Survey on Conversion Measurement APIs

Joel Pfeiffer

Microsoft Advertising

Nomenclature

- Features – Contextual information about the click/impression
  - Keys
  - Sparse – CampaignId in space of uint64 values
  - Dense – Fixed dimension of continuous counts
    - X := [#CampaignClicks, #CampaignConversions, #QueryClicks, #QueryConversions] (4Dim)
- Labels – Value assigned to a conversion
  - Boolean (conversion happened or not)
  - Counts of conversions
  - Conversion values (spend)
- Trusted Helper: Requires complete trust, will not reveal data and will not corrupt outputs
- Semi-Trusted Helper: Requires trust that the helper will not corrupt outputs, but does not require they will violate privacy (true data not revealed)
- Malicious Helper: Robust against a helper actively corrupting the outputs of the function

Order

- Browser Event Reporting APIs
  - Google Event Conversion API
  - Apple Private Conversion Measurement
- Browser Aggregate Reporting
  - Trusted third party server
  - MPC
    - Prio
    - Bucketization
    - Incremental DPF
    - Masked LARk
- Private Join Aggregate Reporting API (Meta / Mozilla)

Browser Event Reporting API (Generic)

- Browser level mechanism to attribute conversion to click
  - Extensions for impressions / views
- Core additional metadata attached to links
  - attributiondestination: eTLD+1 of the advertiser
  - attributionreportto: Usually, the ad network
  - attributioneventid: 64-bit event id
- Metadata stored in browser for click

… <user browses>

- User action on http://<attributiondestination>/site
  - Conversion pixel on page: http://<attributionreportto/conversiontracker
  - Site redirects to: https://<attributionreportto>/.well-known/attribution-reporting/trigger-attribution?trigger_data=trigger_data
- Browser uses the attributiondestination and attributionreportto fields, determines the conversion happened for attributioneventid
  - Reports back to attributionreportto downstream what happened

Browser Event Reporting API

- Number of source event id bits varies between proposals
  - Can possibly be used for training
  - Generally, can be used for aggregation
- Number of trigger event bits varies between proposals
- Various random noise inserted
    - E.g., 5% random bits
- “Fire-and-forget”
- Timing delays / attacks
- For some of the proposals, (e.g., FLEDGE), some features not available

Order

- Browser Event Reporting APIs
  - Google Event Conversion API
  - Apple Private Conversion Measurement
- Browser Aggregate Reporting
  - Trusted third party server
  - MPC
    - Prio
    - Bucketization
    - Incremental DPF
    - Masked LARk
- Private Join Aggregate Reporting API (Meta / Mozilla)

Aggregate Reporting Helper Responsibilities

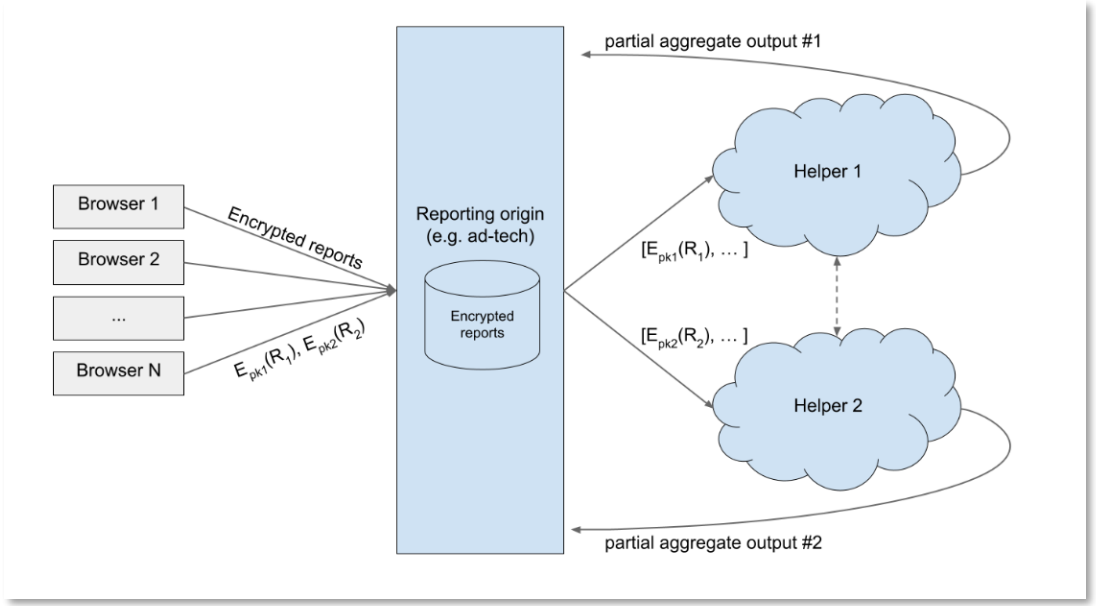
- Given encrypted data d and a function f, computes $f(d)$
- Responsible for **enforcing global privacy**
  - Differentially private outputs
  - K-anonymity
  - Privacy budget
- Most proposals are setup such that
  - Individual helpers cannot reveal user information, as they're always given a partial view of a user interaction
  - Helpers are trusted to return the right result, and not maliciously corrupting the outputs
  - This latter assumption can be periodically validated by advertising services

Trusted 3P Server

- Simply have a trusted third party, which agrees to not release information
- Trust verified through (e.g.) code audits, secure enclaves
- Queries are issued to the trusted server, which carries out the query
- Trusted server in charge restricting allowable queries, adding DP noise, etc.
- Extremely simple concept, easy to implement, efficient, etc.
- Single point of failure
  - Trust broken?
  - Hack / malicious outsiders have info in one place
- Cost / Supply

MPC Third Party Helpers

- Uses Secure MultiParty Compute to implement a trusted third party abstraction
- Rather than returning conversion information directly, returns a secret shared representation
  - Allow side information / contextual information
  - Half of the information is for Helper 1
  - Half of the information is for Helper 2
  - Looking at Helper 1 or Helper 2 alone appears to be random noise
  - Data for each helper is decrypted/encrypted with private/public keys
- Encrypted shares are given to the ad network
- Ad Network must call Helper 1 and Helper 2 to recover aggregates
- Helper 1 and Helper 2 enforce privacy

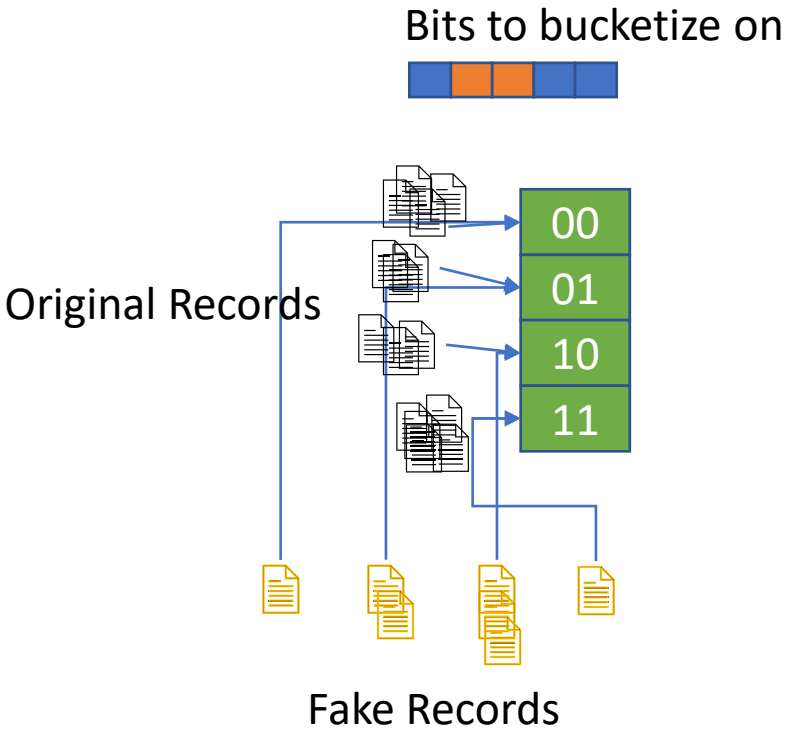


MPC - Prio

- Client secret shares a value intended for multiple servers
- Client secret shares side information to aggregate over
- In addition to secret shared side information, clients additionally provide Secret Shared non-interactive proof (SNIP)
  - Side information for servers to verify that they have a “Valid” sample, preventing poisoning attacks
- Extremely efficient for small number of bits
- Semi-trusted helpers
  - Features and Labels

Bucketization

- Handles aggregation and (in particular) aggregation on subsets
- Assumes some number of bits (e.g., 128)
  - Bits are Boolean shared between 2 parties
- High level idea
  - Take a subset of bits as the key
    - 2^keybits should be tractable
    - Each value of 2^keybits is called a “bucket”
  - Both helpers insert fake records proportional to 2^keybits
  - Shuffle and mask (3rd helper) to reassign keybits and obfuscate records from original 2 parties
  - Simply sum on the bits now to reconstruct
  - Can subsequently split on other bits

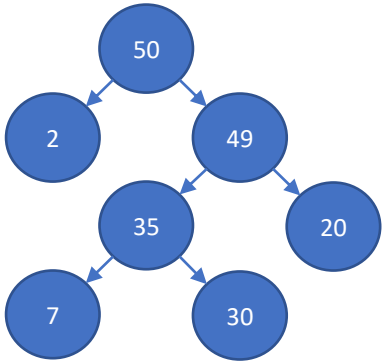


Bucketization

- Hides features / labels from Helpers
- Operates well within 2^|keybit| dense subsets
  - Small enough keybits should be tractable
  - Can operate on any subsets of bits
  - Can perform arbitrary hierarchical evaluations
  - Flexible ordering on the queries
- Needs (not required, but far, far more efficient) a third helper
  - Third helper can be chosen at evaluation time (not determined by browser)
- Semi-trusted helpers
  - Features and Labels
  - Some protection against malicious helpers

Incremental DPF (Google)

- Point Function
  - $x, y \in \{0,1\}^*$ *(e.g., 010011, 11011)*
    - $P_{x,y}(x) = y$
    - $P_{\{x,y\}}(x') = 0^{|y|}$
- Distributed Point Function
  - Key pair k_0, k_1 s.t.
    - $F_{k_0} \oplus F_{k_1} \coloneqq P$
- Incremental Distributed Point Function
  - Evaluate first b bits (using MPC) and get DP sum over all remaining bits
  - Can use b to help with b+1
  - Trim low occurrences to speedup



Incremental DPF

- Works well within sparse spaces
  - E.g., find 2^16 most common out of 2^128 space
- Fixed, hierarchical structure
- Semi-trusted helpers
  - Features and values

Masked LARk

- Intended as a private MapReduce framework
  - Browser is a map
  - Helpers are a reduce
- Primary example function is training, particularly SGD
- Browser creates multiple labels (true and fake) and sends both to the helpers (helpers are not told which are true and which are fake)
- Browser additionally provides secret shared masks which either include or exclude a sample from the computation
  - Masks appear random to the helpers
- Servers compute a function per sample, multiply each sample by its mask, and sum
  - Any bilinear function is fine, which includes summation
- Training requires a modification to always send some value, to handle the negative cases

Masked LARk

- Attacks
  - High cardinality label space – ad network could send ID as a label, then collude with helper
    - Randomly quantized labels
  - Requires default value sent if no conversion
    - Need to randomize / delay the record
- Non-communicative Helpers
  - Labels: Semi-trusted helpers
  - Features: Trusted Helpers
  - Models: Trusted Helpers

Private Joins: Meta Proposal

- Separates out *click* app versus *conversion* app
  - Focuses on LinkedIn/Facebook user view resulting in an Edge/Chrome conversion
- In particular, using a common API, apps and websites will be able to set and use a match key
  - This match key can be set per site, similar to a dictionary:
    { “linkedin.com” : “match_key_li”}
  - These can be used to share event information, but not readable outside a common API
  - Presumably user must be signed on in both locations to set the keys
- Data, either click or conversion, is encrypted using the API and corresponding match key for the reporting site, and sent to the advertising service

Private Joins: Meta Proposal

- Clicks and conversions are subsequently joined using MPC
- Join keys are blinded with homomorphic encryption to destroy the match keys being read by MPC
  - Data is then shuffled
  - Joining done with a garbled circuit and time ordering
- Aggregation can now be done
- Semi-trusted helpers

References

- Google Conversion Event API: https://github.com/WICG/conversion-measurement-api/blob/main/EVENT.md
- Google Aggregate API: https://github.com/WICG/conversion-measurement-api/blob/main/AGGREGATE.md
- PCM: https://webkit.org/blog/11529/introducing-private-click-measurement-pcm/
- Masked LARk
  - Explainer: https://github.com/WICG/privacy-preserving-ads'
  - Paper: https://arxiv.org/abs/2110.14794
  - Prototype: https://github.com/microsoft/MaskedLARk
- Bucketization:
  - https://eprint.iacr.org/2021/1490
- IPA:
  - Prototype: https://github.com/martinthomson/raw-ipa?ref=rustrepo.com
  - Paper: https://docs.google.com/document/d/1KpdSKD8-Rn0bWPTu4UtK54ks0yv2j22pA5SrAD9av4s/edit#heading=h.f4x9f0nqv28x