Building Blocks of Privacy: Differentially Private Mechanisms

Graham Cormode
graham@cormode.org
The data release scenario
Data Release

♦ Much interest in private data release
  – Practical: release of AOL, Netflix data etc.
  – Research: hundreds of papers

♦ In practice, many data-driven concerns arise:
  – How to design algorithms with a meaningful privacy guarantee?
  – Trading off noise for privacy against the utility of the output?
  – Efficiency / practicality of algorithms as data scales?
  – How to interpret privacy guarantees?
  – Handling of common data features, e.g. sparsity?

♦ This talk: describe some tools to address these issues
Differential Privacy

- **Principle**: released info reveals little about any individual
  - Even if adversary knows (almost) everything about everyone else!
- Thus, individuals should be secure about contributing their data
  - What is learnt about them is about the same either way
- Much work on providing differential privacy (DP)
  - Simple recipe for some data types e.g. numeric answers
  - Simple rules allow us to reason about composition of results
  - More complex algorithms for arbitrary data (many DP mechanisms)
- Adopted and used by several organizations:
  - US Census, Common Data Project, Facebook (?)
Differential Privacy Definition

The output distribution of a differentially private algorithm changes very little whether or not any individual’s data is included in the input – so you should contribute your data.

A randomized algorithm $K$ satisfies $\varepsilon$-differential privacy if:
Given any pair of neighboring data sets, $D$ and $D'$, and $S$ in $\text{Range}(K)$:

$$\Pr[K(D) = S] \leq e^\varepsilon \Pr[K(D') = S]$$

Neighboring datasets differ in one individual: we say $|D-D'|=1$.
Achieving Differential Privacy

- Suppose we want to output the number of left-handed people in our data set
  - Can reduce the description of the data to just the answer, \( n \)
  - Want a randomized algorithm \( K(n) \) that will output an integer
  - Consider the distribution \( \Pr[K(n) = m] \) for different \( m \)

- Write \( \exp(\varepsilon) = \alpha \), and \( \Pr[K(n) = n] = p_n \). Then:
  \[
  \Pr[K(n) = n-1] \leq \alpha \Pr[K(n-1) = n-1] = \alpha \ p_{n-1}
  \]
  \[
  \Pr[K(n) = n-2] \leq \alpha \Pr[K(n-1) = n-2] \leq \alpha^2 \Pr[K(n-2) = n-2] = \alpha^2 \ p_{n-2}
  \]
  \[
  \Pr[K(n) = n-i] \leq \alpha^i \ p_{n-i}
  \]

Similarly, \( \Pr[K(n) = n+i] \leq \alpha^i \ p_{n+i} \)
Achieving Differential Privacy

- We have $\Pr[K(n) = n-i] \leq \alpha^i p_{n-i}$ and $\Pr[K(n) = n+i] \leq \alpha^i p_{n+i}$
- Within these constraints, we want to maximize $p_n$
  - This maximizes the probability of returning “correct” answer
  - Means we turn the inequalities into equalities
- For simplicity, set $p_n = p$ for all $n$
  - Means the distribution of “shifts” is the same whatever $n$ is
- Yields: $\Pr[K(n) = n-i] = \alpha^i p$ and $\Pr[K(n) = n+i] \leq \alpha^i p$
  - Sum over all shifts $i$:
    $p + \sum_{i=1}^{\infty} 2\alpha^i p = 1$
    $p + 2p \frac{\alpha}{(1-\alpha)} = 1$
    $p(1 - \alpha + 2\alpha)/(1-\alpha) = 1$
    $p = (1-\alpha)/(1+\alpha)$
Geometric Mechanism

♦ What does this mean?
  – For input $n$, output distribution is $\Pr[K(n) = m] = \alpha^{|m-n|} \cdot \frac{(1-\alpha)}{(1+\alpha)}$

♦ What does this look like?
  – Symmetric geometric distribution, centered around $n$
  – We draw from this distribution centered around zero, and add to the true answer
  – We get the “true answer plus (symmetric geometric) noise”

♦ A first differentially private mechanism for outputting a count
  – We call this “the geometric mechanism”
Truncated Geometric Mechanism

♦ Some practical concerns:
  – This mechanism could output any value, from $-\infty$ to $+\infty$

♦ Solution: we can “truncate” the output of the mechanism
  – E.g. decide we will never output any value below zero, or above $N$
  – Any value drawn below zero is “rounded up” to zero
  – Any value drawn above $N$ is “rounded down” to $N$
  – This does not affect the differential privacy properties
  – Can directly compute the closed-form probability of these outcomes
Laplace Mechanism

- Sometimes we want to output real values instead of integers
- The Laplace Mechanism naturally generalizes Geometric

- Add noise from a symmetric continuous distribution to true answer
- Laplace distribution is a symmetric exponential distribution
- Is DP for same reason as geometric: shifting the distribution changes the probability by at most a constant factor
- PDF: $\Pr[X = x] = \frac{1}{2\lambda} \exp(-|x|/\lambda)$
  
  Variance = $2\lambda^2$
Sensitivity of Numeric Functions

- For more complex functions, we need to calibrate the noise to the influence an individual can have on the output.
  - The (global) sensitivity of a function $F$ is the maximum (absolute) change over all possible adjacent inputs.
  - $S(F) = \max_{D, D'} : |D-D'| = 1 \quad |F(D) - F(D')| = 1$
  - Intuition: $S(F)$ characterizes the scale of the influence of one individual, and hence how much noise we must add.

- $S(F)$ is small for many common functions.
  - $S(F) = 1$ for COUNT
  - $S(F) = 2$ for HISTOGRAM
  - Bounded for other functions (MEAN, covariance matrix...)

Laplace Mechanism with Sensitivity

- Release $F(x) + \text{Lap}(S(F)/\varepsilon)$ to obtain $\varepsilon$-DP guarantee
  - $F(x) =$ true answer on input $x$
  - $\text{Lap}(\lambda) =$ noise sampled from Laplace dbn with parameter $\lambda$
  - Exercise: show this meets $\varepsilon$-differential privacy requirement

- Intuition on impact of parameters of differential privacy (DP):
  - Larger $S(F)$, more noise (need more noise to mask an individual)
  - Smaller $\varepsilon$, more noise (more noise increases privacy)
  - Expected magnitude of $|\text{Lap}(\lambda)|$ is (approx) $1/\lambda$
Sequential Composition

- What happens if we ask multiple questions about same data?
  - We reveal more, so the bound on $\varepsilon$ differential privacy weakens

- Suppose we output via $K_1$ and $K_2$ with $\varepsilon_1$, $\varepsilon_2$ differential privacy:

$$\Pr[K_1(D) = S_1] \leq \exp(\varepsilon_1) \Pr[K_1(D') = S_1],$$

$$\Pr[K_2(D) = S_2] \leq \exp(\varepsilon_2) \Pr[K_2(D') = S_2]$$

$$\Pr[(K_1(D) = S_1), (K_2(D) = S_2)] = \Pr[K_1(D)=S_1] \Pr[K_2(D) = S_2]$$

$$\leq \exp(\varepsilon_1) \Pr[K_1(D') = S_1] \exp(\varepsilon_2) \Pr[K_2(D') = S_2]$$

$$= \exp(\varepsilon_1 + \varepsilon_2) \Pr[(K_1(D') = S_1), (K_2(D') = S_2)]$$

- Use the fact that the noise distributions are independent

- **Bottom line**: result is $\varepsilon_1 + \varepsilon_2$ differentially private

  - Can reason about **sequential composition** by just “adding the $\varepsilon$’s”
Parallel Composition

♦ Sequential composition is pessimistic
  – Assumes outputs are correlated, so privacy budget is diminished
♦ If the inputs are disjoint, then result is $\max(\varepsilon_1, \varepsilon_2)$ private
♦ Example:
  – Ask for count of people broken down by handedness, hair color

<table>
<thead>
<tr>
<th></th>
<th>Redhead</th>
<th>Blond</th>
<th>Brunette</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-handed</td>
<td>23</td>
<td>35</td>
<td>56</td>
</tr>
<tr>
<td>Right-handed</td>
<td>215</td>
<td>360</td>
<td>493</td>
</tr>
</tbody>
</table>

– Each cell is a disjoint set of individuals
– So can release each cell with $\varepsilon$-differential privacy (parallel composition) instead of $6\varepsilon$ DP (sequential composition)
Exponential Mechanism

- What happens when we want to output non-numeric values?
- **Exponential mechanism** is most general approach
  - Captures all possible DP mechanisms
  - But ranges over all possible outputs, may not be efficient

**Requirements:**
- Input value $x$
- Set of possible outputs $O$
- Quality function, $q$, assigns “score” to possible outputs $o \in O$
  - $q(x, o)$ is bigger the “better” $o$ is for $x$
- Sensitivity of $q = S(q) = \max_{x,x',o} |q(x,o) - q(x',o)|$
Exponential Mechanism

- Sample output \( o \in O \) with probability
  \[
  \Pr[K(x) = o] = \frac{\exp(\varepsilon q(x,o))}{\left(\sum_{o' \in O} \exp(\varepsilon q(x,o'))\right)}
  \]

- Result is \((2\varepsilon S(q))\)-DP
  - Shown by considering change in numerator and denominator under change of \( x \) is at most a factor of \( \exp(\varepsilon S(q)) \)

- **Scalability**: need to be able to draw from this distribution

- **Generalizations**:
  - \( O \) can be continuous, \( \sum \) becomes an integral
  - Can apply a prior distribution over outputs as \( P(o) \)
    - We assume a uniform prior for simplicity
Exponential Mechanism Example 1: Count

- Suppose input is a count \( n \), we want to output (noisy) \( n \)
  - Outputs \( O = \) all integers
  - \( q(o,n) = -|o-n| \)
  - \( S(q) = 1 \)
  - Then \( \Pr[K(n) = o] = \exp(-\varepsilon |o-n|)/(\sum_o -\varepsilon |o-n|) = \alpha^{-|o-n|} \cdot (1-\alpha)/(1-\alpha) \)
  - Simplifies to the Geometric mechanism!

- Similarly, if \( O = \) all reals, applying exponential mechanism results in the Laplace Mechanism

- Illustrates the claim that Exponential Mechanism captures all possible DP mechanisms
Exponential Mechanism, Example 2: Median

- Let $M(X) =$ median of set of values in range $[0,T]$ (e.g. median age)
- Try Laplace Mechanism: $S(M) = T$
  - There can be datasets $X, X'$ where $M(X) = 0, M(X') = T, |X-X'|=1$
  - Consider $X = [0^n, 0, T^n], X' = [0^n, T, T^n]$
  - Noise from Laplace mechanism outweighs the true answer!
- Exponential Mechanism: set $q(o,X) = -|\text{rank}_X(o) - |X|/2|$
  - Define $\text{rank}_X(o)$ as the number of elements in $X$ dominated by $o$
  - Note, $\text{rank}_X(M(X)) = |X|/2$ : median has rank half
  - $S(q) = 1$: adding or removing an individual changes $q$ by at most 1
  - Then $\Pr[K(X) = o] = \exp(\epsilon q(o,X))/(\sum_{o' \in O} \exp(\epsilon q(o',X)))$
  - Problem: $O$ could be very large, how to make efficient?
Observation: for many values of $o$, $q(o, X)$ is the same:
- Index $X$ in sorted order so $x_1 \leq x_2 \leq x_3 \leq \ldots \leq x_n$
- Then for any $x_i \leq o < o' \leq x_{i+1}$, $\text{rank}_X(o) = \text{rank}_X(o')$
- Hence $q(o,X) = q(o',X)$

Break possible outputs into ranges:
- $O_0 = [0, x_1]$, $O_1 = [x_1, x_2]$, \ldots, $O_n = [x_n, T]$
- Pick range $O_j$ with probability proportional to $|O_j| \exp(\varepsilon q(O,X))$
- Pick output $o \in O_j$ uniformly from the range
- Time cost is proportional to number of ranges $n$ (after sorting $X$)

Similar tricks make exponential mechanism practical elsewhere
Recap

- Have developed a number of building blocks for DP:
  - Geometric and Laplace mechanism for numeric functions
  - Exponential mechanism for sampling from arbitrary sets
- And “cement” to glue things together:
  - Parallel and sequential composition theorems
- With these blocks and cement, can build a lot
  - Many papers arrive from careful combination of these tools!
- Useful fact: any post-processing of DP output remains DP
  - (so long as you don’t access the original data again)
  - Helps reason about privacy of data release processes
Case Study: Sparse Spatial Data

- Consider location data of many individuals
  - Some dense areas (towns and cities), some sparse (rural)
- Applying DP naively simply generates noise
  - Lay down a fine grid, signal overwhelmed by noise
- Instead: compact regions with sufficient number of points
Private Spatial decompositions

- **Build**: adapt existing methods to have differential privacy
- **Release**: a private description of data distribution (in the form of bounding boxes and noisy counts)
Building a Private kd-tree

- Process to build a private kd-tree
  - **Input**: maximum height \( h \), minimum leaf size \( L \), data set
  - Choose dimension to split
  - Get (private) median in this dimension
  - Create child nodes and add noise to the counts
  - Recurse until:
    - Max height is reached
    - Noisy count of this node less than \( L \)
    - Budget along the root-leaf path has used up
- The entire PSD satisfies DP by the composition property
Building PSDs – privacy budget allocation

- Data owner specifies a total budget $\varepsilon$ reflecting the level of anonymization desired
- Budget is split between medians and counts
  - Tradeoff accuracy of division with accuracy of counts
- Budget is split across levels of the tree
  - Privacy budget used along any root-leaf path should total $\varepsilon$
Privacy budget allocation

- How to set an $\varepsilon_i$ for each level?
  - Compute the number of nodes touched by a ‘typical’ query
  - Minimize variance of such queries
  - Optimization: $\min \sum_i 2^{h-i} / \varepsilon_i^2$ s.t. $\sum_i \varepsilon_i = \varepsilon$
  - Solved by $\varepsilon_i \propto (2^{(h-i)})^{1/3} \varepsilon$ : more to leaves
  - Total error (variance) goes as $2^h / \varepsilon^2$

- Tradeoff between noise error and spatial uncertainty
  - Reducing $h$ drops the noise error
  - But lower $h$ increases the size of leaves, more uncertainty
Post-processing of noisy counts

♦ Can do additional post-processing of the noisy counts
  – To improve query accuracy and achieve consistency
♦ Intuition: we have count estimate for a node and for its children
  – Combine these independent estimates to get better accuracy
  – Make consistent with some true set of leaf counts
♦ Formulate as a linear system in $n$ unknowns
  – Avoid explicitly solving the system
  – Expresses optimal estimate for node $v$ in terms of estimates of ancestors and noisy counts in subtree of $v$
  – Use the tree-structure to solve in three passes over the tree
  – Linear time to find optimal, consistent estimates
Data Transformations

♦ Can think of trees as a ‘data-dependent’ transform of input
♦ Can apply other data transformations
♦ General idea:
  – Apply transform of data
  – Add noise in the transformed space (based on sensitivity)
  – Publish noisy coefficients, or invert transform (post-processing)
♦ Goal: pick a transform that preserves good properties of data
  – And which has low sensitivity, so noise does not corrupt
Wavelet Transform

- **Haar wavelet transform** commonly used to approximate data
  - Any 1D range is expressed using $\log n$ coefficients
  - Each input point affects $\log n$ coefficients
  - Is a linear, orthonormal transform
- Can add noise to wavelet coefficients
  - Treat input as a 1D histogram of counts
  - **Bounded sensitivity**: each individual affects coefficients by $O(1)$
  - Can transform noisy coefficients back to get noisy histogram
- Range queries are answered well in this model
  - Each range query picks up noise (variance) $O(\log^3 n / \varepsilon)$
  - Directly adding noise to input would give noise $O(n / \varepsilon)$
Other Transforms

Many other transforms can be applied within DP

- **(Discrete) Fourier Transform**: also bounded sensitivity
  - Often need only a fixed set of coefficients: further reduces $S(F)$
  - Used for representing data cube counts, time series

- **Hierarchical Transforms**: binary trees and quadtrees

- **Randomized Transforms**: sketches and compressed sensing

\[
A_8 = \sqrt{\frac{1}{8}} \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 \\
1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\
0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]
Local Sensitivity

- **A common fallacy**: using local sensitivity instead of global
  - Global sensitivity \( S(F) = \max_{x,x'} : |x-x'|=1 |F(x)-F(x')| \)
  - Local sensitivity \( S(F,x) = \max_{x'} : |x-x'|=1 |F(x)-F(x')| \)
  - These can be very different: local can be much smaller than global
  - It is tempting (but incorrect) to calibrate noise to local sensitivity

- **Bad case for local sensitivity**: **Median**
  - Consider \( X = [0^n, 0, 0, T^{n-1}] \), \( X' = [0^n, 0, T^n] \), \( X'' = [0^n, T, T^n] \)
  - \( S(F,X) = 0 \) while \( S(F, X') = T \)
  - Scale of the noise will reveal exactly which case we are in

- Still, there has to be something better than always using global?
  - Such bad cases seem artificial, rare
Smooth Sensitivity

- Previous case was bad because local sensitivity was low, but “close” to a case where local sensitivity was high
- “Smooth sensitivity” combines sensitivity from all neighborhoods (based on parameter $\beta$)
  - $SS(F,x) = \max_{o \in O} LS(F,o) \exp(-\beta |o - x|)$
  - Contribution of output $o$ is decayed exponentially based on distance of $o$ from $x$, $|o - x|$
  - Can add Laplace noise scaled by $SS(F,x)$ to obtain (variant of) DP
Smooth Sensitivity: Example

- Consider the median function $M$ over $n$ items again
  - Compute the maximum change in the median for each distance $d$
  - $LS$ measures when median changes from $x_i$ to $x_{i+1}$
- So $LS$ at distance $d$ is at most $\max_{0 \leq j \leq d} (x_n/2 + j - x_n/2 + j - d - 1)$
  - Largest gap that can be created by inserting/deleting at most $d$ items
- Gives $SS(M, x) = \max_{0 \leq d \leq n} \exp(-d\beta) \max_{0 \leq j \leq d} (x_n/2 + j - x_n/2 + j - d - 1)$
  - Can compute in time $O(n^2)$
  - Empirically, exponential mechanism seems preferable
  - No generic process for computing smooth sensitivity
Sample-and-aggregate gives a useful template

- **Intuition**: sampling is almost DP - can’t be sure who is included
- Break input into moderate number of blocks, \( m \)
- Compute desired function on each block
- Snap to some range \([\text{min}, \text{max}]\) and aggregate (e.g. mean)
- Add Laplace noise scaled by sensitivity (max-min)
Sparse Data

- Suppose we have many (overlapping) queries, most of which have a small answer, but we don’t know which
  - We are only interested in large answers (e.g. frequent itemsets)
  - Two problems: time efficiency, and “privacy efficiency”

- Time efficiency:
  - Don’t want to add noise to every single zero-valued query
  - Assume we can materialize all non-zero query answers
  - Count how many are zero
  - Compute probability of noise pushing a zero-query past threshold
  - Sample from Binomial distribution how many to “upgrade”
  - Sample noisy value conditioned on passing threshold
Sparse Data – Privacy Efficiency

- Only want to pay for $c$ queries with that exceed threshold $T$
  - Assume all queries have sensitivity $S$
- Compute noisy threshold $T' = T + \text{Lap}(2S/\varepsilon)$
- For each query, add noise $\text{Lap}(2Sc/\varepsilon)$, only output if above $T'$
- Result is $\varepsilon$-DP
  - For “suppressed” answers, probability of seeing same output is about the same as if $T'$ was a little higher on neighboring input
  - For released answers, DP follows from Laplace mechanism
- Result is reasonably accurate: with high probability,
  - All suppressed answers are smaller than $T + \alpha$
  - All released answers have error at most $\alpha$
  for parameter $\alpha(c, 1/\varepsilon, S)$, and at most $c$ query answers $> T - \alpha$
Multiplicative weights

- The idea of “multiplicative weights” widely used in optimization
  - Up-weight ‘good’ answers, down-weight ‘poor’ answers
  - Applied to output of DP mechanism

- **Set-up:**
  - (Private) input, represented as vector $D$ with $n$ entries
  - $Q$, set of queries over $x$ (matrix)
  - $T$, bound on number of iterations
  - **Output:** $\varepsilon$-DP vector $A$ so that $Q(A) \approx Q(D)$
Multiplicative Weights Algorithm

- Initialize vector $A_0$ to assign uniform weight for each value
- For $i=1$ to $T$:
  - Exponential Mechanism $(\varepsilon/2T)$ to sample $j$ prop. to $|Q_j(A_i) - Q_j(D)|$
    - Try to find query with large error
  - Laplace Mechanism to estimate $\Delta = (Q_j(A) - Q_j(D)) + \text{Lap}(2T/\varepsilon)$
    - Error in the selected query
  - Set $A_i = A_{i-1} \cdot \exp(\Delta Q_j(D)/2n)$, normalize so that $A_i$ is a distribution
    - (Noisily) reward good answers, penalize poor answers
- Output $A = \text{average}_i nA_i$
  - Privacy follows via sequential composition of EM and LM steps
  - Accuracy (should) improve in each iteration, up to $\log$ iterations
Other topics

- Huge amount of work in DP across theory, security, DB...
- Many topics not touched on in this tutorial:
  - Connections to game theory and auction design
  - Mining primitives: regression, clustering, frequent itemsets
  - Efforts in programming languages and systems to support DP
  - Variant definitions: $(\varepsilon, \delta)$-DP, other privacy/adversary models
  - Lower bounds for privacy (what is not possible)
  - Applications to graph data (social networks), mobility data etc.
  - Privacy over data streams: pan-privacy and continual observation
Concluding Remarks

- Differential privacy can be applied effectively for data release
- **Care is still needed** to ensure that release is allowable
  - Can’t just apply DP and forget it: must analyze whether data release provides sufficient privacy for data subjects
- Many open problems remain:
  - Transition these techniques to tools for data release
  - Want data in same form as input: **private synthetic data**?
  - Allow **joining** anonymized data sets accurately
  - Obtain alternate (workable) **privacy definitions**

Thank you!
References – Basic Building Blocks

- **Differential privacy, Laplace Mechanism and Sensitivity**:
  - Differential Privacy. Cynthia Dwork, ICALP 2006

- **Geometric Mechanism**
  - Universally utility-maximizing privacy mechanisms. Arpita Ghosh, Tim Roughgarden, Mukund Sundararajan. STOC 2009

- **Sequential and Parallel Composition, Median Example**
  - Privacy integrated queries: an extensible platform for privacy-preserving data analysis. Frank McSherry. SIGMOD 2009.

- **Exponential Mechanism**
References – Applications & Transforms

♢ **Spatial Data Application**

♢ **Data Transforms**
  – Differential privacy via wavelet transforms. Xiaokui Xiao, Guozhang Wang, Johannes Gehrke, ICDE 2010
  – Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption. Vibhor Rastogi and Suman Nath, SIGMOD 2010
References – Advanced Mechanisms

- **Smooth Sensitivity, Sample and Aggregate**
  - Smooth Sensitivity and Sampling in Private Data Analysis. Kobbi Nissim, Sofya Raskhodnikova and Adam Smith. STOC 07

- **Sparse Data Processing**
  - Differentially Private Summaries for Sparse Data. Graham Cormode, Magda Procopiuc, Divesh Srivastava, and Thanh Tran. ICDT 2012

- **Multiplicative Weights**