CS208: Applied Privacy for Data Science
The Local Model: Implementations

James Honaker & Salil Vadhan

School of Engineering & Applied Sciences
Harvard University

April 1, 2019

Center for Research on Computation and Society

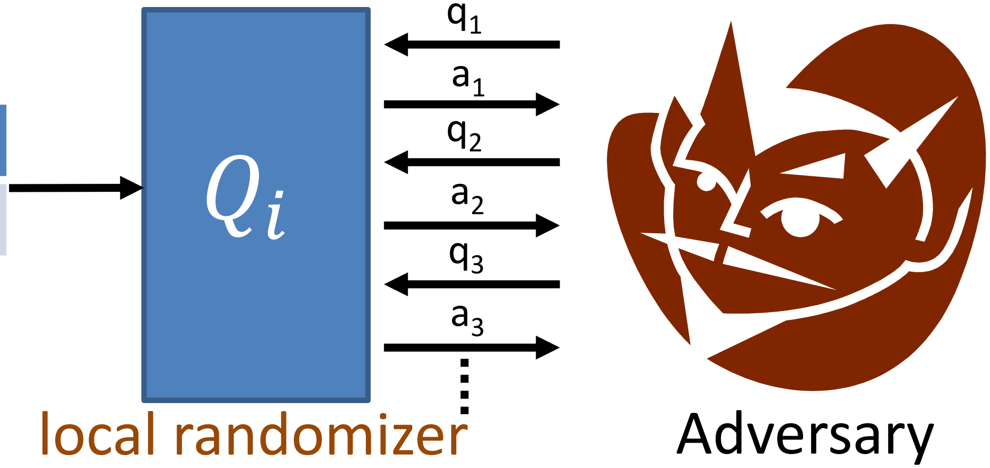
Implementations of Local DP

Telemetry="automated communications process by which measurements and other data are collected at remote or inaccessible points and transmitted to receiving equipment for monitoring" [https://en.wikipedia.org/wiki/Telemetry]

- Google RAPPOR (2014, Chrome Browser)
  - Collecting billions of reports per day, software monitoring
  - What features being used, failure & performance stats, default search engine (could be set by malware), etc.
- Apple "Learning with Privacy" (2016, iOS 10, Mac OS Sierra)
  - New Words, Popular Emojis, Video Playback Preferences in Safari, High Energy&Memory Usage in Safari, and Popular HealthKit Usage.
- Microsoft "Collecting Telemetry Data Privately" (2017, Windows Insiders in Windows 10 Fall Creators Update)
  - App usage statistics

Local DP

Sex	Blood	⋯	HIV?
F	B	⋯	Y



Require: for all i, x_i, x_i' ~~differing on one row,~~ all strategies A

$$\Pr[A \text{ outputs YES after interacting w/} Q_i(x_i)]$$
$$\leq e^{\varepsilon} \cdot \Pr\left[A \text{ outputs YES after interacting w/} Q_i(x_i')\right] + \delta$$

Recap: DP histograms

- Local randomizer $Q(x_i)$ for $x_i \in \{1, \dots, D\}$
  1. Construct "1-hot" vector $e_{x_i} = (0,0, \dots, 0,1,0, \dots, 0) \in \{0,1\}^D$.
  2. Apply $(\varepsilon/2)$-DP RR to each coordinate to get $y_i \in \{0,1\}^D$:
  $$y_i[j] = \begin{cases} e_{x_i}[j] & \text{w.p. } e^{\varepsilon/2}/(1+e^{\varepsilon/2}) \\ 1 - e_{x_i}[j] & \text{o.w.} \end{cases}$$
  3. Send y_i to server.
- Server uses $(y_1, \dots, y_n)$ to estimate histogram $f = \sum_i e_{x_i}$.
- Error per bin $O(\sqrt{n}/\varepsilon)$.

Q: challenges in applying this to telemetry applications?

Handling large domains

- Server chooses random hash func. $h : \{1, \dots, D\} \to \{1, \dots, m\}$, for $m \ll D$, sends h to all users.

- Use RR protocol to obtain approximate histogram $\hat{f} \in \mathbb{Z}^m$ of $h(x_i)$'s.

- For a value $v \in \{1, \dots, D\}$, estimate the frequency of v as:
$$\frac{m \cdot \hat{f}[h(v)] - n}{m - 1}$$

- Claim: this is an unbiased estimator

Reducing the Variance

1. Randomly partition users into k cohorts, each using a different hash function h_j, sum estimators over the cohorts.
   - Used by Google and Apple.

2. Use ℓ hash functions per cohort.
   - Used by Google, inspired by Bloom Filters.
   - Now applying RR to a 2ℓ-hot vector.
   - Experiments show that 2 hash functions per user does best on precision and recall, but doesn’t compare to just 1.

Inference on the Reports (Google)

- For simplicity, assume one hash function per cohort.
- Histogram of cohort j's hashed values: $f_j = \sum_{i \in C_j} e_{h_j(x_i)} \in \mathbb{N}^m$.
- From noisy reports, estimate histogram $\hat{f}_j$.

To estimate frequencies for a given $v_1, \dots, v_s \in \{1, \dots, D\}$:

- Construct $m \times s$ matrices $M_j[a,b] = \begin{cases} 1 & \text{if } h_j(v_a) = b \\ 0 & \text{o.w.} \end{cases}$
- Use LASSO regression to find a sparse $\hat{F} \in \mathbb{N}^s$ such that

$$\begin{pmatrix} M_1 \\ \vdots \\ M_k \end{pmatrix} \hat{F} \approx k \cdot \begin{pmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_k \end{pmatrix}$$

- Do another OLS regression on nonzero coordinates of $\hat{F}$ to re-estimate coeffs, plus std errors, p values (with correction for false discovery)

Discovering Unknown Values (Apple)

- Method described so far requires server to decide on a small set values $v_1, \dots, v_s$ to estimate frequencies of.

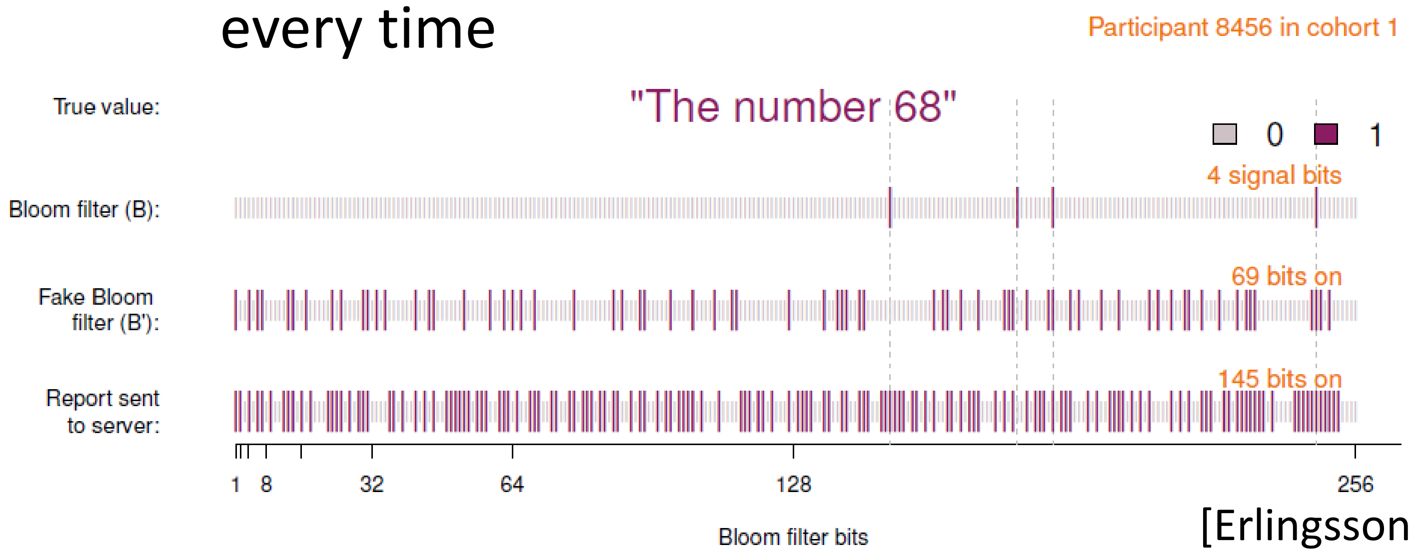
- Goal: find unanticipated frequent values v

- Idea: reconstruct v one symbol at a time
  - do RR on $h(x_i)||x_i[j]$ for each bitposition $j = 1, \dots, \log D$.
  - $\hat{f}[w||\sigma_j]$ is large $\Rightarrow$ there probably is a frequently occurring value $v \in \{1, \dots, D\}$ such that $h(v) = w$ and $v[j] = \sigma_j$.
  - $\hat{f}[w||\sigma_1], \dots, \hat{f}[w||\sigma_{\log D}]$ large $\Rightarrow v = \sigma_1 \cdots \sigma_{\log D}$

Controlling Privacy Loss Over Time

- Hope: lower privacy loss if value not changing often (e.g. stated gender, default search engine)

- Memoization (Google): Two-level RR
  - ε_1-DP "permanent RR": output reused until x_i changes
  - ε_2-DP "instantaneous RR": applied to permanent RR output every time



[Erlingsson, Pihur, Korolova 2014]

Controlling Privacy Loss over Time (cont.)

- Memoization for continuous values v (Microsoft):
  - Round value to a randomly shifted grid.
  - Small changes in v unlikely to change rounded value.
  - Rounded value is unbiased estimator of v.

- Anonymity (Apple):
  - Do not track which reports came from which users.
  - Recent work shows that this amplifies privacy.

Utility

- RAPPOR rule-of-thumb: to detect items that have frequency $p \in [0,1]$, need a sample size $n \gtrsim 10/p^2$.
  - For $p = .01$, need $n \gtrsim 100{,}000$.

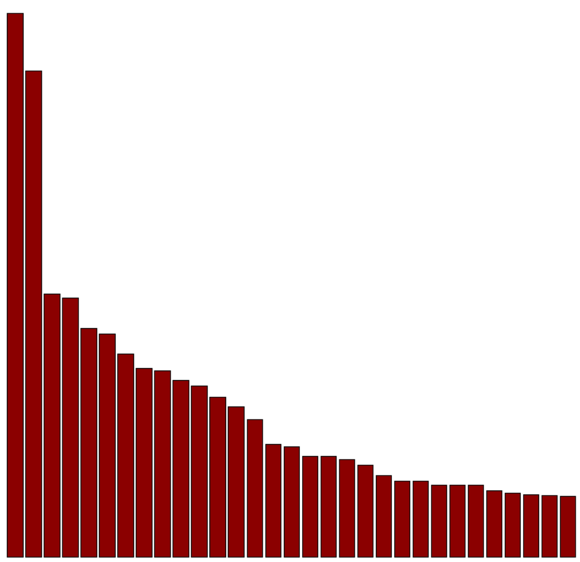


Figure 6: Relative frequencies of the top 31 unexpected Chrome homepage domains found by analyzing ~14 million RAPPOR reports, excluding expected domains (the homepage "google.com", etc.).

- Found from > 8500 candidate domains
- Account for 85% of all users' homepages

[Erlingsson, Pihur, Korolova 2014]

Critiques

- Apple (and Microsoft?) have not open-sourced code or specified their privacy loss parameters.

- Tang et al. [2017] reverse-engineering Apple's use of DP:
  - Privacy loss per report: $\varepsilon = 1$ or 2.
  - Max privacy loss per day over 4 types of reports: $\varepsilon = 14$
  - Privacy loss over multiple days: unbounded

- Google collects lots of user data without DP.

- Google phasing out RAPPOR due to poor accuracy.
  - Switching to Prochlo, using anonymity to boost local DP. [Bittau et al. 2017]

Some Responses

- Formally reasoning about privacy loss in these settings is progress.

- Limitations and heuristic patches inspire theoretical work:
  - Privacy amplification via anonymity/shuffles. [Bittau et al. 2017, Cheu et al. 2019,…]
  - New algorithms that ensure local DP even over long time, and give utility if value doesn’t change too much too often. [Joseph et al. 2018]