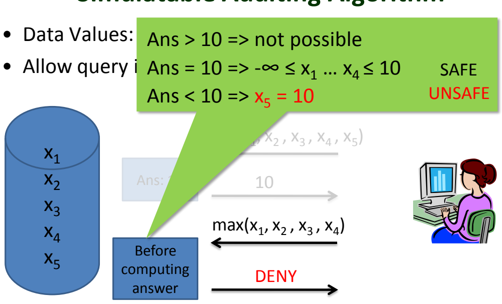
- Data Values: {x_1, … s: MAX.
- Allow query if va… red.

Simulatable Auditing

- An auditor is *simulatable* if the decision to deny a query Q_k is made based on information already available to the attacker.
  - Can use queries $Q_1, Q_2, \dots, Q_k$ and answers $a_1, a_2, \dots, a_{k-1}$
  - ***Cannot*** use a_k or the actual data to make the decision.
- Denials provably do not leak informaiton
  - Because the attacker could equivalently determine whether the query would be denied.
  - Attacker can mimic or *simulate* the auditor.

Simulatable Auditing Algorithm

- Data Values:
- Allow query i



Summary of Simulatable Auditing

- Decision to deny answers must be based on past queries answered in some (*many!*) cases.
- Denials can leak information if the adversary does not know all the information that is used to decide whether to deny the query.

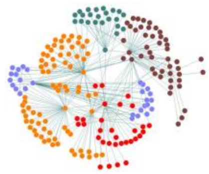
Summary of Minimality Attack

- The decisions made by the algorithm are used to attack the generalization algorithm.
  - The lattice traversal cannot be simulated by the adversary.
- This is not specific to generalization.
- Developing simulatable algorithms for generalizations is an active area of research.

Tutorial Outline

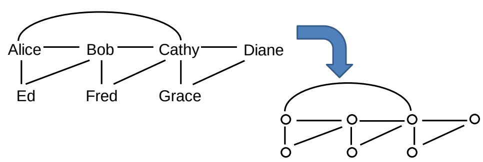
- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
    - The Minimality Attack & Simulatable Auditing
    - Privacy Social Networks
    - Active Attacks in Social Networks
  - Strong adversaries
  - Bridging the Gap
- A Success Story: OnTheMap

Social Network Data



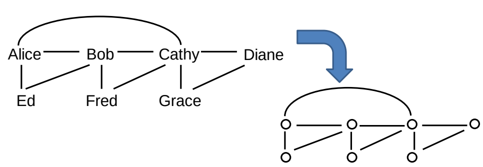
- Social networks: graphs where each node represents a social entity, and each edge represents certain relationship between two entities
- Example: email communication graphs, social interactions like in Facebook, Yahoo! Messenger, etc.

Privacy in Social Networks



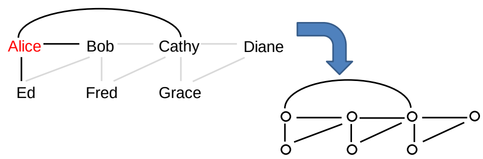
- Naïve anonymization
  - removes the label of each node and publish only the structure of the network
- Information Leaks
  - Nodes may still be re-identified based on network structure

Attacking an Anonymized Network



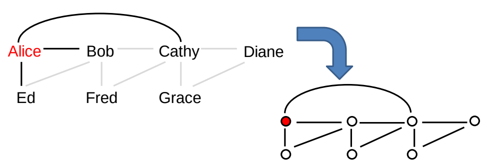
- Consider the above email communication graph
  - Each node represents an individual
  - Each edge between two individuals indicates that they have exchanged emails

Attacking an Anonymized Network



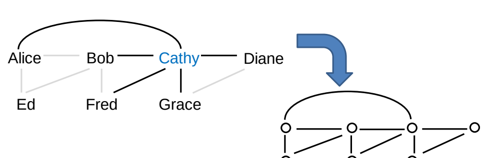
- Alice has sent emails to three individuals only

Attacking an Anonymized Network



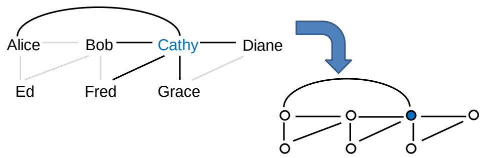
- Alice has sent emails to three individuals only
- Only one node in the anonymized network has a degree three
- Hence, Alice can re-identify herself

Attacking an Anonymized Network



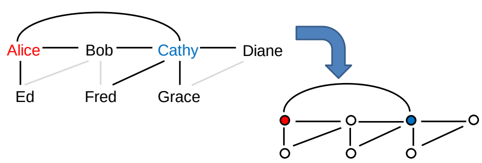
- Cathy has sent emails to five individuals

Attacking an Anonymized Network



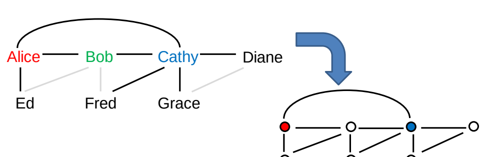
- Cathy has sent emails to five individuals
- Only one node has a degree five
- Hence, Cathy can re-identify herself

Attacking an Anonymized Network



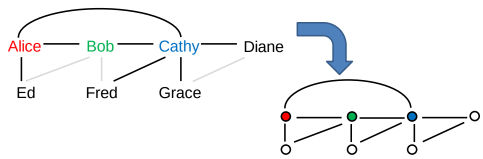
- Now consider that Alice and Cathy share their knowledge about the anonymized network
- What can they learn about the other individuals?

Attacking an Anonymized Network



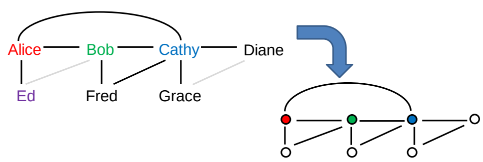
- First, Alice and Cathy know that only Bob have sent emails to both of them

Attacking an Anonymized Network



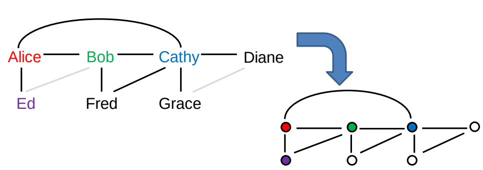
- First, Alice and Cathy know that only Bob have sent emails to both of them
- Bob can be identified

Attacking an Anonymized Network



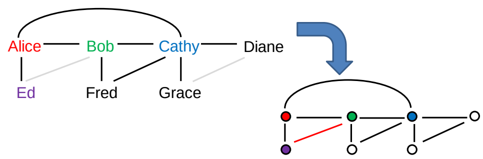
- Alice has sent emails to Bob, Cathy, and Ed only

Attacking an Anonymized Network



- Alice has sent emails to Bob, Cathy, and Ed only
- Ed can be identified

Attacking an Anonymized Network



- Alice and Cathy can learn that Bob and Ed are connected

Attacking an Anonymized Network

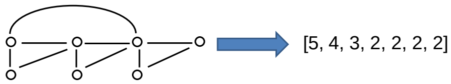
- The above attack is based on knowledge about the degrees of the nodes
- More sophisticated attacks can be launched given additional knowledge about the network structure, e.g., a subgraph of the network.
- Protecting privacy becomes even more challenging when the nodes in the anonymized network are labeled

K-degree Anonymity

[Liu and Terzi, SIGMOD 2008]

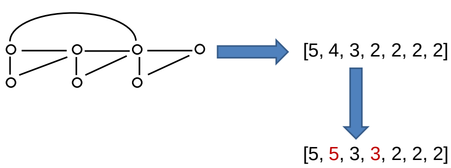
- Objective: prevent re-identification based on node degrees
- Solution: add edges into the graph, such that each node has the same degree as at least k-1 other nodes

K-degree Anonymity Algorithm



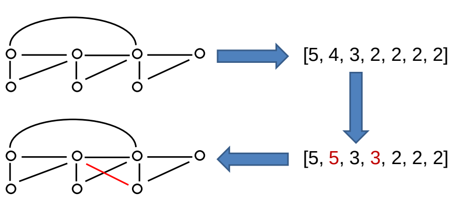
- Given a graph, calculate the degree of each node, and stores the degrees in a vector

K-degree Anonymity Algorithm



- Modify the degree vector, such that each degree appears at least k times

K-degree Anonymity Algorithm



- Add edges into the graph, such that the degrees of the nodes conform to the modified degree vector

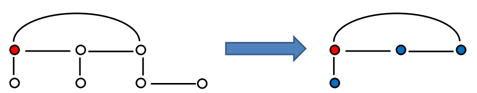
K-degree Anonymity Algorithm

- How do we modify the degree vector?
  - A dynamic programming algorithm can be used to minimize the L1 distance between the original and modified vectors
- How do we modify the graph according to the degree vector?
  - Greedily add edges into the graph to make the node degrees closer to the given vector

K-neighborhood Anonymity

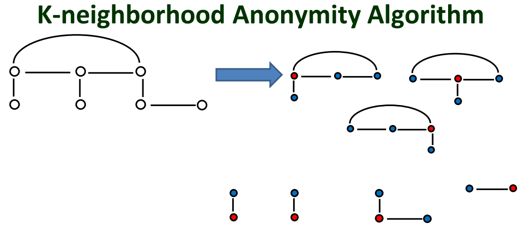
[Zhou and Pei, ICDE 2008]

- Neighborhood: sub-graph induced by one-hop neighbors



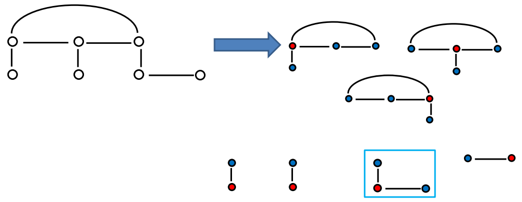
- Objective: prevent re-identification based on neighborhood structure
- Solution: add edges into the graph, such that each node has the same *neighborhood* as at least k-1 other nodes

K-neighborhood Anonymity Algorithm



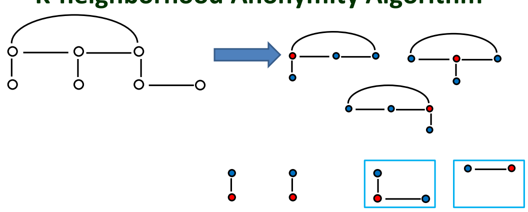
- Compute the neighborhood of each node

K-neighborhood Anonymity Algorithm



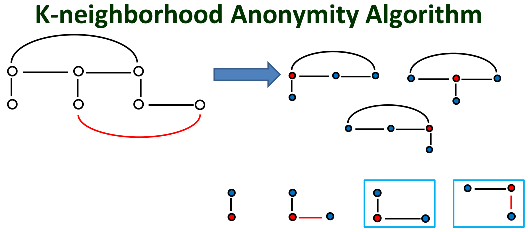
- While there is a node N whose neighborhood is not k-anonymous
  - Find a node N' whose neighborhood is similar to that of N
  - Greedily add edges in the graph to make the neighborhoods of N and N' isomorphic

K-neighborhood Anonymity Algorithm



- While there is a node N whose neighborhood is not k-anonymous
  - Find a node N' whose neighborhood is similar to that of N
  - Greedily add edges in the graph to make the neighborhoods of N and N' isomorphic

K-neighborhood Anonymity Algorithm

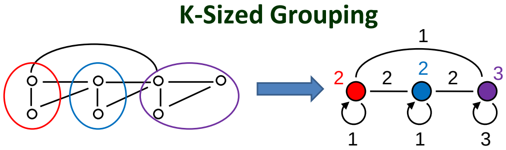


- While there is a node N whose neighborhood is not k-anonymous
  - Find a node N' whose neighborhood is similar to that of N
  - Greedily add edges in the graph to make the neighborhoods of N and N' isomorphic

K-neighborhood Anonymity Algorithm

- The algorithm always terminates: in the worst case it returns a complete graph
- How do we check whether two neighborhood structures are the same?
  - Graph isomorphism is NP-hard in general
  - But neighborhoods are usually small, in which case a brute-force checking is feasible
  - Some pre-processing can be done to reduce computation cost

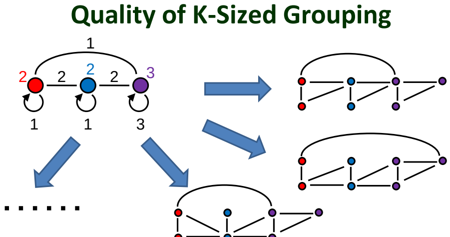
K-Sized Grouping



[Hay et al., VLDB 2008]

- Objective: prevent re-identification based on network structure
- Solution:
  - Partition the nodes into groups with sizes at least k
  - Coalesce the nodes in each group into a *super-node*
  - Each super-node has a weight that denotes its size
  - Super-nodes are connected by *super-edges* with weights

Quality of K-Sized Grouping

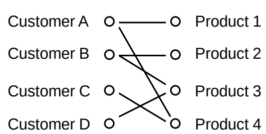


- A k-sized grouping represents a number of possible worlds
- The smaller number of possible worlds, the more accurate the anonymized network

A Simulated Annealing Algorithm

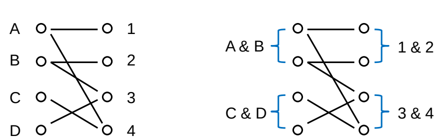
- Start from an arbitrary k-sized grouping of the graph
- Iteratively refine the grouping
  - Randomly transforms the grouping into another k-sized grouping, by splitting a group into two parts, or merging two groups, or moving a node from one group to another
  - If the new grouping is better, keep it; otherwise, fall back to the previous grouping with certain probability p
  - Decreases p by a certain amount before the next iteration
- Terminate when the algorithm converges

(k, l)-Grouping



- Targets at bipartite graphs with labeled nodes
- Assumes that the adversary does not have network structure knowledge
- Aims to conceal the associations between the labels

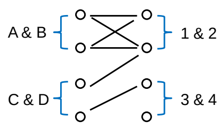
(k, l)-Grouping



- Partition the nodes on the left into k-sized groups
- Partition the nodes on the right into l-sized groups
- Unify the labels of the nodes in each group (reminiscent of generalization)

Unsafe (k, l)-Grouping

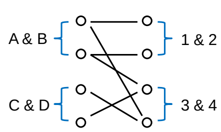
- Some (k, l)-grouping leaks information:
- Example:



- The above (2, 2)-grouping shows that both customers A and B have bought products 1 and 2

Safe (k, l)-Grouping

- A (k, l)-grouping is *safe*, if no two nodes in the same group are connected to a common neighbor
- Example: a safe (2, 2)-grouping



- Rationale: nodes in the same group should have sufficiently diverse neighbors (reminiscent of l-diversity)

Finding Safe (k, l)-Groupings

- Theorem: Finding a safe (k, l)-grouping is NP-hard in general
- Reduction from partitioning a graph into triangles
- Greedy algorithm: Iteratively add a node to a group so long as it is safe
- Works well when the bipartite graph is sparse enough

Summary of Social Network Publishing

- Structural information of a social network can be exploited to infer sensitive information
- Edge insertion and node grouping reduce the risk of re-identification
- Limitations
  - k-degree anonymity, k-neighborhood anonymity, and k-sized grouping only achieve k-anonymity
  - (k, l)-grouping cannot guard against attacks based on knowledge of network structure

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
    - The Minimality Attack & Simulatable Auditing
    - Privacy Social Networks
    - Active Attacks in Social Networks
  - Strong adversaries
  - Bridging the Gap
- A Success Story: OnTheMap

Active Attacks on Social Networks

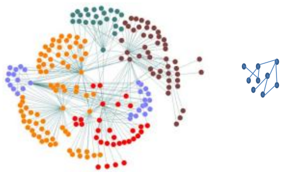
What can go wrong if an unlabeled graph is published?

[Backstrom et al., WWW 2007]

- Attacker may create a few nodes in the graph
  - Creates a few 'fake' Facebook user accounts.
- Attacker may add edges from the new nodes.
  - Create friends using 'fake' accounts.
- Goal: Discover an edge between two legitimate users.

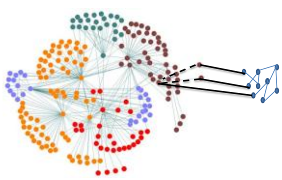
High Level View of Attack

- Step 1: Create a graph structure with the 'fake' nodes such that it can be identified in the anonymous data.



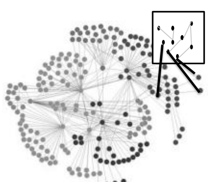
High Level View of Attack

- Step 2: Add edges from the 'fake' nodes to real nodes.



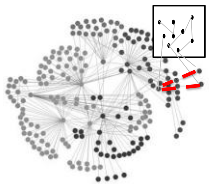
High Level View of Attack

- Step 3: From the anonymized data, identify fake graph due to its special graph structure.



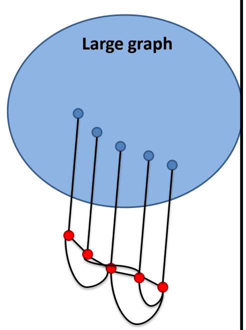
High Level View of Attack

- Step 4: Deduce edges by following links



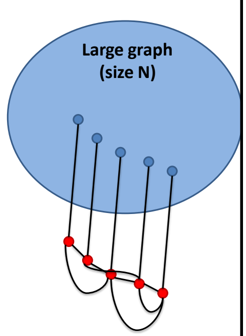
Details of the Attack

- Choose k real users $W = \{w_1, \ldots, w_k\}$
- Create k fake users $X = \{x_1, \ldots, x_k\}$
- Creates edges (x_i, w_i)
- Create edges (x_i, x_{i+1})
- Create all other edges in X with probability 0.5.



Uniqueness

X is guaranteed to be unique when k is $2+\delta \log N$, for small δ

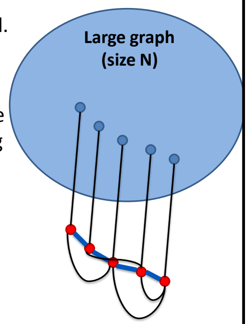


Recovery

Subgraph isomorphism is NP-hard.

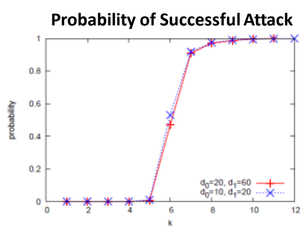
But since we have a path, with random edges, there is a simple brute force search with pruning algorithm.

Run Time: $\mathbf{O(N\ 2^{O(\log \log N)^2})}$



Works in Real Life!

- LiveJournal – 4.4 million nodes, 77 million edges
- Success all but guaranteed by adding 10 nodes.
- Recovery typically takes a second.



Summary of Social Networks

- Several simple algorithms proposed for variants of k-anonymity.
- Active attacks that add nodes and edges are shown to be very successful.
  - Reminiscent of Sybil attacks.
- Guarding against active attacks is an open area for research !

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
  - Strong adversaries
    - Differential Privacy
    - Algorithms satisfying Differential Privacy
  - Bridging the Gap
- A Success Story: OnTheMap

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
  - Strong adversaries
    - Differential Privacy
    - Algorithms satisfying Differential Privacy
  - Bridging the Gap
- A Success Story: OnTheMap

Impossibility of Semantic Disclosure Risk

- Suppose …
  - *salary* is a sensitive piece of information.
  - database *D* publishes average salaries of employees in different schools.
  - adversary knows: "Johannes earns $10 less than the average Cornell professor".

Given *D* we know exactly how much Johannes earns …

… even if Johannes' information is not in *D*!!

Differential Privacy

[Dwork, ICALP 2006]

INTUITION:

Releasing information from a database D should not increase the privacy risk of an individual x_i, if x_i does not appear in D.

Algorithm A satisfies ε-differential privacy if for every function f: dom(x_i) → {0,1}, and all prior distributions p on x_i ,

$$\log \left(\frac{\Pr[f(x_i) = 1 \mid \text{prior distribution on } x_i \text{ and } D - x_i]}{\Pr[f(x_i) = 1 \mid \text{prior on } x_i,\ D - x_i \text{ and } A(D)]} \right) \leq \varepsilon$$

Differential Privacy

[Dwork, ICALP 2006]

INTUITION:

Releasing info… increase the p… appear in D. …not …es not

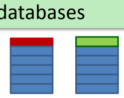
x_i not in D implies **D – xi = D**.
Hence, **no privacy breach**.

Algorithm A satisfies ε-differential privacy if for ev… unction f: dom(x_i) → {0,1}, and all prior distributions … on x_i ,

$$\log \left(\frac{\Pr[f(x_i) = 1 \mid \text{prior distribution on } x_i \text{ and } D - x_i]}{\Pr[f(x_i) = 1 \mid \text{prior on } x_i,\ D - x_i \text{ and } A(D)]} \right) \leq \varepsilon$$

Differential Privacy

Set of all possible input databases



Adversary knows x_1 is **either green or red**.

Adversary knows $\{x_2, x_3, \ldots, x_n\}$ are **blue**.

blue, green and red are three possibilities for each x_i.

Differential Privacy

For every pair of inputs that differ in one value

For every output …

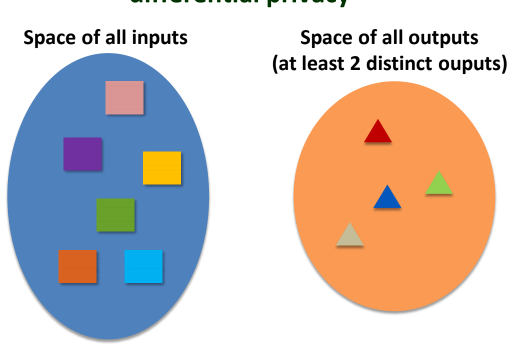
Adversary should not be able to distinguish between any D_1 and D_2 based on any O

$$\log\left(\frac{Pr[D_1 \rightarrow O]}{Pr[D_2 \rightarrow O]}\right) < \varepsilon \qquad (\varepsilon > 1)$$

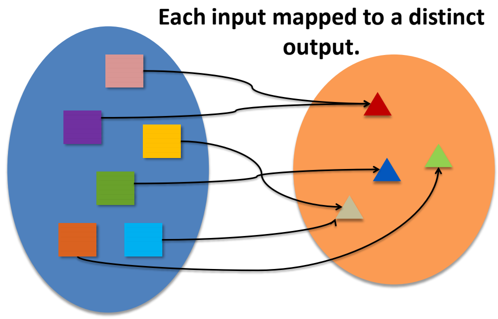
Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
  - Strong adversaries
    - Differential Privacy
    - Algorithms satisfying Differential Privacy
  - Bridging the Gap
- A Success Story: OnTheMap

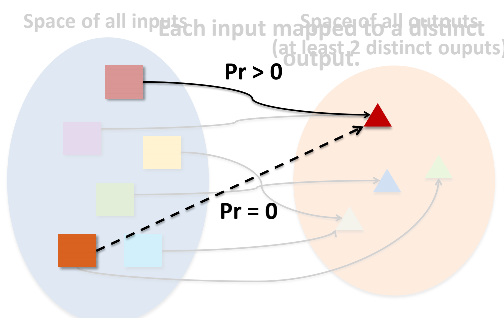
Deterministic Algorithms do not satisfy differential privacy



Deterministic Algorithms do not satisfy differential privacy

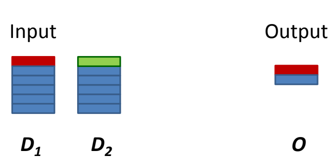


There exist two inputs that differ in one entry mapped to different outputs.



Random Sampling

- Also does not satisfy differential privacy



$$\log\left(\frac{\Pr[D1 \rightarrow O]}{\Pr[D2 \rightarrow O]}\right) = \infty$$

Random Sampling

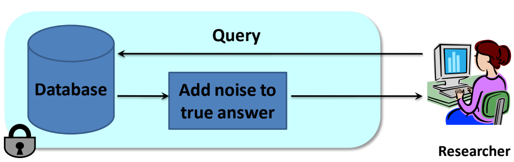
- Also does not satisfy differential privacy

[Chauduri et al., 2006]

- **If uniques are rare**, then differential privacy can be guaranteed with high probability.

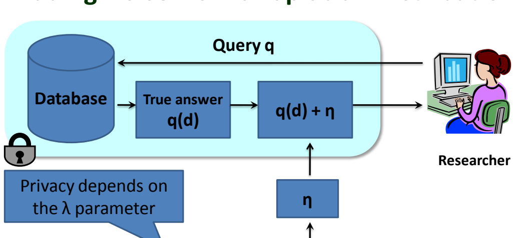
Most interesting data have many uniques!

Output Randomization



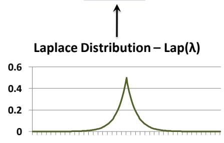
- Add noise to answers such that:
  - Each answer does not leak too much information about the database.
  - Noisy answers are close to the original answers.

Adding Noise from a Laplacian Distribution



$h(\eta) = \exp(-\eta / \lambda)$

Mean: 0,
Variance: $2\lambda^2$



Sensitivity of a Query – S(q)

[Dwork et al., TCC 2006]

Smallest number s.t. for any d, d' differing in one entry,

$$|| q(d) - q(d') || \leq S(q)$$

Example 1: **SUBSET-AGG queries**

- S(q) = |*b* – *a*| for a subset-SUM/MAX query when entries of d are in [*a*,*b*].

Let *d* and *d'* differ in position i.

$a \leq d(i), d'(i) \leq b$

$q(d) - q(d') \leq d(i) - d'(i) \leq b - a$

Sensitivity of a Query – S(q)

[Dwork et al., TCC 2006]

Smallest number s.t. for any d, d' differing in one entry,

$$|| q(d) - q(d') || \leq S(q)$$

Example 2: **HISTOGRAM queries**

- Suppose each entry in d takes values in $\{c_1, c_2, \ldots, c_n\}$.
- Histogram(d) = $\{m_1, \ldots, m_n\}$, where m_i = (# entries in d with value c_i)
- S(q) = 2 for Histogram(d).

Changing one entry in d from c_i to c_j

- reduces the count of m_i by 1, and
- increases the count of m_j by 1.

Laplacian noise and Differential Privacy

Theorem: Adding noise drawn from a laplacian guarantees ε-differential privacy if,

$$\lambda \geq S(q)/\varepsilon.$$

- Subset-AGG queries:
  Return q(d) + η ,
  η sampled from Lap((b-a)/ε)
- Histogram queries:
  Return $\{m_1 + \eta_1, m_2 + \eta_2, \ldots, m_n + \eta_n\}$,
  η_i sampled **i.i.d.** from Lap(2/ε)

Proof of Differential Privacy

- Let {$\mathbf{x_1}$, x_2, …, x_n} & {$\mathbf{y_1}$, x_2, …, x_n} be 2 inputs.
- Let q be a query with sensitivity S(q)
  - $q(x_1, x_2, \ldots, x_n) = \{o_1, o_2, \ldots, o_k\}$ & $q(y_1, x_2, \ldots, x_n) = \{p_1, p_2, \ldots, p_k\}$.
  - **$\sum |oi - pi| \le S(q)$**
- Perturbed output for $q(x_1, x_2, \ldots, x_n)$:
  $\{\tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_k\} = \{o_1 + \eta_1, o_2 + \eta_2, \ldots, o_k + \eta_k\}$,
  η_i sampled **i.i.d.** from $Lap(S(q)/\varepsilon)$

Proof of Differential Privacy

- Let q be a query with sensitivity S(q)
  - $q(x_1, x_2, \ldots, x_n) = \{o_1, o_2, \ldots, o_k\}$ & $q(y_1, x_2, \ldots, x_n) = \{p_1, p_2, \ldots, p_k\}$.
  - $\sum |oi - pi| \le S(q)$
- Perturbed output for $q(x_1, x_2, \ldots, x_n)$:
  $\{\tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_k\} = \{o_1+\eta_1, o_2+\eta_2, \ldots, o_k+\eta_3\}$, η_i sampled **i.i.d.** from $Lap(S(q)/\varepsilon)$

$$\log\left(\frac{Pr[q(\mathbf{x_1}, x_2, \ldots, x_n) = \{\tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_k\}]}{Pr[q(\mathbf{y_1}, x_2, \ldots, x_n) = \{\tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_k\}]}\right)$$

$$= \log\left(\frac{Pr[\ldots, \eta_i = o_i - \tilde{o}_i, \ldots]}{Pr[\ldots, \eta_i = p_i - \tilde{o}_i, \ldots]}\right)$$

Proof of Differential Privacy

- $\sum |oi - pi| \le S(q)$
- **each η_i sampled i.i.d. from $Lap(\lambda)$,** $\lambda = S(q)/\varepsilon$

$$\log\left(\frac{Pr[\ldots, \eta_i = o_i - \tilde{o}_i, \ldots]}{Pr[\ldots, \eta_i = p_i - \tilde{o}_i, \ldots]}\right) = \log\left(\frac{\prod_i \exp(-|o_i - \tilde{o}_i|/\lambda)}{\prod_i \exp(-|p_i - \tilde{o}_i|/\lambda)}\right)$$

$$= \sum_i |p_i - \tilde{o}_i|/\lambda \; - \; \sum_i |o_i - \tilde{o}_i|/\lambda$$

$$\le \sum_i |o_i - p_i| / \lambda$$

Proof of Differential Privacy

- **$\sum |oi - pi| \leq S(q)$**
- each η_i sampled i.i.d. from $Lap(\lambda)$, $\lambda = S(q)/\varepsilon$

$$\log\left(\frac{Pr[\ldots, \eta_i = o_i - \tilde{o}_i, \ldots]}{Pr[\ldots, \eta_i = p_i - \tilde{o}_i, \ldots]}\right) = \log\left(\frac{\prod_i \exp(-|o_i - \tilde{o}_i|/\lambda)}{\prod_i \exp(-|p_i - \tilde{o}_i|/\lambda)}\right)$$

$$\leq \sum_i |o_i - p_i| / \lambda \quad \leq \quad S(q) / \lambda$$

Proof of Differential Privacy

- $\sum |oi - pi| \leq S(q)$
- each η_i sampled i.i.d. from $Lap(\lambda)$, **$\lambda = S(q)/\varepsilon$**

$$\log\left(\frac{Pr[\ldots, \eta_i = o_i - \tilde{o}_i, \ldots]}{Pr[\ldots, \eta_i = p_i - \tilde{o}_i, \ldots]}\right) = \log\left(\frac{\prod_i \exp(-|o_i - \tilde{o}_i|/\lambda)}{\prod_i \exp(-|p_i - \tilde{o}_i|/\lambda)}\right)$$

$$\leq \sum_i |o_i - p_i| / \lambda \quad \leq \quad S(q) / \lambda$$

$$\leq \varepsilon$$

Differential Privacy & Multiple Releases

Theorem **(Composability)**:

If *k* queries $q_1, q_2, \ldots, q_k$ are answered, s.t., each q_i satisfies $\boldsymbol{\varepsilon_i}$-differential privacy, resp.

Then, publishing all the answers together satisfies differential privacy with

$$\boldsymbol{\varepsilon = \varepsilon_1 + \varepsilon_2 + \ldots + \varepsilon_k}$$

Summary of Laplacian noise addition

- Guarantees privacy against strong adversaries.
- Good for queries with low sensitivities
  - Subset-AGG (with small domain sizes)
  - Histograms
- Data publishers only needs to:
  - Choose ε.
  - Know how to compute S(q).

Queries with Large Sensitivity

- Median, MAX, MIN …
- Let $\{x_1, \ldots, x_{10}\}$ be numbers in $[0, \Lambda]$. (assume x_i are sorted)
- $q_{med}(x_1, \ldots, x_{10}) = x_5$

Sensitivity of q_{med} = Λ

- $d_1 = \{0, 0, 0, 0, 0, \Lambda, \Lambda, \Lambda, \Lambda, \Lambda\}$ – $q_{med}(d_1) = 0$
- $d_2 = \{0, 0, 0, 0, \Lambda, \Lambda, \Lambda, \Lambda, \Lambda, \Lambda\}$ – $q_{med}(d_2) = \Lambda$

Queries with Large Sensitivity

However for most inputs q_{med} is not very sensitive.

d x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}

d' Λ x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 0

$$x_4 \le q_{med}(d') \le x_6$$

$= \max(x_5 - x_4, x_6 - x_5)$ **<< Λ**

d' differs from d in k=1 entry

Local Sensitivity of q at d – $LS_q(d)$

[Nissim et al., STOC 2007]

Smallest number s.t. for any d' differing in one entry from d,

$$|| q(d) - q(d') || \leq LS_q(d)$$

Sensitivity = Global sensitivity

$S(q) = \max_d LS_q(d)$

Can we add noise proportional to local sensitivity?

~~Noise proportional to Local Sensitivity~~

- $d_1 = \{0, 0, 0, 0, 0, 0, \Lambda, \Lambda, \Lambda, \Lambda\}$
  $q_{med}(d_1) = 0$
  $LS_{qmed}(d_1) = 0$ => Noise sampled from
- $d_2 = \{0, 0, 0, 0, 0, \Lambda, \Lambda, \Lambda, \Lambda, \Lambda\}$
  $q_{med}(d_2) = 0$
  $LS_{qmed}(d_2) = \Lambda$ => Noise sampled from $Lap(\Lambda/\varepsilon)$

differ in one value

$$\frac{\Pr[\text{answer} > 0 \mid d_1]}{\Pr[\text{answer} > 0 \mid d_2]} = \infty$$

$LS_{qmed}(d_1) = 0$ & $LS_{qmed}(d_2) = \Lambda$ implies $S(LS_q(.)) \geq \Lambda$

$LS_q(d)$ has very high sensitivity.

Smooth Sensitivity

[Nissim et al., STOC 2007]

S(.) is a **β-smooth upper bound** on the local sensitivity if,

For all d, $S_q(d) \geq LS_q(d)$

For all d, d' differing in one entry, $S_q(d) \leq \exp(\beta)\, S_q(d')$

- The smallest upper bound is called **β-smooth sensitivity**.

$$S^*_q(d) = \max_{d'} (\, LS_q(d') \exp(-m\beta)\,)$$

where d and d' differ in **m** entries.

Smooth sensitivity of q_{med}

d	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
d'	Λ	Λ	Λ	x_4	x_5	x_6	x_7	0	0	0

- $x_{5-k} \leq q_{med}(d') \leq x_{5+k}$
- $LS(d') = \max(x_{med+1} - x_{med}, x_{med} - x_{med-1})$

$$S^*_{qmed}(d) = \max_k (\exp(-k\beta) \times \max_{5-k \leq med \leq 5+k}(x_{med+1} - x_{med}, x_{med} - x_{med-1}))$$

Smooth sensitivity of q_{med}

For instance, Λ = 1000, β = 2.

d	1	2	3	4	5	6	7	8	9	10

$$S^*_{qmed}(d) = \max(\, \max_{0\leq k\leq 4}(\exp(-\beta\cdot k)\cdot 1),\ \max_{5\leq k\leq 10}(\exp(-\beta\cdot k)\cdot \Lambda)\,) = 1$$

Calibrating Noise to Smooth Sensitivity

$$A(d) = q(d) + Z \cdot (S^*_q(x) / \alpha)$$

- **Z** sampled from **h(z)** $\propto$ **$1/(1 + |z|^\gamma)$, $\gamma > 1$**
- **$\alpha = \varepsilon/4\gamma$,**
- S* is **ε/γ smooth sensitive**

Summary of Smooth Sensitivity

- Many functions have large global sensitivity.
- Local sensitivity captures sensitivity of current instance.
  - Local sensitivity is very sensitive.
  - Adding noise proportional to local sensitivity causes privacy breaches.
- Smooth sensitivity
  - Not sensitive.
  - Much smaller than global sensitivity.

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
  - Weak adversaries
  - Strong adversaries
    - Differential Privacy
    - Algorithms satisfying Differential Privacy
  - Bridging the Gap
- A Success Story: OnTheMap

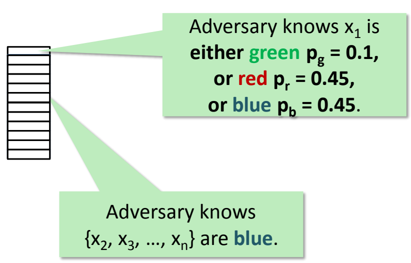
Search for the right privacy definition …

[Machanavajjhala et al. arxiv 2009]

- L-diversity, T-closeness, etc.,
  - Make restrictive assumptions about the adversary
  - Weak privacy definition
- Differential privacy
  - Makes very few assumptions about the adversary
  - Guards against very powerful adversaries

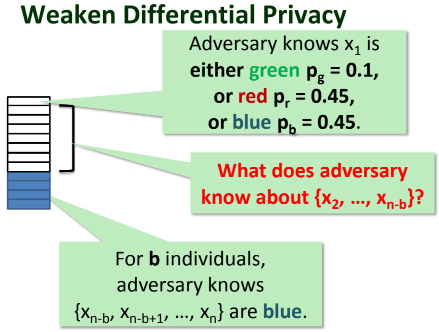
How to define privacy for the space in between?

Differential Privacy

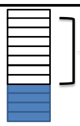


blue, green and red are three possibilities for each x_i.

Weaken Differential Privacy



blue, green and red are three possibilities for each x_i.



What does adversary know about $\{x_2, \ldots, x_{n-b}\}$?

Independent Entries: $\{x_1, x_2, \ldots, x_{n-b}\}$ are drawn **independently** from a single prob. vector $\{p_g, p_r, p_b\}$.

Independent Entries Privacy definition:

For every function f: dom(x_i) → {0,1},

$\Pr[f(x_i) = 1 \mid$ prior on x_i, $\{x_{n-b+1}, \ldots, x_n\}$ and $A(D - x_i)]$

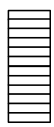
should be close to

$\Pr[f(x_i) = 1 \mid$ prior on x_i, $\{x_{n-b+1}, \ldots, x_n\}$ and $A(D)]$

Independent Entries Privacy Definition

- Suppose **b = 0**.
- $A(D) = \{m_{green} = 8, m_{red} = 2, m_{blue} = 2\}$

Publish a histogram without perturbation.



Independent Entries Privacy Definition

- Suppose **b = 0**.
- $A(D) = \{m_{green} = 8, m_{red} = 2, m_{blue} = 2\}$

$f(x_i) = 1$ iff x_i = green

$\Pr[f(x_i) = 1 \mid$ prior on x_i and $A(D)] = 8/12$

$\Pr[f(x_i) = 1 \mid$ prior on $x_i]$ $= p_g = 0.1$

due to the independence assumption

Independent Entries Privacy Definition

Independent Entries: $\{x_1, x_2, \ldots, x_{n-b}\}$ are drawn **independently** from a single prob. vector $\{p_g, p_r, p_b\}$.

Independent Entries Privacy definition:
For every function f: dom(x_i) → {0,1},

Pr[f(x_i) = 1 | prior on x_i, $\{x_{n-b+1}, \ldots, x_n\}$]
should be close to
Pr[f(x_i) = 1 | prior on x_i, $\{x_{n-b+1}, \ldots, x_n\}$ and A(D)]

But, Entries are Inherently Correlated …

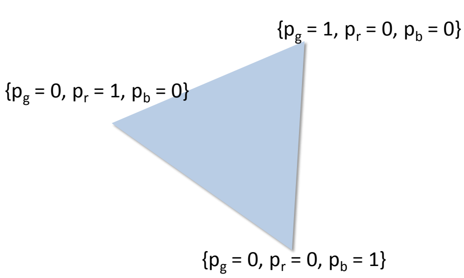
- Suppose **b = 0**.
- A(D – x_i) = $\{m_{green} = 7, m_{red} = 2, m_{blue} = 2\}$
  or $\{m_{green} = 8, m_{red} = 1, m_{blue} = 2\}$
  or $\{m_{green} = 8, m_{red} = 2, m_{blue} = 1\}$

Pr[f(x_i) = 1 | prior on x_i and A(D – x_i)]
is closer to 8/11 rather than 0.1 is D is sufficiently large.

Adversaries learn on seeing new data.

Modeling Adversaries who Learn

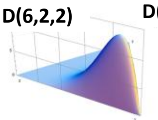
- Use Dirichlet D(α_g, α_r, α_b).
  - Defines a probability distribution over various $\{p_g, p_r, p_b\}$.



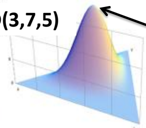
Modeling Adversaries who Learn

- Use Dirichlet D(α_g, α_r, α_b).
  - Maximum probability given to {p^*_g, p^*_r, p^*_b}, where $p^*_g = \alpha_g / (\alpha_g + \alpha_r + \alpha_b)$

D(6,2,2)

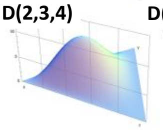


D(3,7,5)

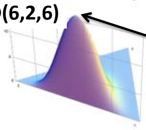


{p^*_g=0.2, p_r=0.47, p_b = 0.53}

D(2,3,4)



D(6,2,6)

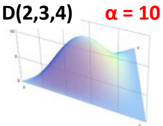


{p^*_g=0.43, p_r=0.14, p_b = 0.43}

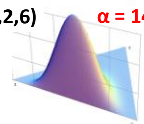
Modeling Adversaries who Learn

- Use Dirichlet D(α_g, α_r, α_b).
  - Call $\alpha = (\alpha_g + \alpha_r + \alpha_b)$ the **stubbornness** of the prior.
  - As α increases, more probability is given to {p^*_g, p^*_r, p^*_b}.

D(2,3,4) $\alpha = 10$



D(6,2,6) $\alpha = 14$



Modeling Adversaries who Learn

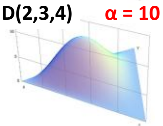
- **Think as follows …**

  Adversary is forming prior from external data.

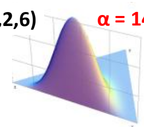
  More data seen => more certain about the prior distribution.

  α measures amount data seen/certainty.

D(2,3,4) $\alpha = 10$



D(6,2,6) $\alpha = 14$



Modeling Adversaries who Learn

- Use Dirichlet $D(\alpha_g, \alpha_r, \alpha_b)$.
  - Call $\alpha = (\alpha_g + \alpha_r + \alpha_b)$ the **stubbornness** of the prior.
  - As α increases, more probability is given to $\{p^*_g, p^*_r, p^*_b\}$.
  - When $\alpha \rightarrow \infty$, $\{p^*_g, p^*_r, p^*_b\}$ has probability 1 and **we get the independence assumption.**

Modeling Adversaries who Learn

- Use Dirichlet $D(\alpha_g, \alpha_r, \alpha_b)$.
  - Suppose b = n-1 (like in differential privacy).
  - A single entry sampled from $D(\alpha_g, \alpha_r, \alpha_b)$ is mathematically equivalent to an entry sampled from $\{p^*_g, p^*_r, p^*_b\}$.
    - Because there are no more correlations across entries.
  - **When adversary knows all but one entries, Dirichlet distribution degenerates to a point distribution.**

Modeling Adversaries who Learn

- Use Dirichlet $D(\alpha_g, \alpha_r, \alpha_b)$.
  - When $\alpha \rightarrow \infty$, $\{p^*_g, p^*_r, p^*_b\}$ has probability 1 and **we get the independence assumption.**

UNREASONABLE: infinite stubbornness => infinite prior data seen.

  - **When adversaries knows all but one entries, Dirichlet distribution degenerates to a point distribution.**

UNREASONABLE: adversary usually does not know all but one values.

Modeling Adversaries who Learn

- Use Dirichlet D(α_g, α_r, α_b).
  - When **α → ∞**, {p^*_g, p^*_r, p^*_b} has probability 1 and **we get the independence assumption.**

L-diversity, T-closeness, Personalized privacy, δ-disclosure.

  - **When adversaries knows all but one entries, Dirichlet distribution degenerates to a point distribution.**

UNREASONABLE: adversary usually does not know all but one values.

Modeling Adversaries who Learn

- Use Dirichlet D(α_g, α_r, α_b).
  - When **α → ∞**, {p^*_g, p^*_r, p^*_b} has probability 1 and **we get the independence assumption.**

L-diversity, T-closeness, Personalized privacy, δ-disclosure.

  - **When adversaries knows all but one entries, Dirichlet distribution degenerates to a point distribution.**

Differential Privacy.

ε-Privacy

For every function f: dom(x_i) → {0,1},

Pr[f(x_i) = 1 | Dirichlet prior on x_i, {x_{n-b+1}, …, x_n} and A(D – x_i)]

should be close to

Pr[f(x_i) = 1 | Dirichlet prior on x_i, {x_{n-b+1}, …, x_n} and A(D)]

Use Guassian, Pareto or Poisson for numeric valued attributes.

ε-Privacy: Adversary Classes

- Class I: **Fixed α_g, α_r, α_b**
  A single adversary with a fixed prior $D(\alpha_g, \alpha_r, \alpha_b)$.
- Class II: **Variable α_g, α_r, α_b, but Fixed α**.
  Any adversary with $D(\alpha_g, \alpha_r, \alpha_b)$ such that
  $\alpha_g, \alpha_r, \alpha_b$ add up to α.
- Class III: **Fixed α_g/α, α_r/α, α_b/α , but Variable α**.
- Class IV: **Variable α_g, α_r, α_b and Variable α**.

ε-Privacy and Generalizations

- K-anonymity:
  … each group has at least k tuples …
- L-diversity:
  … most frequent sensitive value appears in at most c/(c+1) fraction …
- ε-Privacy-Class II:
  … each group has at least α/ε-1 tuples …
  … and, the most frequent sensitive value appears in at most 1 – 1/(ε+δ) fraction …
  … δ depends on how large the group is.

ε-Privacy Summary

- One way to define privacy in between weaker definitions like L-diversity etc., and strong definitions like differential privacy.
- Key challenge is modelling the adversary's prior knowledge about the individuals in the table.
  - Both independence and knowledge of all but one entries in the table are unreasonable.
- ε-Privacy allows deterministic anonymization algorithms.

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
- A Success Story: OnTheMap

Privacy in the real world

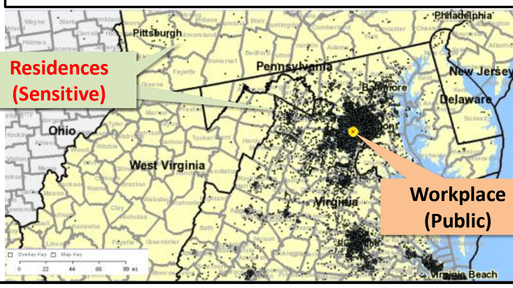
- OnTheMap: A real census application.
  - Synthetically generated data published for economic research.
  - Privacy implications were poorly understood.
- Walk through privacy analysis of this application.
  - Challenge 1 (routine): New statistical algorithms for data publishing.
    - Derived conditions under which published data is private.
  - Challenge 2 (not routine): Data is very sparse.
    - No existing tools that enhance utility in the face of data sparsity.

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
- A Success Story: OnTheMap
  - OnTheMap and existing synthetic data generation algorithms.
  - Privacy analysis
    - Privacy but no utility!
  - Publishing usable synthetic data with privacy guarantees.

OnTheMap: A Census application that plots commuting patterns of workers

http://lehdmap3.did.census.gov/



OnTheMap: A Census application that plots commuting patterns of workers

Residence (Sensitive)

Workplace (Public)

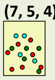
Worker ID	Origin	Destination
1223	MD11511	DC22122
1332	MD2123	DC22122
1432	VA11211	DC22122
2345	PA12121	DC24132
1432	PA11122	DC24132
1665	MD1121	DC24132
1244	DC22122	DC22122

Census Blocks

A Synthetic Data Generator (Dirichlet resampling)

Step 1: **Noise Addition** *(for each destination)*

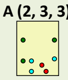
D (7, 5, 4)



Multi-set of Origins for workers in Washington DC.

+

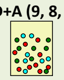
A (2, 3, 3)



Noise (fake workers)

=

D+A (9, 8, 7)

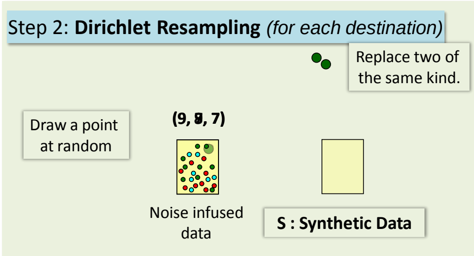


Noise infused data

● **Washington DC** ● **Somerset** ○ **Fuller**

Noise added to an origin with at least 1 worker is > 0

A Synthetic Data Generator (Dirichlet resampling)



frequency of block *b* in D+A = 0 ➔ frequency of *b* in S = 0
i.e., block *b* is ignored by the algorithm.

How should we add noise (fake workers)?

- Intuitively, more noise yields more privacy …

1. How much noise (fake workers) should we add?
2. To which blocks should we add noise (fake workers)?

- This was poorly understood.
  - **Total amount of noise added is a state secret**
  - **Only 3-4 people in the US know this value in the current implementation of OnTheMap.**

Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
- A Success Story: OnTheMap
  - OnTheMap and existing synthetic data generation algorithms.
  - Privacy analysis
    - Choosing a privacy definition (Differential Privacy).
    - Deriving conditions for privacy.
    - Privacy but no utility!
  - Publishing usable synthetic data with privacy guarantees.

Privacy requirements for OnTheMap

- The link between an individual and a (group of) residence block(s) is the sensitive information.
- Adversarial background knowledge:
  - *Alice knows co-worker Bob has the longest commute time.*
  - *Alice can deduce Bob comes from a small region on the map.*
  - *Alice also knows no other individual comes from that region.*
- Privacy metric, or "when is privacy breached"?
  - Pr[*"Bob resides around DC22122" | T*, adv. knowledge*] differs from *adversary's prior knowledge.*

Privacy of Synthetic Data

Theorem 1:

The Dirichlet resampling algorithm preserves ε-differential privacy if and only if for every destination *d*, the noise added to each block is at least

$$\frac{m(d)}{\varepsilon - 1}$$

where *m(d)* is the size of the synthetic population for destination *d* and *ε* is the privacy parameter.

1. How much noise should we add?

Noise required per block: (differential privacy)

lesser privacy →

Privacy (ε =)	5	10	20	50
Noise per block (x 10^6)	0.25	0.11	0.05	0.02

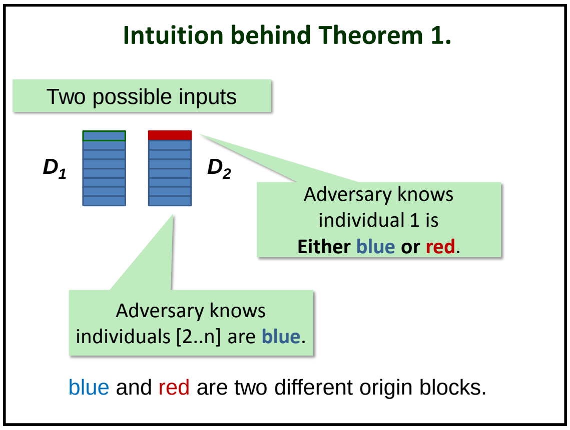
Input: 1 million original workers.
Output: 1 million synthetic workers.

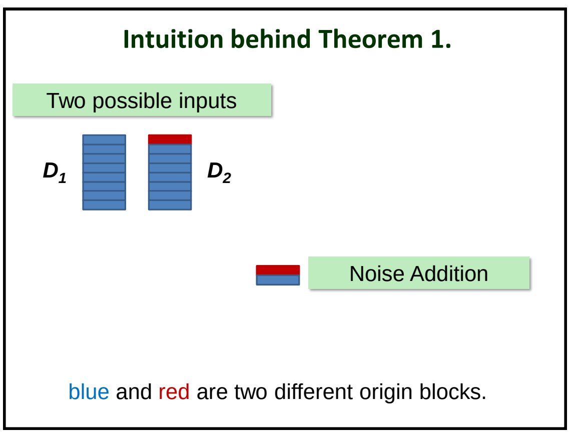
2. To which blocks should we add noise?

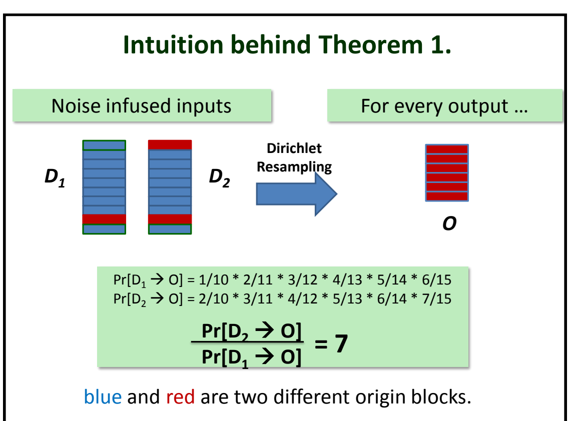
Add noise to every block on the map.

There are 8 million Census blocks on the map!

1 million original workers and 160 billion fake workers!!!


Intuition behind Theorem 1.
Two possible inputs
D1
D2
Adversary knows individual 1 is Either blue or red.
Adversary knows individuals [2..n] are blue.
blue and red are two different origin blocks.

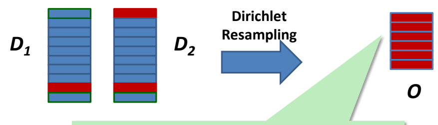

Intuition behind Theorem 1.
Two possible inputs
D1
D2
Noise Addition
blue and red are two different origin blocks.


Intuition behind Theorem 1.
Noise infused inputs
For every output …
D1
D2
Dirichlet Resampling
O
Pr[D1 → O] = 1/10 * 2/11 * 3/12 * 4/13 * 5/14 * 6/15
Pr[D2 → O] = 2/10 * 3/11 * 4/12 * 5/13 * 6/14 * 7/15
Pr[D2 → O] / Pr[D1 → O] = 7
blue and red are two different origin blocks.

Intuition behind Theorem 1.

Noise infused inputs

For every output …



Adversary infers that it is very likely individual 1 is red …
… unless noise added is very large.

blue and red are two different origin blocks.

Privacy Analysis of OnTheMap: Summary

- We chose differential privacy …
  - Guards against powerful adversaries.
  - Measures privacy as a distance between prior and posterior.
- … but synthetic data that satisfies differential privacy is useless!

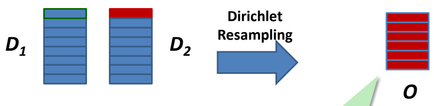
Tutorial Outline

- Untrusted Data Collector
- Trusted Data Collector
- A Success Story: OnTheMap
  - OnTheMap and existing synthetic data generation algorithms.
  - Privacy analysis
  - Publishing usable synthetic data with privacy guarantees.

But, breach occurs with very low probability.

Noise infused inputs

For every output …



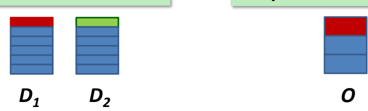
Probability of O ≈ 10^{-4}

blue and red are two different origin blocks.

Probabilistic Differential Privacy

For every pair of inputs that differ in one value

For every ***probable*** output



Adversary may distinguish between D_1 and D_2 based on a set of unlikely outputs with probability at most δ

$$\Pr\left[O \;\middle|\; \frac{\Pr[D_1 \rightarrow O]}{\Pr[D_2 \rightarrow O]} < \varepsilon\right] > 1 - \delta$$

1. How much noise should we add?

Noise required per block:

lesser privacy →

Privacy (ε =)	5	10	20	50	
Noise per block	25×10^4	11×10^4	5×10^4	2×10^4	← Differential Privacy
Noise per block	17.5	5.5	2.16	0.74	← Probabilistic Differential Privacy ($\delta = 10^{-5}$)

Input: 1 million original workers.
Output: 1 million synthetic workers.

1. How much noise should we add?

Noise required per block:

lesser privacy →

Privacy (ε =)	5	10	20	50	
Noise per block	$25x10^4$	$11x10^4$	$5x10^4$	$2x10^4$	← Differential Privacy
Noise per block	17.5	5.5	2.16	0.74	← Probabilistic Differential Privacy ($\delta = 10^{-5}$)

Input: 1 million original workers.
Output: 1 million synthetic workers.

2. To which blocks should we add noise?

Why not add noise to every block?

Why not add noise to every block?

Noise required per block: (probabilistic differential privacy)

1 million original and synthetic workers.

lesser privacy →

Privacy (ε =)	5	10	20	50
Noise per block	17.5	5.5	2.16	0.74

- There are about **8 million** blocks on the map!
  - Total noise added is about **6 million.**
- Causes non-trivial spurious commute patterns.
  - Roughly 1 million fake workers from West Coast (out of a total 7 million points in the noise infused data).
  - Hence, 1/7 of the synthetic data have residences in West Coast and work in Washington DC.

2. To which blocks should we add noise?

Noise required per block: (probabilistic differential privacy)

1 million original and synthetic workers.

lesser privacy →

Privacy (ε =)	5	10	20	50
Noise per block	17.5	5.5	2.16	0.74

Adding noise to all blocks creates spurious commute patterns.

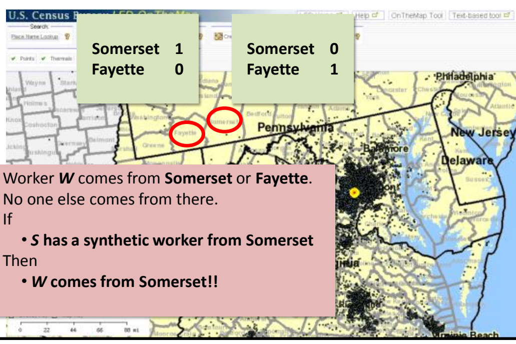
Why not add noise only to blocks that appear in the original data?

Theorem 2: Adding noise only to blocks that appear in the data breaches privacy.

If a **block *b* does not appear** in the original data and **no noise** is added to *b*

then ***b* cannot appear** in the synthetic data.

Theorem 2: Adding noise only to blocks that appear in the data breaches privacy.

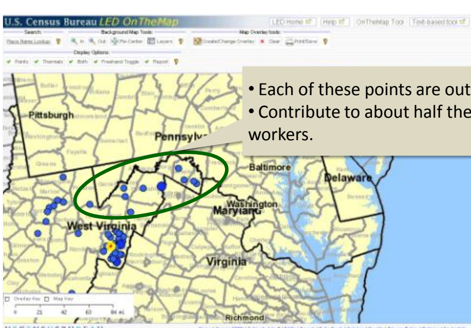


Somerset	1
Fayette	0

Somerset	0
Fayette	1

- Worker ***W*** comes from **Somerset** or **Fayette**.
- No one else comes from there.
- If
  - ***S* has a synthetic worker from Somerset**
- Then
  - ***W* comes from Somerset!!**

Ignoring outliers degrades utility



- Each of these points are outliers.
- Contribute to about half the workers.

Our solution to “Where to add noise?”

Step 1 : **Coarsen the domain**

- Based on an existing public dataset (Census Transportation Planning Package, CTPP).

Our solution to “Where to add noise?”

Step 1 : **Coarsen the domain**

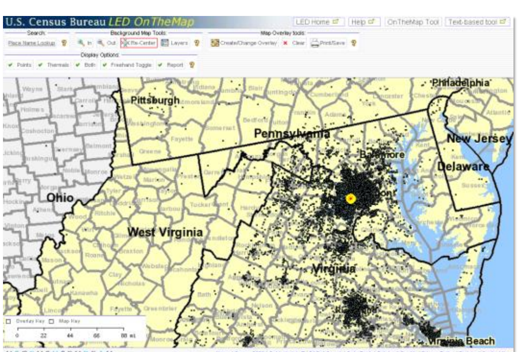
Step 2: **Probabilistically drop blocks that do not appear.**

- Pick a function $f: \{b_1, \ldots, b_k\} \rightarrow (0,1]$ (based on external data)
- For every block b that do not appear, *ignore b with probability f(b)*

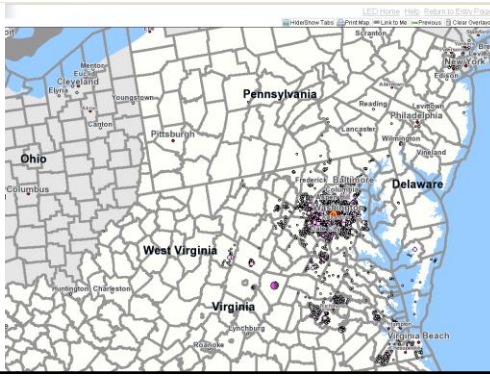
Theorem 3:

Parameter ε increases by $\max_b (\max (2^{\text{noise per block}}, f(b)))$

OnTheMap version 2



OnTheMap version 3



OnTheMap: Summary

- OnTheMap: A real census application.
  - Synthetically generated data published for economic research.
  - Currently, privacy implications are poorly understood.
    - Parameters to the algorithm are **state secret**.
- Walked through privacy analysis of this application.
  - Analyzed the privacy of OnTheMap using Differential Privacy.
  - How to publish useful information despite sparse data.
- Provably private algorithms are currently being used.

Tutorial Summary 1

Tutorial Summary 2

Backup slides

Dinur-Nissim negative results

Negative Result

Theorem 1 *(Exponential Adversaries)*:
If a database answers subset-SUM queries with an additive noise of $\varepsilon = o(n)$, then an adversary can recover 99% of the database by issuing exp(n) queries.

Theorem 2 *(Polynomially-bounded Adversaries)*:
If a database answers subset-SUM queries with an additive noise of $\varepsilon = o(\sqrt{n})$, then an adversary can recover 99% of the database using poly(n) queries.

Proof of Theorem 1

Let *A* be within $\varepsilon = o(n)$ perturbation on database $d \in \{0,1\}^n$.

[Query Phase]
For all queries *q*: let $\tilde{a}_q = A(d,q) \leq q(d) + \varepsilon$

[Weeding Phase]
Output database *c*,
if $|q(c) - \tilde{a}_q| \leq \varepsilon$ for all queries q

Claim: *d* and *c* differ by atmost $4\varepsilon = o(n)$.

Claim: *d* and *c* differ by atmost $4\varepsilon = o(n)$

Suppose d and c differ by $> 4\varepsilon$.

d

Id	value
1	0
2	0
3	1
4	1
5	0
6	0
7	1

c

Id	value
1	0
2	1
3	1
4	0
5	0
6	1
7	1

q_1 = sum(2, 6) (d has 0, c has 1)

q_2 = sum(4) (d has 1, c has 0)

Then,
$q1(c) – q1(d) > 2\varepsilon$ OR $q1(c) – q1(d) > 2\varepsilon$

But,
$\tilde{a}_q = A(d,q_1) \leq q_1(d) + \varepsilon$

Hence,
$|q(c) - \tilde{a}_q| > \varepsilon$ – contradiction

Feasibility Result

Theorem:

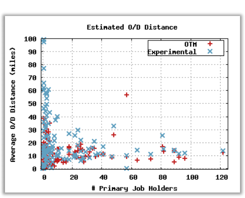
There exist output perturbation algorithms within O(√T(n)) perturbation that guarantee privacy against adversaries who ask at most T(n) queries.

- Algorithms satisfies differential privacy.

On The Map Utility

Utility of the provably private algorithm

Utility measured by average commute distance for each destination block.

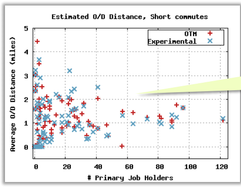


Experimental Setup:

- **OTM**: Currently published OnTheMap data used as original data.
- All destinations in Minnesota.
- 120, 690 origins per destination.
  - chosen by pruning out blocks that are > 100 miles from the destination.
- $\varepsilon = 100$, $\delta = 10^{-5}$
- Additional leakage due to probabilistic pruning = 4 *(min f(b) = 0.0378)*

Utility of the provably private algorithm

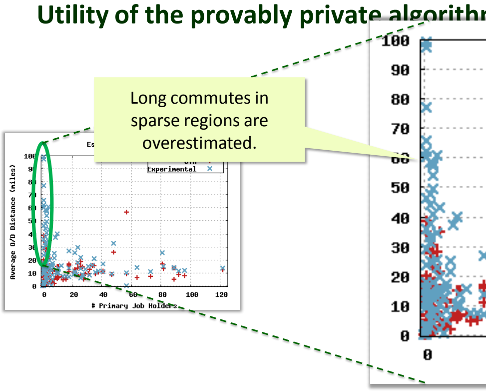
Utility measured by average commute distance for each destination block.



Short commutes have low error in both sparse and dense regions.

Utility of the provably private algorithm

Long commutes in sparse regions are overestimated.



Questions?

johannes@cs.cornell.edu
mvnak@yahoo-inc.com