CS573 Data Privacy and Security

Local Differential Privacy

Li Xiong
Privacy at Scale: Local Differential Privacy in Practice (Module 1)

Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang
Differential Privacy in the Wild (Part 2)
A Tutorial on Current Practices and Open Challenges

Ashwin Machanavajjhala, Michael Hay, Xi He
Outline

• Local differential privacy - definition and mechanisms
• Google: RAPPOR
• Apple: learning with LDP
Differential Privacy - Centralized Setting

Private Data D → Differential Privacy Mechanism → Statistics/Models

Trusted Data Aggregator
Problem

What are the frequent unexpected Chrome homepage domains?

→ To learn malicious software that change Chrome setting without users’ consent

Finance.com

Fashion.com

WeirdStuff.com

[Erlingsson et al CCS’14]
Why privacy is needed?

Liability (for server)
Storing unperturbed sensitive data makes server accountable (breaches, subpoenas, privacy policy violations)
Trying to Reduce Trust

• Centralized differential privacy setting assumes a trusted party
  • Data aggregator (e.g., organizations) that sees the true, raw data
  • Can compute exact query answers, then perturb for privacy

• A reasonable question: can we reduce the amount of trust?
  • Can we remove the trusted party from the equation?
  • Users produce locally private output, aggregate to answer queries
Local Differential Privacy Setting

\[ x_1 \quad Y_1 \leftarrow A(x_1) \]

\[ x_2 \quad Y_2 \leftarrow A(x_2) \quad Y_1 \]

\[ \ldots \ldots \]

\[ x_N \quad Y_N \leftarrow A(x_N) \quad Y_N \]

Data collector and analytics
Local Differential Privacy

- Having **each user run a DP algorithm** on their data
  - Then combine all the results to get a final answer

- On first glance, this idea seems crazy
  - Each user adds noise to mask their own input
  - So surely the **noise** will always **overwhelm the signal**?

- But ... noise can **cancel out** or be **subtracted out**
  - We end up with the true answer, plus noise which can be smaller
  - However, noise is still **larger than** in the **centralized case**
Local Differential Privacy: Example

• Each of \( N \) users has 0/1 value, estimate total population sum
  • Each user adds independent Laplace noise: mean 0, variance \( \frac{2}{\epsilon^2} \)

• Adding user results: true answer + sum of \( N \) Laplace distributions
  • Error is random variable, with mean 0, variance \( \frac{2N}{\epsilon^2} \)
  • Confidence bounds: ~95% chance of being within \( 2\sigma \) of the mean
  • So error looks like \( \sqrt{N/\epsilon} \), but true value may be proportional to \( N \)

• Numeric example: suppose true answer is \( N/2 \), \( \epsilon = 1 \), \( N = 1M \)
  • We see \( 500K \pm 2800 \): about 1% uncertainty
  • Error in centralized case would be close to 1 (0.001%)
Local Differential Privacy

• We can achieve LDP, and obtain reasonable accuracy (for large N)
  • The error typically scales with $\sqrt{N}$

• Generic approach: apply centralized DP algorithm to local data
  • But error might still be quite large
  • Unclear how to merge private outputs (e.g. private clustering)

• So we seek to design new LDP algorithms
  • Maximize the accuracy of the results
  • Minimize the costs to the users (space, time, communication)
  • Ensure that there is an accurate algorithm for aggregation
Randomized Response (a.k.a. local randomization)

With probability $p$, 
Report true value

With probability $1-p$, 
Report flipped value
Differential Privacy Analysis

• Consider 2 databases D, D’ (of size M) that differ in the j\textsuperscript{th} value
  • D[j] \neq D’[j]. But, D[i] = D’[i], for all i \neq j

• Consider some output O

\[
\frac{P(D \rightarrow O)}{P(D' \rightarrow O)} \leq e^\varepsilon \iff \frac{1}{1 + e^\varepsilon} < p < \frac{e^\varepsilon}{1 + e^\varepsilon}
\]
Utility Analysis

• Suppose \( n_1 \) out of \( n \) people replied “yes”, and rest said “no”

• What is the best estimate for \( \pi = \text{fraction of people with disease} = Y \)?

\[
\hat{\pi} = \frac{n_1/n - (1-p)}{(2p-1)}
\]

• \( E(\hat{\pi}) = \pi \)

• \( \text{Var}(\hat{\pi}) = \frac{\pi(1-\pi)}{n} + \frac{1}{n(16(p - 0.5)^2 - 0.25)} \)

  Sampling  Variance due to coin flips
LDP framework

• Client side
  • Encode: \( x = \text{Encode}(v) \)
  • Perturb: \( y = \text{Perturb}(\text{Encode}(v)) \)

• Server side
  • Aggregate: aggregate all \( y \) from users
  • Estimate the function (e.g. count, frequency)
Privacy in practice

• Differential privacy based on coin tossing is widely deployed!
  • In Google Chrome browser, to collect browsing statistics
  • In Apple iOS and MacOS, to collect typing statistics
  • In Microsoft Windows to collect telemetry data over time
  • From Snap to perform modeling of user preference
  • This yields deployments of over 100 million users each

• All deployments are based on RR, but extend it substantially
  • To handle the large space of possible values a user might have

• Local Differential Privacy is state of the art in 2018
  • Randomized response invented in 1965: five decades ago!
Outline

• Local differential privacy definition and mechanisms
• Google: RAPPOR
• Apple: learning with LDP
Google’s RAPPOR

• Each user has one value out of a very large set of possibilities
  • E.g. their favourite URL, www.nytimes.com

• Basic RAPPOR
  • Encode: 1-hot encoding
  • Perturb: run RR on every bit
  • Aggregate

• Privacy: $2\varepsilon$-LDP (2 bits change: 1 $\rightarrow$ 0, 0 $\rightarrow$ 1)
• Communication: sends 1 bit for every possible item in the domain
Bloom Filters & Randomized Response

- **RAPPOR**
  - **Encode**: Bloom filter using $h$ hash functions to $k$-bit vector
  - **Perturb**: apply Randomized Response to the bits in a Bloom filter (2-step approach)
  - **Aggregate**: Combine all user reports and observe how often each bit is set

- **Communication reduced to $m$ bits**
Client Input Perturbation

• Step 1: Compression: use $h$ hash functions to hash input string to $k$-bit vector (Bloom Filter)

Finance.com

Bloom Filter $B$
Permanent RR

• Step 2: Permanent randomized response $B \rightarrow B'$
  • Flip each bit with probability $f/2$
  • $B'$ is memorized and will be used for all future reports
Instantaneous RR

• Step 4: Instantaneous randomized response $B' \rightarrow S$
  • Flip bit value 1 with probability 1-r
  • Flip bit value 0 with probability 1-p

Why randomize two times?
- Chrome collects information each day
- Want perturbed values to look different on different days to avoid linking
Server Report Decoding

- Step 5: estimates bit frequency from reports $\tilde{f}(D)$
  - Take minimum estimate out of the k bits
- Step 6: estimate frequency of candidate strings with regression from $\tilde{f}(D)$

[Fanti et al. arXiv’16] no need of candidate strings
Privacy Analysis

• Recall RR for a single bit
  • RR satisfies $\varepsilon$-DP if reporting flipped value with probability $1 - p$, where $\frac{1}{1+e^{\varepsilon}} \leq p \leq \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$

• Exercise: if Permanent RR flips each bit in the k-bit bloom filter with probability $1-p$, which parameter affects the final privacy?
  1. # of hash functions: $h$
  2. bit vector size: $k$
  3. Both 1 and 2
  4. None of the above
Privacy Analysis

• Answer: # of hash functions: $h$
  • Remove a client’s input, the maximum changes to the true bit frequency is $h$.
  • Permanent RR satisfies $(h\varepsilon)$-DP

• Change a client’s input, 0->1, 1->0, permanent RR satisfies $(2h\varepsilon)$-DP
RAPPOR Demo
http://google.github.io/rappor/examples/report.html

**Simulation Input**

- Number of clients: 100,000
- Total values reported / obfuscated: 700,000
- Unique values reported / obfuscated: 50

**RAPPOR Parameters**

- $k$: Size of Bloom filter in bits, 16
- $h$: Hash functions in Bloom filter, 2
- $m$: Number of Cohorts, 64
- $p$: Probability $p$, 0.5
- $q$: Probability $q$, 0.75
- $f$: Probability $f$, 0.5
The RAPPOR approach is implemented in the Chrome browser.

- Collects data from opt-in users, tens of millions per day
- Open source implementation available

Tracks settings in the browser, e.g. home page, search engine:
- Many users unexpectedly change home page → possible malware

Typical configuration:
- 128 bit Bloom filter, 2 hash functions, privacy parameter ~0.5
- Needs about 10K reports to identify a value with confidence
Outline

• Local differential privacy definition and mechanisms
• Google: RAPPOR
• Apple: learning with LDP
Apple: Learning with Privacy at Scale

- Similar problem to RAPPOR: count frequencies of many items
  - For simplicity, assume that each user holds a single item
  - To reduce burden of collection, can size of summary be reduced?

- Instead of Bloom Filter, make use of sketches
  - Similar idea, but better suited to capturing frequencies

Adapted from: Privacy at Scale: Local Differential Privacy in Practice

Learning with Privacy at Scale, Apple Machine Learning Journal, Vol 1, Issue 8, December 2017

• Similar problem to RAPPOR: count frequencies of many items
  • For simplicity, assume that each user holds a single item
  • To reduce burden of collection, can size of summary be reduced?

• Instead of Bloom Filter, make use of sketches
  • Similar idea, but better suited to capturing frequencies
Count-Mean Sketch (CMS)

- Client side
  - Encode: randomly samples a hash function $j$ from a set of candidate hash functions, and encode the item into a 1-hot vector of size $m$
  - Perturb: Random Response on each bit
  - Send the perturbed vector and the selected hash function index $j$ to server

- Privacy: $2\varepsilon$-LDP
- Communication: $m$ bits
  - Can also use multiple hash functions and send multiple vectors for better utility

Adapted from: Privacy at Scale: Local Differential Privacy in Practice
Count-Mean Sketch (CMS)

- Server side aggregation
  - Construct a sketch matrix M by aggregating the perturbed vectors
  - k rows – one for each hash function
  - m columns - size of the perturbed vector
  - Adds the perturbed count for row j given hash index j from the device
  - Estimate frequency for each row j and compute mean of the estimate

- Utility
  - Variance inversely proportional to m and k

Adapted from: Privacy at Scale: Local Differential Privacy in Practice
Hadamard Count Mean Sketch (HCMS)

• Goal: reduce client communication without sacrificing utility by transmitting 1 bit
• Intuition: spread information from the 1-hot sparse vector to a dense vector so we can sample 1 bit to keep the signal
• Idea: use Hadamard transform (a discrete Fourier transform)
  • The user can sample one entry in the transformed vector
  • No danger of missing the important information – it’s everywhere!
• Aggregator can invert the transform to get the sketch back

\[
\begin{bmatrix}
H^* & H^*
\end{bmatrix} = \begin{bmatrix}
-1 & 1 & 1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 & -1 & 1 \\
1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & -1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & -1 & -1 & 1
\end{bmatrix}.
\]
Hadamard Count Mean Sketch (HCMS)

- Client side
  - Encode: randomly sample a hash function \( j \), and encode the item into a 1-hot vector \( \mathbf{v} \)
  - Hadamard transform: \( \mathbf{v}' = H_m \mathbf{v} \)
  - Sampling 1 bit \( l \) from \( \mathbf{v}' \)
  - Perturb the bit and send hash function index \( j \), sampled bit index \( l \), and perturbed bit

Randomly select hash index (3)
Randomly select column (2)

Flip with DP

Encrypted channel
Hadamard Count Mean Sketch (HCMS)

- Server side aggregation
  - Construct a sketch matrix $M$
  - $k$ rows – one for each hash function
  - columns based on the sampled bit index
  - Transform $M$ back using inverse Hadamard matrix
  - Estimate frequency for each row and compute mean

User device  $\rightarrow (3, 2, x^{(n)})$

Debias $\rightarrow y^{(n)}$

Sketch matrix $M$

$\begin{pmatrix}
0 & 0 & 0 & 0 & \ldots \\
0 & 0 & 0 & 0 & \ldots \\
0 & 0 & 0 & 0 & \ldots \\
0 & 0 & 0 & 0 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{pmatrix}$

Transform $MH$

1  120  231  \ldots  98  \ldots  72  \ldots  271  \ldots
2  823  82  \ldots  879  \ldots  314  \ldots  21  \ldots
3  68  81  \ldots  254  \ldots  64  \ldots  681  \ldots
\vdots
k  124  434  \ldots  345  \ldots  543  \ldots  444  \ldots

Count(example.com) = Average( 231  21  254  \ldots  543 )
Apple’s Differential Privacy in Practice

• CMS settings: $m=1024$, $k=65,356$, $\varepsilon=4$ (dictionary of 2600 emojis)

• Apple uses their system to collect data from iOS and OS X users
  • Popular emojis: (heart) (laugh) (smile) (crying) (sadface)
  • “New” words: bruh, hun, bae, tryna, despacito, mayweather
  • Which websites to mute, which to autoplay audio on!

Adapted from: Privacy at Scale: Local Differential Privacy in Practice
Microsoft telemetry data collection

• Microsoft want to collect data on app usage
  • How much time was spent on a particular app today?
  • Allows finding patterns over time

• Makes use of multiple subroutines:
  • 1BitMean to collect numeric data
  • dBtFlip to collect (sparse) histogram data
  • Memoization and output perturbation to allow repeated probing

• Has been implemented in Windows since 2017
MS Telemetry Collection in Practice

• Deployed in Windows 10 Fall Creators Update (October 2017)
  • Collects number of seconds users spend in different apps
  • Parameters: $\epsilon = 1$ and $\gamma = 0.2$
  • Collection period: every 6 hours

• Collects data on all app usage, not just one at a time
  • Can analyze based on the fact that total time spent is limited
  • Gives overall guarantee of $\epsilon = 1.672$ for a round of collection