# Gordon: Privacy Budget Management for W3C's Privacy-Preserving Attribution API


## Abstract

Privacy-preserving advertising APIs like Privacy-Preserving Attribution (PPA) promise to enhance web privacy while enabling effective ad measurement. PPA replaces cross-site tracking with encrypted reports governed by differential privacy (DP), but current designs lack a principled approach to privacy budget management—creating uncertainty around critical design decisions. We present *Gordon*, a privacy budget manager for PPA that clarifies per-site budget semantics and introduces a global budgeting system grounded in resource isolation principles. Gordon enforces utility-preserving limits via quota budgets and improves global budget utilization through a novel batched scheduling algorithm. Together, these mechanisms establish a robust foundation for enforcing privacy protections in adversarial environments. We implement Gordon in Rust and Firefox and evaluate it on real-world ad data, demonstrating its resilience and effectiveness.


## 1 Introduction

Privacy-preserving advertising APIs, now under development and standardization in major browsers via the W3C, offer a rare opportunity to enhance online privacy while sustaining the web's primary funding model. Historically, browsers have lacked structured support for ad-related tasks like *conversion attribution measurement*, which requires linking ads viewed on content sites to purchases made on seller sites—a cross-origin function fundamentally at odds with the same-origin principle that underpins browser design. This lack of support for the advertising workload has fueled widespread cross-site tracking through third-party cookies, fingerprinting, and other workarounds. The goal of the new APIs is to provide a structured, privacy-preserving alternative that aligns with browser principles while meeting advertising needs. However, these APIs remain in early stages, with technical challenges still unresolved—creating an opportunity for academic contribution.

Such collaborations have already had impact, underscoring that the space is ripe for foundational work. The *Cookie Monster* paper, presented at SOSP last year [26], introduced the first formal framework based on individual differential privacy (individual DP) [8] to systematically analyze and optimize these APIs—a framework later adopted by Google in privacy analysis of its ARA API [11]. That same Cookie Monster framework now underpins Privacy-Preserving Attribution (PPA) [20], the API standard being drafted by Private Advertising Technology Working Group (PATWG), a W3C working group that includes representatives from all browsers [22]. We are active participants in PATWG, tackling technical challenges from a scientific perspective to help advance the APIs' practicality under strong privacy guarantees.

In this paper, we address a key open challenge: *privacy budget management in PPA*. PPA replaces cross-site tracking with a system where content sites register ads with the browser, seller sites request encrypted reports, and reports are only accessible via DP aggregation using secure multi-party computation or a trusted execution environment. Before sending an encrypted report, the browser deducts privacy loss from a *per-site privacy budget*, limiting how much new information a site can infer about a user. While PPA, through Cookie Monster's algorithm, optimizes privacy loss accounting within each per-site budget using individual DP, it does not address how to manage these granular budgets to balance privacy with utility in an adversarial advertising ecosystem.

The absence of a principled approach to privacy budget management has led to unresolved questions within PATWG, creating uncertainty in key design decisions. For instance, should some sites get budget while others do not—and if so, based on what criteria?[1] Should there be a cap on how many sites are allocated budget, and if so, how can we prevent a denial-of-service attack where one entity exhausts it?[2] Should API invocations be rate-limited to prevent privacy or DoS attacks? To date, there is no consensus, largely due to the lack of a principled foundation to drive the design.

We describe *Gordon*, a *privacy budget manager for PPA* that addresses semantic gaps in per-site privacy loss accounting and challenges introduced by the coarse-grained global budget PPA incorporates to protect users against adversaries controlling many sites. To clarify the semantics of PPA's ambiguous per-site budgeting—often affected by shifting roles of third parties in the advertising ecosystem—we propose changes to the PPA interface, protocol, and terms of use, some of which have already been accepted by PPA.

For the global budget, the challenge is configuring and managing it to support benign workloads while resisting depletion by malicious actors. Our insight is to treat the global privacy budget as a *shared resource*—analogous to traditional computing resources but governed by privacy constraints—and to apply classic resource isolation techniques, such as quotas and fair scheduling [18, 12], in this new domain. Beyond per-site and global budgets, Gordon introduces *quota budgets* that regulate global-budget consumption, ensuring graceful utility degradation for benign sites under attack. It does so by forcing adversaries to operate within expected workload bounds—which they can currently evade to wreak havoc on PPA's global budget. Further, to address underutilization of

[1] Live discussion in W3C's PAT community group, April 2024.

[2] https://github.com/w3c/ppa/issues/69, January 2025.

the global privacy budget caused by static quota partitioning, we propose a scheduling algorithm that reallocates unused capacity to otherwise-blocked requests. Together, these mechanisms establish a principled, practical foundation for PPA and give browsers a basis for enforceable defenses, along with guidance on where to focus.

We implement Gordon in two components: (1) pdslib, a generic on-device individual DP library that subsumes Cookie Monster and extends it with Gordon's budget management, and (2) integration into Mozilla Firefox's Private Attribution, a minimal PPA implementation. Upon release, these prototypes will serve as reference implementations for PPA, a service the PATWG has acknowledged as valuable.

We evaluate Gordon on a dataset from the Criteo ad-tech company, showing that: (1) well-chosen quotas preserve high utility for benign workloads, (2) quotas isolate benign sites under attack, and (3) batched scheduling boosts utilization without sacrificing isolation.

# 2 PPA Overview and Gaps

## 2.1 PPA architecture

Fig. 1(a) illustrates the architecture of *Privacy-Preserving Attribution (PPA)*, W3C's browser-based API that enables *conversion attribution measurement* while preserving user privacy. Traditionally, browsers enforce a *same-origin policy*, while conversion attribution–the process of determining whether users who see an ad later make a purchase–is inherently *cross-origin*. It requires linking ad impressions shown on content sites (e.g., *news.ex*, *blog.ex*) to conversions occurring on advertiser sites (e.g., *shoes.ex*). In the absence of a structured API for this, advertisers rely on workarounds like third-party cookies, fingerprinting, and backend data exchanges—bypassing browser policies to accommodate workloads misaligned with current API structures.

PPA addresses this gap by enabling *cross-origin ad measurement* while preserving *single-origin privacy*, using differential privacy (DP) and secure aggregation via secure multi-party computation (MPC) or a trusted execution environment (TEE). This design bounds cross-origin information leakage, allowing the API to support effective ad measurement while upholding the intention of the browser's same-origin policy.

PPA defines four principals. **Impression sites** (*news.ex*, *blog.ex*) are content sites where ads are displayed. These sites register ad impressions with the browser using the function `saveImpression()`. **Conversion sites**, a.k.a. **advertiser sites** (*shoes.ex*), are sites where purchases or other conversions occur. When a user does a conversion, these sites invoke `measureConversion()` to link the event to any relevant prior ad impressions. **Intermediary sites** (*r1.ex*, *r2.ex*) are adtechs, typically embedded as frames in impression and conversion sites, that facilitate ad delivery and measurement. Unlike traditional tracking-based adtechs, they don't collect cross-site data directly but receive encrypted reports via a third function, `getReport()`, which they then submit for secure aggregation. **Aggregation services** (e.g., *divviup.org*) are trusted MPC/TEE services that aggregate encrypted reports, applying DP to produce aggregated conversion metrics while ensuring no single entity can reconstruct individual user data.

## 2.2 Example workflow

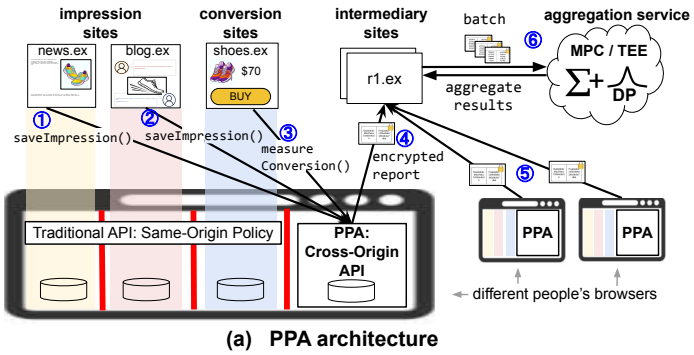
Fig. 1(b) shows an example workflow for PPA, consisting of six steps (the same steps are also marked in the Fig. 1(a) architecture). The example entails an advertiser, *shoes.ex*, that launches an ad campaign to promote a new product. To compare the effectiveness of two ad creatives–a colorful ad highlighting the shoe's design and a black-and-white ad emphasizing materials and comfort–*shoes.ex* partners with two placement adtechs, *r1.ex* and *r2.ex*. Each adtech places the ads on content sites, *e.g., r1.ex* on *blog.ex* and *r2.ex* on *news.ex*. In addition to placing ads, these adtechs provide a *measurement service* that allows *shoes.ex* to compare the performance of its creatives within their respective networks.

① When a user visits *blog.ex*, *r1.ex* displays the colorful ad and registers the impression by calling `saveImpression()` with the parameters shown in the figure. ② Later, the user visits *news.ex*, where *r2.ex* displays the black-and-white ad and registers it by also calling `saveImpression()`. These impressions are stored *locally in the browser* within an Impression Store, along with important metadata, shown in Fig. 1(c).

③ Subsequently, if the user visits *shoes.ex* and purchases the shoes for $60, the site invokes `measureConversion()` with the parameters shown in Fig. 1(b). This function searches the Impression Store in the browser for *relevant impressions*, matching the `impressionSite` and `conversionSite` metadata of the impressions to the parameters of `measureConversion()`. It then generates an *attributionObject*, which encapsulates the attribution histogram and manages privacy loss accounting. Assuming that PPA applies *uniform attribution*, it will assign the $60 conversion value equally between the two registered impressions, assigning $30 to each and resulting in the following attribution histogram: `{1:30, 2:30}`.

④ The attributionObject is *lazy*, i.e., no privacy loss occurs until it is used to request a report. To support DP queries, *shoes.ex* hands over the attributionObject to the *r1.ex* and *r2.ex* contexts within the browser, which invoke `attributionObject.getReport()`, specifying the aggregation service they intend to use (from a list of such services trusted by the browser). The browser processes these invocations by: (1) filtering the attribution histogram so that each intermediary only sees its own contributions (*r1.ex* gets `{1:30}`, *r2.ex* gets `{2:30}`); (2) encrypting the report and secret-sharing it (if MPC is used), while attaching some critical parameters as authenticated data, such as `epsilon` and `maxValue`; and (3) performing privacy loss accounting before sending the encrypted reports over to the intermediaries.

⑤ As more users purchase *shoes.ex*'s advertised product, additional encrypted reports are generated, each containing zero, one, or two attributed ads. ⑥ The intermediaries




(a) PPA architecture

① **Epoch e1:** User visits *blog.ex*. *r1.ex* script embedded within *blog.ex* displays colorful ad (*id 1*) for *shoes.ex*'s campaign (*0x22*) and invokes:
`saveImpression(conversionSites=[shoes.ex], filterData=0x22, histogramBucket=1)`

② **Epoch e2:** User visits *news.ex*. *r2.ex* script displays a black-and-white ad (*id 2*) and invokes:
`saveImpression(conversionSites=[shoes.ex], filterData=0x22, histogramBucket=2)`

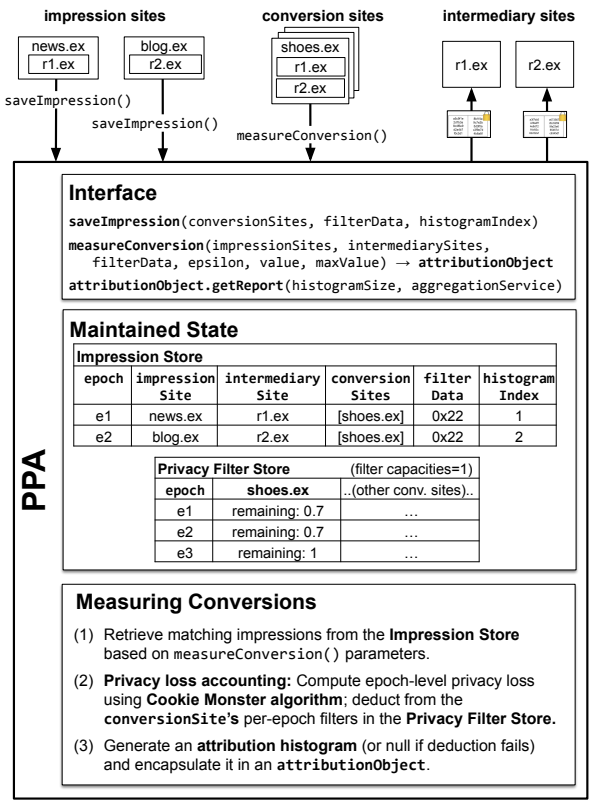
③ **Epoch e3:** User visits *shoes.ex* and purchases $60 of advertised shoes. *Shoes.ex* invokes:
`measureConversion(impressionSites=[blog.ex,news.ex], intermediarySites=[r1.ex,r2.ex], filterData=0x22, epsilon=0.5, value=60, maxValue=100) → attributionObject`
**AttributionObject** contains the full attribution histogram `{r1.ex:1=30, r2.ex:2=30}` and privacy loss tracking. Privacy loss is applied when `getReport(.)` is called.

④ *Shoes.ex* hands attributionObject to *r1.ex* and *r2.ex* embedded scripts, which invoke:
`attributionObject.getReport(histogramSize=2, aggregationService=divviup.org)`
to retrieve **encrypted reports**. *r1.ex* gets encryption of `{1:30}`, *r2.ex* gets encryption of `{2:30}`, both with authenticated data: `{epsilon=0.5, maxValue=100}`.

⑤ As other users convert on *shoes.ex*, *r1.ex* and *r2.ex* get encrypted reports from them and batch them.

⑥ At some point, *r1.ex* sends the batch to the aggregation service, which uses MPC or TEE to check consistency of authenticated data and uniqueness of report use. Then sums all the histograms and adds noise, e.g. from `Laplace(maxValue/epsilon)`, to ensure `epsilon`-DP.

(b) Example workflow




(c) Privacy loss accounting with Cookie Monster

**Fig. 1. PPA overview.**

batch these reports and submit them to an aggregation service, which performs the final step: (1) validating the reports, ensuring all parameters in authenticated data match and that no report is reused; (2) summing the attribution values; and (3) applying DP, adding noise (such as from a Laplace distribution with scale `maxValue/epsilon`) to protect individual users. The resulting *noised, aggregated conversion metrics* are then provided to *r1.ex* and *r2.ex*, which relay the ad-effectiveness comparison back to *shoes.ex*, helping it discern which of the colorful vs. black-and-white ads leads to higher revenue.

## 2.3 Privacy loss accounting with Cookie Monster

PPA enforces privacy using the individual differential privacy (individual DP) framework from the Cookie Monster paper [26], which tracks each user's privacy loss separately and optimizes for on-device attribution. This is a key departure from traditional DP, which maintains a single global guarantee across users. individual DP allows PPA to bound privacy loss more efficiently—based only on the actual contribution of a device to a query.

Within each browser, PPA enforces individual DP at the *epoch* level, dividing the impression stream into time intervals (e.g., a week), each with its own privacy budgets. Each device maintains an *Impression Store* to log impressions per epoch and a *Privacy Filter Store* to track per-epoch budgets. A *privacy filter* acts as the epoch's budget manager: it deducts privacy loss only if sufficient budget remains and only when data from that epoch contributes to a query; if depleted, it blocks further use of that epoch's data. Importantly, PPA maintains separate epoch-level privacy filters *per site*, a design choice that we show raises budget management questions.

Fig. 1(c) shows these internal components and illustrates how privacy loss is computed and enforced. When a conversion occurs (`measureConversion()`), the browser uses the Cookie Monster algorithm to: (1) retrieve all relevant impressions from the Impression Store, grouped by epoch; (2) compute *individual privacy loss per epoch*, using `value / maxValue * epsilon` if an epoch has at least one relevant impression, or zero otherwise; and (3) deducts this loss across all contributing epochs from the *conversion site's filters*, returning a null attribution if deduction fails, and the real one otherwise.

For example, in Fig. 1(b), epochs $e1$ and $e2$ each incur an individual privacy loss of `value / maxValue * epsilon = 0.3`, while traditional DP would charge the full `epsilon = 0.5` loss. Even better, epoch $e3$, which contains no relevant impressions, incurs *zero* individual privacy loss. Although this process is nominally part of `measureConversion()`, in practice it is deferred until `getReport()`: if no report is requested, no privacy loss is incurred. Fig. 1(c) shows the resulting filter state after *r1.ex* and *r2.ex* request their reports: assuming an initial filter capacity of 1, the conversion site *shoes.ex* retains 0.7 budget in $e1$ and $e2$, and the full 1 in $e3$. In contrast,

standard DP would leave only 0.5 in each epoch.

This example highlights how individual DP limits privacy loss based on actual contributions. To further understand this dynamic—which is significant for our own system's design—we introduce a stock-and-flow analogy that captures the behavioral pattern that individual DP induces in PPA.

### 2.4 Stock-and-flow pattern

A key informal argument for PPA's practicality, voiced in PATWG discussions, is that individual DP accounting naturally limits privacy consumption by tying it to *user actions on both impression and conversion sites*. Non-zero privacy loss arises only when both an impression (signifying a user visit to an impression site) and a conversion (a visit to a conversion site) are present. This induces a *stock-and-flow pattern*: *privacy stock* is created on impression sites as impressions are saved, and *privacy flow* is triggered on conversion sites when reports are requested over those impressions—*both gated by user actions*. PATWG discussions generally acknowledge that users who engage with more impression and conversion sites should incur more privacy loss—up to a limit, discussed next.

### 2.5 Global privacy filter

PPA acknowledges that relying solely on per-site filters risks exposing users to adversaries capable of coordinating API activity across multiple sites. Such behavior amplifies information gain from attribution, proportional to the number of sites involved. Since per-site filters impose no bound on this, PPA proposes "safety limits"—per-epoch global filters that span site boundaries—originally suggested by [21]. While the spec gives no detail on how to manage these filters—a gap this paper addresses (see next section)—our input has shaped the spec's guiding principles: (1) global filter capacities must be much larger than per-site budgets, by necessity; and (2) these filters should "remain inactive during normal browsing and [trigger] only under high-intensity use or attack" [20].

### 2.6 Foundational gaps

We identify two key gaps in PPA related to managing its two filter types–per-site and global–which we address in Gordon.

**Gap 1: Unclear semantics of per-site filters.** PPA adopts Cookie Monster's accounting model, which tracks privacy loss *per querier*, but is ambiguous about who counts as a querier in real-world deployments. For **single-advertiser queries**, PPA maps the querier to the conversion site—e.g., an intermediary requests a report on behalf of a specific advertiser like *shoes.ex*, and privacy loss is charged to that advertiser's budget. Yet intermediaries also receive these reports and may reuse them for their own purposes, raising the question of whether they too should be considered queriers. The ambiguity grows with PPA's planned support for **cross-advertiser queries**, where intermediaries aim to optimize across multiple advertisers (e.g., training models to choose the best ad for a given context). Since intermediaries directly benefit from such queries, PATWG plans to charge privacy loss against their own budgets. This blurs the boundary between client-serving and self-serving queries, complicating the semantics of per-site accounting and increasing the risk of report misuse. In PATWG discussions, the global filter is often cited as a fallback, offering clearer semantics even when per-site guarantees break down—but this introduces its own configuration and management challenges, which we explore next. This paper proposes changes to the PPA API, protocol, and terms of use to clarify per-site semantics and the assumptions under which they remain sound (§4.1).

**Gap 2: Lack of mechanisms to manage the global filter.** A critical yet under-specified part of PPA is the configuration and management of the global filter, a shared resource across all parties requesting reports from a browser. This raises two key challenges: (1) how to set its capacity to support benign workloads and (2) how to prevent malicious actors from depleting it—either to boost their own utility or to deny service to others (e.g., competitors). While per-site budgets cap consumption per domain, they offer weak protection, as domain names are cheap and easily acquired. PATWG-discussed mitigations range from requiring sites to register with a trusted authority to browser-side heuristics for identifying illegitimate use of the API. But site registration faces resistance from some industry participants for undermining the API's open nature while heuristics rely on notions of "legitimacy" that are hard to define, especially for a nascent API with no deployment history and potentially valuable, unforeseen use cases. For instance, should the number of invocations be limited? Over what period and to what value? Should access to device-side budgets be restricted? On what grounds? While discussion in PATWG continues, we argue that the group lacks a foundation—a minimal set of principled mechanisms with well-defined properties under clear assumptions—to guide browsers toward targeted, defense-in-depth strategies that are both protective and not over-constraining for the API. This paper contributes such a foundation, from the vantage point of PPA's internal privacy budget management (§4.2).

## 3 Gordon Overview

We address PPA's gaps by (1) clarifying the two distinct threat models that per-site and global guarantees address (§3.1) and (2) introducing Gordon to both restore the semantics of per-site filters and manage the global filter to support legitimate use while limiting abuse (§3.3). §3.2 introduces an example.

### 3.1 Threat model

PPA and Gordon share similar threat models. Users trust their OS, browser, and browser-supported aggregation services. They extend limited trust to first-party sites they visit intentionally—i.e., through *explicit actions* like direct navigations or clicks—granting them access to first-party data and cookies. Embedded intermediaries are not trusted at all, and no site—first-party or otherwise—is trusted with cross-site data.

As API designers, we must address two threat levels. The

first is **intended use**, which assumes well-intentioned actors. Our goal here is to make compliance easy through careful API design and well-defined semantics. In the security literature, such actors are termed *honest-but-curious*: they follow the protocol but aim to extract as much information as permitted. Because some rules cannot be enforced by protocol alone, the API must include terms of use to close this gap. We define honest-but-curious adversaries as those who respect both the protocol and its terms of use.

PPA's *per-site filters* are meant to provide strong privacy guarantees against individual honest-but-curious sites. However, ambiguities in the current API and the lack of formal terms of use leave these guarantees semantically underspecified. Gordon addresses these gaps directly.

The second level involves **adversarial use**, where actors subvert the protocol and terms of use to extract excessive user information or maximize query utility through unauthorized budget consumption. Per-site budgets offer some protection when queriers operate independently or with limited coordination, leveraging DP's compositionality. But they fail under *large-scale Sybil attacks*, where an adversary registers many fake domains to bypass per-site caps. For example, a malicious conversion site X may use automatic redirection to cycle through Sybils, each triggering a single-advertiser report that maxes out its respective filter—multiplying the user's privacy loss by the number of Sybils.

PPA's *global filter* is designed to mitigate large-scale Sybil attacks by enforcing a coarser-grained budget. However, it introduces a new vulnerability: *denial-of-service (DoS) depletion attacks*. A malicious actor can deliberately exhaust the global budget, blocking legitimate queries—either to boost their own utility or harm competitors. These attacks can mirror the Sybil strategies used against per-site filters. §4.2 gives example attacks to which PPA is currently vulnerable.

Gordon embeds resilience directly into privacy budget management to defend against DoS depletion. While this layer alone does not provide complete end-to-end protection, it establishes a strong foundation and clarifies what browsers must enforce to achieve it. Building on the stock-and-flow model from §2.4, Gordon assumes that under intended use, privacy consumption is driven by explicit user actions—such as navigations or clicks—on distinct content and conversion first-party sites. As long as benign usage adheres to this pattern, Gordon fulfills PPA's guiding principles for the global filter (§2.5): supporting normal workloads under benign conditions and degrading gracefully under attack.

For this graceful degradation to hold in practice, two assumptions must be enforced: (1) browsers can reliably distinguish intentional user actions from automatic navigations, and (2) malicious actors cannot easily induce large numbers of users to intentionally visit many distinct attacker-controlled domains. If these assumptions fail, Gordon still upholds its privacy guarantees, but its DoS resilience will diminish.

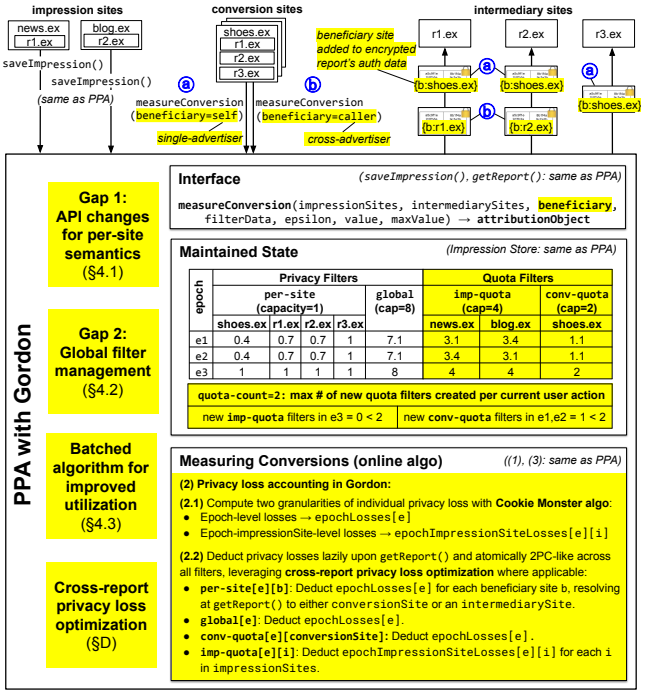



**Fig. 2. Gordon architecture**. Changes vs. PPA (Fig. 1(c)) in yellow.

## 3.2 Running example

We update the *shoes.ex* example to support *cross-advertiser queries*, a feature PPA plans to add soon. Our Gordon design anticipates this shift, which significantly impacts privacy budget management. To reflect this, we modify the example: *shoes.ex* contracts with *r1.ex* and *r2.ex* for ad placement and evaluation as before, but now *r1.ex* and *r2.ex* also optimize placements across advertisers and content sites. They will each therefore be interested in obtaining two encrypted reports for each conversion: one for single-advertiser measurement on behalf of *shoes.ex* and one for cross-advertiser optimization on their own behalf. Additionally, we introduce *r3.ex*, which focuses solely on single-advertiser measurements and specializes in cross-intermediary reporting, providing a complete view of *shoes.ex*'s ad performance across the two placement intermediaries *r1.ex* and *r2.ex*. *r3.ex* will require only one encrypted report for the single-advertiser measurement on *shoes.ex*'s behalf.

## 3.3 Gordon architecture

Fig. 2 shows Gordon's architecture, with proposed changes to PPA highlighted in yellow (relative to Fig. 1(c)). Gordon modifies all three layers of PPA: the interface, the privacy filter architecture, and how privacy loss is accounted for during conversion measurement and report requests. These changes span four major conceptual shifts (yellow boxes on the left).

**API changes for per-site semantics (Gap 1, §4.1).** We modify the API, protocol, and terms of use to eliminate ambiguity in budget attribution. Specifically, we introduce a *beneficiary site* parameter, authenticate it to the aggregation service, and enforce its use—both technically and contractually—so that

reports can only support the intended site's DP queries. These changes prevent intermediaries from misusing reports funded by conversion sites, restoring clear per-site semantics for parties that comply with the protocol and its terms. This addresses PPA's Gap 1 from §2.6.

**Global filter management (Gap 2, §4.2).** Without changing the API or protocol, we rework PPA's internal state to restore its intended stock-and-flow model of privacy loss, where user actions create "stock" at impression sites and trigger "flow" at conversion sites. Depletion attacks break this structure by automating flows or collapsing domain roles, draining the global filter without real user input. To defend against this, we introduce three quotas: one limits how much stock an impression site can create, another caps how much flow a conversion site can trigger, and a third bounds how many new sites can participate per user action. These quotas don't just limit indirect proxies (like API calls or intermediaries); they act directly on the core protected resource–the global filter–enforcing a privacy budget flow tightly coupled to actual user behavior and curbing adversarial misuse. We show that this leads to graceful degradation of utility for benign workloads under attack. This addresses PPA's Gap 2.

**Batched algorithm to improve utilization (§4.3).** Static quotas can underutilize the global privacy budget, especially when some impression sites see little demand while others face heavy load. To address this inefficiency, Gordon introduces a batched scheduling algorithm that collects unserved requests over a time interval and reallocates unused impression-site quota toward them at the end of each interval. This improves utilization without sacrificing resilience in our evaluation.

**Cross-report privacy loss optimization (§D).** Gordon introduces an optimization that reduces overcounting when multiple reports for the same conversion involve disjoint impression sets (e.g., across intermediaries). Since such reports reveal no more than the original attribution histogram to shared filters, Gordon accounts for them jointly across privacy and quota filters. We defer details to Appendix D, as they are not vital to understanding our main contributions. §D.1 illustrates how Fig. 2's filter states are calculated from it.

# 4 Detailed Design

## 4.1 API changes for per-site semantic (Gap 1)

We begin by addressing ambiguities in PPA's per-site filters, which aim to ensure privacy against honest-but-curious actors but currently fall short. Intermediaries like *r1.ex* and *r2.ex* can request reports on behalf of *shoes.ex*, causing PPA to deduct privacy loss from *shoes.ex* 's budget—even though *r1.ex* and *r2.ex* receive the reports and may reuse them for their own analytics. If those same intermediaries later run cross-advertiser queries (e.g., to train a model to choose between ads for *shoes.ex*, *toys.ex*, or *tvs.ex* based on content-site context), PPA charges their budgets directly. But when a single entity serves both roles, the line between client-serving and self-serving blurs. Even honest actors may be tempted to misuse reports charged to others. Conversion sites may also shard themselves into subdomains (e.g., *shoes-cart.ex*, *shoes-purchase.ex*) to extend their budget. Without clear constraints on report use, per-site accounting loses semantic integrity.

Gordon changes PPA's API, protocol, and terms of use to clarify the *beneficiary* for each DP query. In the **API**, we add a `beneficiary` parameter to `measureConversion()`. During `getReport()`, browsers resolve the beneficiary: to the conversion site for single-advertiser measurement, or to the requesting intermediary for cross-advertiser optimization. Privacy loss is then charged to the beneficiary's per-epoch filters, which are created as needed. Under the honest-but-curious model, we permit unrestricted filter creation. In the **protocol**, the beneficiary is included in the report's authenticated data, and aggregators are required to reject any batch with inconsistent beneficiaries. This blocks intermediaries from reusing reports charged to other clients' budgets. In the **terms of use**, we prohibit using DP results tied to one `beneficiarySite` to benefit another. Reports and results must remain siloed by beneficiary, even across shared infrastructure. This prohibits report-sharing among sharded identities (e.g., *shoes-cart.ex*, *shoes-purchase.ex*), cross-company collusion, and Sybil behavior (§3.1). Honest-but-curious sites will avoid these.

Together, these changes make per-site privacy loss accounting meaningful and enforceable under clear assumptions. PPA has already adopted the protocol update, and we plan to propose the API and policy changes shortly. Appendix A models beneficiary-based behavior, as required for subsequent proofs.

**Example.** In Fig. 2, *shoes.ex* issues two `measureConversion()` calls for a $60 purchase: (a) one for its own use (`beneficiary = self`), and (b) one for intermediaries (`beneficiary = caller`). In the first case, intermediaries like *r1.ex*, *r2.ex*, and *r3.ex* request reports on behalf of *shoes.ex*, which deduct from its budget. In the second, *r1.ex* and *r2.ex* request reports on their own behalf, triggering deductions from their own budgets. *r3.ex* does not participate and preserves its budget. Each encrypted report includes the beneficiary in authenticated data: `b:shoes.ex` for single-advertiser use (a); `b:r1.ex` and `b:r2.ex` for cross-advertiser reports (b).

## 4.2 Online global filter management (Gap 2)

With clarified semantics, per-site filters offer *strong privacy protection against honest-but-curious sites*, assuming tight configuration (e.g., capacity $\epsilon_{per-site} = 1$). But non-compliant behavior remains possible, making the global filter essential to safeguard against worst-case privacy loss—i.e., an adversary capable of accessing and combining results from *all sites*. To enforce both DP guarantees, Gordon implements a two-phase commit-like algorithm: data-driven (non-null) reports are returned only if they can be atomically funded by both per-site and global filters; otherwise, null reports are returned. Appendix B.2 formalizes this algorithm and its dual cross-granularity (individual) DP guarantee—a relevant property in

practice that, to our knowledge, has never been formalized.

A key challenge in managing the global filter is balancing competing goals: supporting benign workloads, resisting depletion attacks on this shared resource, and minimizing its guarantee to offer the strongest privacy protections that practical deployment can afford. We exemplify anticipated depletion attacks, then discuss limitations of existing defenses.

**DoS depletion attacks.** An adversary X may attempt to exhaust the global budget—either to boost their own utility or to disrupt others'. This threat already exists in PPA through single-advertiser queries and will grow with cross-advertiser support. To carry it out, X registers $s = \epsilon_{\text{global}}/\epsilon_{\text{per-site}}$ Sybil domains and distributes queries across them.

*Attack 1: Cross-advertiser reports.* X builds a site embedding the $s$ Sybil domains as intermediaries. When user $u$ visits, the site: (1) registers $s$ impressions with X as the conversion site and a different Sybil as intermediary; and (2) has each intermediary request a cross-advertiser report, exhausting its per-site budget. This drains the global budget for $u$. If many users visit X once per epoch, X can disrupt others' measurements for that epoch. If users continue arriving across epochs, X can sustain disruption, mounting a persistent attack with one popular site and just one visit per user per epoch. PPA isn't currently vulnerable, lacking cross-advertiser support. But a similar attack works using single-advertiser reports:

*Attack 2: Single-advertiser reports.* Here, the Sybils serve as both impression and conversion domains. When $u$ visits X, X auto-redirects $s$ times, switching domains to register impressions and trigger single-advertiser reports in lock step; each Sybil can register impressions for a different Sybil as the conversion site. As before, this depletes the global budget. While aggressive redirection may be heuristically flagged, redirection is too common for browsers to block outright.

*Attack 3: Single-advertiser reports, subtler version.* Upon user $u$'s visit, X (1) registers $s$ impressions with Sybil conversion sites, and (2) redirects once to load a new Sybil domain that requests a report. Reports use maximum attribution windows to draw budget across past epochs via impressions previously registered by X. If many users visit X's site roughly $s$ times during each epoch's data lifetime (typically months), X can sustain global budget depletion—again with one site but requiring multiple user visits.

**Limitations of existing defenses.** Some behaviors in these attacks clearly exceed reasonable use and should be disabled. (1) PPA is designed for cross-site measurement, so queries with identical `impressionSite` and `conversionSite` (Attack 1) should be disallowed. (2) A single user action shouldn't simultaneously register an impression and trigger conversion measurement of it—even across domains (Attack 1). (3) Excessive redirection (Attack 2) should be detectable. (4) Allowing an epoch's entire global budget to be exhausted in seconds is a fundamental flaw (Attacks 1 and 2). While these heuristics offer minimal protection, more principled defense is needed for subtler abuse like Attack 3.

In PATWG discussions, several mitigations have been proposed: restricting which sites receive per-site filters (e.g., via mandatory registration), rate-limiting API calls, or capping the number of sites granted filters per epoch. While potentially useful, these measures risk over-constraining a nascent, evolving workload. Mandatory registration could limit access to the API, undermining the web's openness. Hard limits on per-site impression counts are tricky: some sites show many ads, others few. The same goes for conversions, which may range from rare purchases to frequent landing-page visits. Capping intermediaries per conversion could constrain advertisers' ability to work with diverse partners. And if the API is repurposed beyond advertising—e.g., to measure engagement or reach—workload patterns may evolve further. Fixed constraints that seem reasonable today could stifle innovation or penalize legitimate new uses.

**Our approach: Enforce stock-and-flow.** We aim for defenses that make *minimal assumptions about workloads*, enabling browsers to provide strong protection without overly restricting the API. In §2.3, we introduce a stock-and-flow pattern for PPA's intended use: privacy loss is driven by explicit user actions—like navigations or clicks—across distinct impression and conversion domains. The above attacks break this pattern by automating flows and collapsing domain roles.

We restore the pattern via *quotas*: impression-site quotas cap stock creation, conversion-site quotas cap triggered flow, and a count-based limit bounds the number of new sites that can create the preceding quotas from a single user action. Unlike indirect metrics (e.g., API call frequency, number of intermediaries, or domains with per-site filters), our first two quotas operate directly on the protected resource: the global filter. Each represents a *share* of the global budget calibrated to a browser-defined "normal" workload. The third quota anchors the stock-and-flow pattern to explicit user action. Together, these quotas constrain adversaries to operate within the contours of normal workloads, preventing global budget drainage by a single site with limited user interaction.

**Gordon quota system.** Fig. 2 (yellow background) highlights the internal state maintained by Gordon to manage global privacy filters in PPA. Appendix B formalizes the system's behavior and proves its privacy and resilience properties. We use two types of quotas: (1) *quota filters*, `imp-quota` and `conv-quota`, which are implemented as DP filters—not for privacy accounting, but to regulate global filter consumption, a novel use in DP literature; (2) a standard *count-based quota* limiting the number of new quota filters created per user action.

The impression-site quota filter, `imp-quota`, is scoped per impression site and per epoch. It bounds the global privacy loss from flows that use stock created by impressions from that site. When site `i` first calls `saveImpression()` in epoch `e`, Gordon creates `imp-quota[e][i]`, with capacity set to a share of the global filter. This quota is consumed only if a

**"Normal" workload parameters:**
**M**: max # of impression sites in an epoch contributing to non-zero loss in epoch.
**N**: max # of conversion sites that request non-zero loss from an epoch.
**n**: max # of conversion sites that request non-zero loss from a single (epoch, impression site) pair.
**r**: max budget consumed by an intermediary's cross-advertiser queries on a single conversion site, as a fraction of the intermediary's $\epsilon_{\text{per-site}}$.

| Filter | Capacity configuration |
|---|---|
| Per-site filter | $\epsilon_{\text{per-site}}$: configuration parameter |
| Global filter | $\epsilon_{\text{global}} = \max(N, n \cdot M)(1+r)\epsilon_{\text{per-site}}$ |
| Impression-site quota | $\epsilon_{\text{imp-quota}} = n(1+r)\epsilon_{\text{per-site}}$ |
| Conversion-site quota | $\epsilon_{\text{conv-quota}} = (1+r)\epsilon_{\text{per-site}}$ |

**Tab. 1. Gordon filter configurations**.

later report matches an impression from `i` in that epoch—that is, if `i`'s stock is used.

The conversion-site quota filter, `conv-quota`, is scoped per conversion site and per epoch. It bounds global privacy loss from flows initiated by conversions on that site. `conv-quota [e][c]` is created when site `c`, in or after epoch `e`, first calls `measureConversion()` in a way that could incur non-zero loss in `e`. It is consumed on `getResult()`—i.e., a flow occurs.

Fig. 2 sketches Gordon's privacy loss accounting algorithm (box "Measuring Conversions"; full version in Appendix B.1). Per-epoch individual privacy losses are first computed using the Cookie Monster algorithm. Then, for each `getReport()`, Gordon attempts to deduct losses across relevant filters in an atomic transaction per epoch: success only alters state if all checks pass. A non-null report is returned only if checks succeed in all epochs. Relevant filters include the beneficiary's `per-site` filter, the `global` filter, the conversion site's `conv-quota`, and an `imp-quota` for each impression site with non-zero loss. To efficiently enforce impression-site quotas, we compute loss at the (epoch, impression site) level and charge it to the corresponding `imp-quota`. Although total quota capacities may exceed the global filter at any moment, Gordon's atomic checks ensure global budget is never breached.

Quota filters cap how much each first-party site contributes to global privacy consumption. In a world without automatic redirects—where every domain change reflects a user action—this would suffice to reestablish user-driven stock-and-flow. But redirects are pervasive, so we allow a bounded number of first-party domains to trigger new quota creation after a single explicit user action. This bound, `quota-count`, is configurable and expected to be small (e.g., 2 or 3). We also recommend disallowing a single domain from registering both an impression and a conversion on the same user action.

**Configuration to "normal" workload.** How should filters be configured to avoid disrupting benign workloads? We take three steps. First, we define four browser-adjustable parameters describing expected workload scale ($N$, $M$, $n$, $r$), defined in Table 1. Second, given these parameters and $\epsilon_{\text{per-site}}$, we express constraints the other capacities must meet to support this workload: $\epsilon_{\text{conv-quota}} \geq (1+r)\epsilon_{\text{per-site}}$; $\epsilon_{\text{imp-quota}} \geq n \cdot \epsilon_{\text{conv-quota}}$; $\epsilon_{\text{global}} \geq \max(N \cdot \epsilon_{\text{conv-quota}}, M \cdot \epsilon_{\text{imp-quota}})$. Third, we derive capacity formulas from these constraints, as shown in Table 1.

**Resilience to DoS depletion.** We prove the following:

**Theorem 1** (Resilience to DoS depletion (proof in B.3))**.** *Consider an adversary who manages to create $M^{adv}$ and $N^{adv}$ `imp-quota` and `conv-quota` filters, respectively. The maximum budget $\epsilon^{adv}_{global}$ that the adversary can consume from the global filter on a device $d$ is such that:*

$$\epsilon^{adv}_{global} \leq \min(M^{adv}\epsilon_{imp\text{-}quota},\ N^{adv}\epsilon_{conv\text{-}quota}).$$

This blocks *Attack 1*, where all impressions and conversions occur under one domain, yielding $M^{\text{adv}} = N^{\text{adv}} = 1$ and capping consumption at $\min(\epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$, far from depletion. The `quota-count` bound blocks *Attack 2*, where a single user visit triggers automatic redirection. This bound—small by design—limits how many quota filters can be created per user action, allowing only modestly more budget use than in Attack 1.

In general cases like *Attack 3*, an adversary who receives $U_{\text{adv}}$ interactions from user $u$ during epoch $e$ can create at most $M^{\text{adv}} + N^{\text{adv}} \leq$ quota-count $\cdot\, U^{\text{adv}}$. Our quotas ensure *graceful degradation* for benign workloads as a function of $U^{\text{adv}}$. We prove the following:

**Theorem 2** (Graceful degradation (proof here))**.** *Consider a adversary collecting $U^{adv}$ user actions on sites under their control for device $d$. Under the configuration of Table 1, the budget $\epsilon^{adv}_{global}$ that this adversary can consume from the global filter is upper-bounded by:*

$$\epsilon^{adv}_{global} \leq (1+r)\epsilon_{per\text{-}site} \times \frac{n}{1+n}\left(\texttt{quota-count} \times U^{adv}\right).$$

*Proof.* Thm. 1 implies the most efficient way to allocate the $\texttt{quota-count} \times U^{\text{adv}} = M^{\text{adv}} + N^{\text{adv}}$ filter creations available to the attacker is such that $M^{\text{adv}}\epsilon_{\text{imp-quota}} = N^{\text{adv}}\epsilon_{\text{conv-quota}}$, or $M^{\text{adv}} n(1+r)\epsilon_{\text{per-site}} = N^{\text{adv}}(1+r)\epsilon_{\text{per-site}}$. This yields $M^{\text{adv}} = \frac{1}{n+1}\texttt{quota-count} \times U^{\text{adv}}$ and $N^{\text{adv}} = \frac{n}{n+1}\texttt{quota-count} \times U^{\text{adv}}$. Applying Thm. 1 concludes the proof. □

## 4.3 Batched scheduling to improve utilization

Static quota partitioning can underutilize the global filter —even in benign scenarios. Suppose a device visits only two impression sites: *news.ex*, with many conversions, and *blog.ex*, with just one. The lone advertiser of *blog.ex* consumes up to its per-site filter, leaving much of *blog.ex*'s `imp-quota` unused, while the many advertisers of *news.ex* are bottlenecked by the `imp-quota` of *news.ex*. As a result, significant global budget remains idle, despite no added privacy risk from reallocating it to blocked *news.ex* advertisers. This limitation, flagged by PATWG participants, motivates our algorithmic solution that improves utilization while maintaining resilience to depletion.

Dynamically adjusting quotas based on observed demand would invite attacks, but we observe that if PPA supports *batched mode*—collecting requests over a period of time and servicing them gradually—we can make smarter scheduling decisions. In particular, we can gradually release unused

impression-site quota to support otherwise-blocked requests. The challenge is to (1) preserve some formal resilience guarantees, and (2) avoid scheduling decisions that depend on cross-epoch filter state, which would violate individual DP semantics. We present an algorithm that satisfies both constraints and shows significant utilization gains in evaluation.

**Algorithm.** Algorithm 1 outlines the approach (full version in Appendix C). We divide each epoch data's lifetime into $T$ *scheduling intervals* (e.g., one week). We extend the PPA API to support a *response time*—the interval after which a report is returned. For privacy, reports are only delivered at their response time; unscheduled requests yield encrypted null reports. Each interval has three phases: *(1) Initialization:* We *release* a portion $\epsilon_{\text{global}}/T$ of the global budget, adding it to any leftover from prior intervals. With both `imp-quota` and `conv-quota` filters active, we try allocating queued requests using this budget. TryAllocate() decides whether to attempt allocation for a request $r$ based solely on public metadata (as required for individual DP); if yes, it removes $r$ from the queue and applies all active filters. *(2) Online:* As requests arrive, and with both quotas on, we decide immediately based on the same process, whether to allocate or queue them. *(3) Batch:* At the interval's end, we disable `imp-quota` (keeping `conv-quota`), sort the queue via a max-min-fairness heuristic, and allocate requests one-by-one until no more succeed. TryAllocate() always attempts allocation if a request's response is due.

**Algorithm 1** Batched algorithm

```
1:  function SCHEDULE(queue)
2:    // (1) Initialization phase:
3:    ImpQuotasOn()
4:    ReleaseGlobalFilter()
5:    for r ∈ queue do TryAllocate(r)
6:    // (2) Online phase (new request r):
7:    if !TryAllocate(r) then queue.append(r)
8:    // (3) Batch phase:
9:    ImpQuotasOff()
10:   repeat
11:     Sort(queue)
12:     for r ∈ queue do
13:       if TryAllocate(r) then break
14:   until no new allocation
15:   return queue
```

**Sorting the queue.** Inspired by max-min fairness [18], we sort requests by the impression site with the least estimated budget consumption so far—based only on public metadata, per individual DP constraints. Within each site, requests are ordered by ascending requested privacy budget. For multi-site requests, we sort by site with lowest estimated budget usage.

**Resilience to DoS depletion.** During the online phase, both the `imp-quota` and `conv-quota` quotas are active, preserving the same resilience properties as in Thm. 2. In the batch phase, we lift the `imp-quota`, allowing any unused global filter capacity released so far to be reallocated. This helps support constrained benign workloads—such as some of *news.ex*'s advertisers—but also opens the door to adversarial exploitation. Nonetheless, consumption remains bounded by `conv-quota`, yielding the following bound on adversarial consumption:

$$\epsilon^{\text{adv}}_{\text{global}} \leq (1+r)\epsilon_{\text{per-site}} \times \texttt{quota-count} \times (U^{\text{adv}} - 1).$$

This bound is pessimistic: Appendix C.2 proves a tighter one, but real-world attacks are likely harder. Success would require (1) depleting budget ahead of legitimate online requests, (2) coordinating hybrid attacks across online and batch phases, and (3) defeating the scheduler's sorting mechanism, which favors low-budget and underrepresented impression sites.

## 4.4 Recommendations for PATWG

Gordon provides browsers with foundational building blocks for defending against DoS depletion attacks on PPA's global filter—though not an end-to-end solution. Operating within the budget management layer, our techniques offer built-in resilience independent of specific web attack vectors. However, they rest on assumptions—namely, that attackers cannot easily induce many users to visit many attacker-controlled domains—which browsers must enforce to achieve full protection. Our threat model (§3.1) leaves enforcement out of scope, but Gordon establishes a foundation to drive end-to-end solutions, which has so far been lacking in PATWG, hindering its progress. We conclude this section with a set of **Don'ts** and **Do's**, some addressing directions raised in PATWG.

**Don'ts:** (1) *Don't rate-limit API invocations*: This is not directly useful and risks stifling benign use cases. In Gordon, sites may register arbitrary impressions and conversions, and intermediaries may request any number of reports. The true limit is on how many distinct *domains* can act after a single user action. (2) *Don't limit the number of per-site filters*: These are meant to track privacy loss from honest-but-curious sites. They are not suitable levers for defending the global filter from malicious actors trying to deplete it. (3) *Don't require intermediary registration*: With proper budget management—by Gordon and by first parties managing their own quotas—intermediaries do not impact privacy or resilience guarantees. While Gordon leaves to future work the ability for first parties to control how intermediaries consume their quota, we believe this can be done rigorously, further reducing the need for intermediary registration with a PPA authority.

**Do's:** Focus on *detecting and disabling patterns of site sharding across domains*. For example, sites may attempt to shard themselves—e.g., routing each user interaction through a distinct domain—to inflate their quota access and deplete the global filter. While some level of sharding is inevitable (e.g., legitimate third-party integrations like shopping carts), aggressive self-sharding for DoS purposes should be explicitly prohibited. First, PPA should ban such behavior in its terms of use, which large, legitimate sites are likely to respect. Second, browsers should develop heuristics to detect noncompliant patterns and block offending sites from using the API. One possible signal is when a landing site frequently links to dynamically changing domains that invoke the API before returning users to the same main site. Third, Gordon's sorting algorithm could be extended to penalize suspicious-but-not-yet-blocked behavior. These are examples of concrete, actionable directions that PATWG can now pursue based on the resilience foundation provided by Gordon.

# 5 Prototype

We implement Gordon in two components: (1) pdslib, a general-purpose on-device individual DP library in Rust, and (2) its integration into Firefox's Private Attribution, a basic PPA prototype. **pdslib**: Gordon's core logic lives in pdslib, a Rust library for privacy budget management designed for broader individual DP use cases beyond advertising—e.g., location services in mobile apps. pdslib provides a generic interface: clients (sites or apps) register events (e.g., ad views, location visits) and request reports (e.g., attributions, model updates), receiving encrypted responses under strict privacy and isolation constraints. It implements all filters, quotas, privacy accounting, batching, and cross-report optimizations. Gordon is a PPA-specific instantiation—a 350 LoC shim atop pdslib's 2k LoC, specializing its generics to the PPA spec.

**Firefox integration:** We integrate pdslib and the Gordon shim into Firefox's Private Attribution, replacing its primitive report-count-based accounting with full privacy loss tracking (Firefox's PA lacks even Cookie Monster logic) [9]. Appendix E shows a Firefox extension dashboard we built to visualize filter and quota usage. We plan to open-source pdslib, the shim, and the integration to support PATWG and broader private-aggregation use cases.

# 6 Evaluation

We seek to answer the following questions: **(Q1)** What parameters define "normal" operation in the Criteo workload? **(Q2)** Do query error rates vary with different quota capacities? **(Q3)** Do quotas preserve low error rates for benign queries under DoS attacks? **(Q4)** Do quotas lead to under-utilization, and does our batching algorithm mitigate this?

## 6.1 Methodology

**Dataset.** We evaluate Gordon on CriteoPrivateAd [25], a dataset released by the Criteo ad-tech company, a PATWG participant, for the purpose of benchmarking private advertising systems. The dataset samples 30 days of production traffic using third-party cookies, with 104M impressions across 220k publisher sites (`publisher_id`) and 10k conversion sites (`campaign_id`, a good proxy for advertiser domains [25]). The data involves a *single intermediary*–Criteo itself.

Each impression includes contextual and user features, a daily-reset device ID, and attribution information indicating whether it led to a click, visit, or sale on a conversion site. This lets us reconstruct per-device, per-day conversion lists.

The dataset is impression-subsampled, not device-subsampled, so most devices have only one impression. Criteo provides the true device-level impression distribution and a resampling method to match it [25]. Using this, we construct a dataset with 4.6M impressions and 5.6M conversions across 1.4M devices, in which the median (resp. 90th percentile) device has 2 (resp. 6) impressions and 4 (resp. 16) conversions. We tune our algorithms and workload parameters on the first 10 days and report results on the remaining 20 days that we explicitly hold out for this evaluation.

**Benign workload process.** We consider a single-advertiser measurement scenario for benign queries. For each conversion, the advertiser–or Criteo acting on its behalf–invokes `measureConversion` and immediately after `getReport()` on the returned `attributionObject`, to request a report that attributes the conversion to the most recent relevant impression. Impressions are grouped into five buckets based on the `features_ctx_not_constrained_0` field, which is anonymized but we assume it represents region, device type, or user category. The end-to-end DP query produces a per-advertiser histogram estimating the number of conversions attributed to each bucket. We adopt $\mathbf{RMSRE}_\tau$—relative root mean square error truncated at $\tau$—to measure DP histogram error against the ground-truth histogram, following [2].

When requesting attribution reports, each advertiser must specify a privacy budget, denoted by $\epsilon$. We assign this budget in a way that mimics how a real advertiser might choose it—by aiming for a certain level of accuracy in their reports. First, we determine how many conversions each advertiser typically sees per day. To ensure reports can be reasonably accurate under differential privacy, we only include advertisers that average at least 100 conversions daily. There are 73 such advertisers in the dataset. Next, we decide how many conversions to include in each aggregation batch. We set this batch size to be either ten times the advertiser's daily average or 5,000 conversions—whichever is smaller. This ensures that advertisers produce roughly one report every 10 days, without aggregating on too small batches for low-volume advertisers. Finally, once we know how many conversions go into each batch, we choose the privacy budget $\epsilon$ so that the expected error in the report is about 5%, using a standard formula based on the Laplace mechanism and expected histogram statistics. This entire process is meant to reflect a realistic scenario, where advertisers select a privacy budget based on their volume and desired accuracy.

**Attack workload process.** To evaluate Gordon's resilience to DoS depletion attacks (specifically, Attacks 2 and 3 from §4.2), we inject a synthetic adversarial workload into real benign traffic. (Attack 1 is excluded, as the dataset only includes a single intermediary.) Our setup simulates an attacker who controls one popular impression site and 10 popular conversion sites that each redirects to 10 new Sybil domains per real user action. This corresponds to a highly permissive configuration of $quota - count = 10$ to ensure a strong attack.

We instantiate the attack as follows. We first introduce a *malicious impression site*, **X**, created by duplicating the most active impression site to ensure it interacts with many devices. Then, we identify the top 10 real conversion sites most frequently attributing conversions to impressions from the original version of X. For each, we generate 10 fake copies, simulating automatic redirections from each user action on the original conversion site to 10 new Sybil domains. Using

impression and conversion records as proxies for user activity, we duplicate the real traffic across the malicious sites: impressions between the original impression site and conversion sites are copied to generate fake impressions between X and the new malicious conversions. Similarly, we duplicate conversions from the original sites attributed to X, replaying them on the corresponding Sybil sites. We thus have 100 Sybils.

**Baselines.** We compare Gordon against two baselines. The first is **Cookie Monster**, which enforces only per-site filters. The second, **PPA**, extends Cookie Monster by adding a global filter, aligning with the current PPA draft specification.

**Defaults.** Unless otherwise stated, we use $\epsilon_{\text{per-site}} = 1$, $\epsilon_{\text{conv-quota}} = 1$, $\epsilon_{\text{imp-quota}} = 4$, and $\epsilon_{\text{global}} = 8$, reflecting our single-advertiser query workload (which implies $r = 0$) and a filter configuration derived from a "normal" workload. We define this workload using the 95th percentile values of $N$, $M$, and $n$ shown in Tab. 2 and detailed in the next section. Finally, since Criteo resets user IDs daily, we fix epoch duration to 1 day.

## 6.2 "Normal" workload parameters in Criteo (Q1)

Gordon's global and quota filter capacities are configured based on parameters intended to support a "normal" workload (Tab. 1). While our single-ad-tech dataset doesn't provide reliable values for these parameters, we present a methodology that browser vendors can apply once PPA is trialed at scale—and we illustrate this methodology on Criteo.

On a "training" dataset—the first 10 days of Criteo in our case—we can compute a distribution of $N, M, n$ values across devices, either by (1) running conversion attribution and computing their precise values as defined in Tab. 1, or (2) more efficiently, by computing upper bounds $\tilde{N} \geq N$, $\tilde{M} \geq M$, $\tilde{n} \geq n$ from the number of unique impression (resp. conversion) sites per device for $\tilde{M}$ (resp. $\tilde{N}$), and unique conversion sites per (device, impression-site) pair for $\tilde{n}$. We adopt the latter option for simplicity.

Tab. 2 shows these percentile values along with the corresponding global and impression-site quota capacities. Because our evaluation involves only single-advertiser queries, we force $r = 0$ and thus $\epsilon_{\text{conv-quota}} = \epsilon_{\text{per-site}}$. In more general settings, $r$ would also need to be set as a policy parameter.

| %ile | $\tilde{N}$ | $\tilde{M}$ | $\tilde{n}$ | $\epsilon_{\text{global}}$ | $\epsilon_{\text{imp-quota}}$ |
|---|---|---|---|---|---|
| p50 | 2 | 1 | 2 | 2 | 2 |
| p90 | 4 | 2 | 4 | 8 | 4 |
| p95 | 4 | 2 | 4 | 8 | 4 |
| p99 | 6 | 3 | 6 | 18 | 6 |
| p100 | 12 | 7 | 14 | 98 | 14 |

**Tab. 2. Criteo "normal" workload.**

Choosing quotas to support 100% of devices maximizes utility—since no quota-induced errors occur—but results in a very loose privacy budget, $\epsilon_{\text{global}} = 98$. A more balanced choice is the 95th percentile, which yields $\epsilon_{\text{global}} = 8$ and still avoids quota errors for the vast majority of devices. We validate the impact of this choice on accuracy in §6.3.

In general, selecting a percentile reflects a tradeoff between quota size (hence, utility) and the tightness of the global privacy guarantee $\epsilon_{\text{global}}$. This tradeoff depends on workload characteristics: Criteo's short epoch (1 day) and single-adtech scope suggest that real-world deployments—spanning multiple adtechs and longer epochs—will likely have higher $N, M, n$ values than in our table. For such workloads, stronger privacy guarantees may require adopting lower percentiles. We evaluate the effect of such tighter settings next.

## 6.3 Query errors under normal workload (Q2)

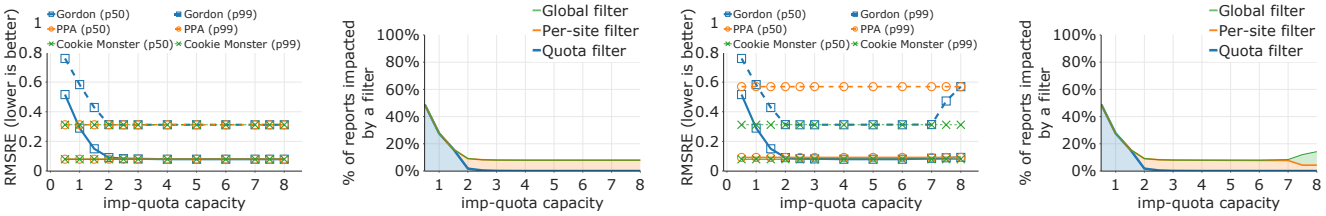
We vary $\epsilon_{\text{imp-quota}}$ and measure its effect on query error in the benign case. Fig. 3a shows the median and tail (99th percentile) RMSRE. Since Cookie Monster and PPA lack an `imp-quota`, their errors remain constant across $\epsilon_{\text{imp-quota}}$ values. Moreover, they show identical error: at the p95 setting in Tab. 2, $\epsilon_{\text{global}} = 8$ is high enough to eliminate any error from PPA's global filter —effectively reducing PPA to Cookie Monster. In contrast, Gordon's error rises at low $\epsilon_{\text{imp-quota}}$, as the quota forces some reports to be null, inducing error in query results. For $\epsilon_{\text{imp-quota}} \geq 2$, the filter no longer affects query error, suggesting that reasonably sized quotas preserve utility. The p95 values from Tab. 2 are sufficient to support normal operation—and are even conservative, since actual privacy loss may be lower than the worst-case upper bounds we use to configure $N$, $M$, and $n$.

Fig. 3b breaks down the sources of Gordon's error. RMSRE stems from two factors: DP noise (variance), and null reports due to filter/quota blocking (bias). We isolate the latter by counting how many and which filters were out-of-budget for each report. If multiple filters are exceeded, we break ties in this order: `per-site`, `global`, `conv-quota`, and `imp-quota`. We then compute the average fraction of affected reports per query. At $\epsilon_{\text{imp-quota}} = 1$, nearly a third of reports are blocked by imp-quota, explaining the high error in Fig. 3a. For $\epsilon_{\text{imp-quota}} \geq 2$, most blocked reports result from per-site filters—which also exist in Cookie Monster and PPA—explaining why Gordon's error converges to theirs.

## 6.4 Query errors under DoS attack (Q3)

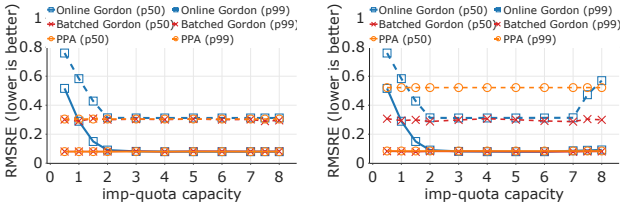
We evaluate Gordon's resilience under X's attack. Fig. 3c plots median and tail error for benign queries as a function of imp-quota capacity. Cookie Monster, lacking a global filter, is unaffected (but also lacks a global privacy guarantee). PPA includes a global filter but no protection from X, so its tail error rises sharply under attack. Gordon with a well-configured quota (e.g., $\epsilon_{\text{imp-quota}} = 4$ per p95 in Tab. 2) fully isolates honest queriers, matching Cookie Monster's error levels.

As before, a too-small quota harms utility even without an attacker. But under attack, too-large quota ($\epsilon_{\text{imp-quota}} \geq 7$) lets X drain the global budget, degrading benign-query utility—consistent with our theoretical bounds. Fig. 3d confirms these observations by showing the break-down of error causes across imp-quota settings. At low $\epsilon_{\text{imp-quota}}$, errors stem from honest reports blocked by the quota; the attack itself plays no role. At high $\epsilon_{\text{imp-quota}}$, errors arise from global filter depletion, permitted by a loose imp-quota.




**(a)** Error, normal workload **(b)** Error causes, normal workload **(c)** Error under attack **(d)** Error causes under attack

**Fig. 3. Quota system evaluation.** (a), (b): Query error and its root causes in a benign case. (c), (d): Benign-query error and root causes under attack.




**(a)** Error, normal workload **(b)** Error under attack

**Fig. 4. Batched algorithm evaluation.**

### 6.5 Batched algorithm evaluation (Q4)

Above error analyses show that low imp-quota capacities can degrade benign-query utility, with or without attack. Can the batched algorithm boost utilization while still protecting against attack? Fig. 4 explores this. In the **benign case**, Fig. 4a shows that batching substantially improves utilization: it sustains low tail error even at very low quotas (as low as $\epsilon_{\text{imp-quota}} = 0.5$), closely tracking PPA (and Cookie Monster), which impose no quotas. In the **attack case**, Fig. 4b shows that batching preserves low error for benign queries, unlike PPA, which is overwhelmed. The online algorithm performs well in the "safe" range ($2 \leq \epsilon_{\text{imp-quota}} \leq 7$), but fails beyond that as the attacker drains the global filter. In contrast, batching maintains low error across all $\epsilon_{\text{imp-quota}}$ values, mainly thanks to its max-min fairness-like sorting that spreads budget across queriers during batch scheduling. Thus, the batched algorithm improves utilization without sacrificing resilience.

## 7 Related Work

Our main contribution advances the **PPA API**—an emerging W3C standard poised to become the foundation for browser-based advertising measurement, and thus a critical part of the web's infrastructure. Gordon fills two key gaps in PPA: it clarifies the semantics of per-site filters and introduces a system for configuring and managing both these filters and the global filter. This system upholds strong privacy guarantees, supports benign workloads, and resists global filter depletion. Gordon also provides the basis of how budgeting should work for cross-advertiser queries, a planned PPA extension.

Among prior work on PPA and related APIs, the most relevant is Cookie Monster [26], which tracks privacy loss using per-epoch filters tied to individual queriers. Gordon goes further by managing these filters alongside the global filter and introducing a cross-report optimization that Cookie Monster lacks. Other foundational work includes Google's ARA papers [2, 6, 11], research on the MPC components of these APIs [4, 5, 3], now-retired proposals like IPA [14] and PAM [21], or Hybrid [13], which proposed several planned extensions to PPA. More broadly, there is related work on privacy-preserving ad targeting [29, 30].

Beyond ad measurement, this paper contributes to the broader challenge of **privacy budget management**, a crucial but understudied area in DP. While budget allocation *within* a single query is well-studied [17, 1, 23], budget management *across* queries from mutually distrustful parties is less explored. Relevant systems include those for global budget scheduling [18, 27, 15, 28], which inspire our batched scheduling approach but differ in key ways. For example, [23] considers fairness for a single batch, and [28] enforces DP for non-colluding analysts. [18] proposes a max-min-fair algorithm for allocating global budget across epochs, similar in spirit to our scheduler. Gordon departs from these systems in two ways. First, it operates under epoch-level *individual DP*, which rules out relying on cross-epoch budget information. Second, it supports adaptive, multi-task queriers and defends against DoS depletion—gaps unaddressed in prior work.

Gordon builds on the literature on privacy filters[24], particularly individual filters[10]—core DP primitives for adaptive composition and halting. Like prior work [19, 7, 16], we use filters to enforce DP. But we also repurpose them as quotas to limit consumption and provide isolation. A key contribution is our formalization of multi-granularity filter management—per-site and global—that supports simultaneous DP guarantees. These guarantees are essential in practice but have not been formalized in adaptive settings, nor has prior work shown how to manage them. We do both.

## 8 Conclusions

PPA is emerging as the foundation for browser-based advertising measurement—and thus a key part of the web's infrastructure. Yet critical technical gaps remain, creating a timely opportunity for academic work to advance the standard's deployability. Gordon addresses two such gaps: it restores clear semantics to per-site filters and introduces principled mechanisms for managing PPA's global filter, supporting expected workloads while defending against malicious depletion. These contributions lay a foundation for PPA to deliver strong privacy and robust utility—even in adversarial environments.

## References


[1] John M. Abowd et al. "The 2020 Census Disclosure Avoidance System TopDown Algorithm". In: *Harvard Data Science Review* Special Issue 2 (June 2022).

[2] Hidayet Aksu et al. "Summary Reports Optimization in the Privacy Sandbox Attribution Reporting API". In: *Proc. Priv. Enhancing Technol.* 2024.4 (2024), pp. 605–621. DOI: 10.56553/POPETS-2024-0132. URL: https://doi.org/10.56553/popets-2024-0132.

[3] James Bell et al. "Distributed, Private, Sparse Histograms in the Two-Server Model". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 307–321. ISBN: 9781450394505. DOI: 10.1145/3548606.3559383. URL: https://doi.org/10.1145/3548606.3559383.

[4] Henry Corrigan-Gibbs and Dan Boneh. "Prio: Private, Robust, and Scalable Computation of Aggregate Statistics". In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, Mar. 2017, pp. 259–282. ISBN: 978-1-931971-37-9. URL: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs.

[5] Hannah Davis et al. "Verifiable Distributed Aggregation Functions". In: *Proc. Priv. Enhancing Technol.* 2023.4 (2023), pp. 578–592. DOI: 10.56553/POPETS-2023-0126. URL: https://doi.org/10.56553/popets-2023-0126.

[6] Matthew Dawson et al. *Optimizing Hierarchical Queries for the Attribution Reporting API*. Comment: Appeared at AdKDD 2023 workshop; Final proceedings version. Nov. 27, 2023. arXiv: 2308.13510 [cs].

[7] David Durfee and Ryan M Rogers. "Practical Differentially Private Top-k Selection with Pay-what-you-get Composition". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[8] Hamid Ebadi, David Sands, and Gerardo Schneider. "Differential Privacy: Now It's Getting Personal". In: *Proceedings of the 42nd Annual ACM SIGPLAN - SIGACT Symposium on Principles of Programming Languages*. POPL '15: The 42nd Annual ACM SIGPLAN SIGACT Symposium on Principles of Programming Languages. Mumbai India: ACM, Jan. 14, 2015, pp. 69–81. ISBN: 978-1-4503-3300-9. DOI: 10.1145/2676726.2677005.

[9] *Experiment: Privacy-Preserving Attribution Measurement API*. https://github.com/mozilla/explainers/tree/main/ppa-experiment. 2024.

[10] Vitaly Feldman and Tijana Zrnic. "Individual Privacy Accounting via a Rényi Filter". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 28080–28091.

[11] Badih Ghazi et al. *On the Differential Privacy and Interactivity of Privacy Sandbox Reports*. 2024. arXiv: 2412.16916 [cs.CR].

[12] Ali Ghodsi et al. "Dominant resource fairness: fair allocation of multiple resource types". In: *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*. NSDI'11. Boston, MA: USENIX Association, 2011, pp. 323–336.

[13] *Hybrid Proposal*. https://github.com/patcg-individual-drafts/hybrid-proposal. 2024.

[14] *Interoperable Private Attribution (IPA)*. https://github.com/patcg-individual-drafts/ipa. 2022.

[15] Nicolas Küchler et al. "Cohere: Privacy Management in Large Scale Systems". In: *CoRR* abs/2301.08517 (2023). DOI: 10.48550/ARXIV.2301.08517. arXiv: 2301.08517. URL: https://doi.org/10.48550/arXiv.2301.08517.

[16] Mathias Lécuyer. *Practical Privacy Filters and Odometers with Rényi Differential Privacy and Applications to Differentially Private Deep Learning*. 2021. arXiv: 2103.01379 [stat.ML]. URL: https://arxiv.org/abs/2103.01379.

[17] Chao Li et al. "Optimizing linear counting queries under differential privacy". In: *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '10. Indianapolis, Indiana, USA: Association for Computing Machinery, 2010, pp. 123–134. ISBN: 9781450300339. DOI: 10.1145/1807085.1807104. URL: https://doi.org/10.1145/1807085.1807104.

[18] Tao Luo et al. "Privacy Budget Scheduling". In: *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. USENIX Association, July 2021, pp. 55–74. ISBN: 978-1-939133-22-9. URL: https://www.usenix.org/conference/osdi21/presentation/luo.

[19] Ryan McKenna et al. "AIM: an adaptive and iterative mechanism for differentially private synthetic data". In: *Proc. VLDB Endow.* 15.11 (July 2022), pp. 2599–2612. ISSN: 2150-8097. DOI: 10.14778/3551793.3551817. URL: https://doi.org/10.14778/3551793.3551817.

[20] *Privacy-Preserving Attribution: Level 1*. https://w3c.github.io/ppa/. 2024.

[21] *Private Ad Measurement (PAM)*. https://github.com/patcg-individual-drafts/private-ad-measurement. 2023.

[22] *Private Advertising Technology Working Group*. https://www.w3.org/groups/wg/pat/. 2024.

[23] David Pujol et al. "Budget sharing for multi-analyst differential privacy". In: *Proc. VLDB Endow.* 14.10 (June 2021), pp. 1805–1817. ISSN: 2150-8097. DOI:

10.14778/3467861.3467870. URL: https://doi.org/10.14778/3467861.3467870.

[24] Ryan Rogers et al. "Privacy odometers and filters: pay-as-you-go composition". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 1929–1937. ISBN: 9781510838819.

[25] Mehdi Sebbar et al. *CriteoPrivateAd: A Real-World Bidding Dataset to Design Private Advertising Systems*. 2025. arXiv: 2502.12103 [cs.CR]. URL: https://arxiv.org/abs/2502.12103.

[26] Pierre Tholoniat et al. "Cookie Monster: Efficient On-Device Budgeting for Differentially-Private Ad-Measurement Systems". In: *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*. SOSP '24. New York, NY, USA: Association for Computing Machinery, Nov. 15, 2024, pp. 693–708. ISBN: 9798400712517. DOI: 10.1145/3694715.3695965.

[27] Pierre Tholoniat et al. "DPack: Efficiency-Oriented Privacy Budget Scheduling". In: *Proceedings of the Twentieth European Conference on Computer Systems*. EuroSys '25. Rotterdam, Netherlands: Association for Computing Machinery, 2025, pp. 1194–1209. ISBN: 9798400711961. DOI: 10.1145/3689031.3696096. URL: https://doi.org/10.1145/3689031.3696096.

[28] Shufan Zhang and Xi He. "DProvDB: Differentially Private Query Processing with Multi-Analyst Provenance". In: *Proc. ACM Manag. Data* 1.4 (Dec. 2023). DOI: 10.1145/3626761. URL: https://doi.org/10.1145/3626761.

[29] Ke Zhong, Yiping Ma, and Sebastian Angel. "Ibex: Privacy-preserving Ad Conversion Tracking and Bidding". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 3223–3237. ISBN: 9781450394505. DOI: 10.1145/3548606.3560651. URL: https://doi.org/10.1145/3548606.3560651.

[30] Ke Zhong et al. "Addax: A fast, private, and accountable ad exchange infrastructure". In: *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 825–848. ISBN: 978-1-939133-33-5. URL: https://www.usenix.org/conference/nsdi23/presentation/zhong.

# A API changes for per-site semantic (Gap 1)

This section formalizes API changes to clarify the per-site semantics. Starting from Cookie Monster's formalism, we adapt it to capture Gordon's notion of beneficiaries. While this section does not present a standalone result, its formalism underpins the main theorems proved later and evoked in the body of the paper. We begin by mapping terminology between Cookie Monster and Gordon (§A.1), then outline modeling restrictions (§A.2), after which we define Gordon's data (§A.3) and query models (§A.4), and conclude with sensitivity analyses needed in subsequent sections (§A.5).

## A.1 Sites and roles

In this paper, we align with the terminology of PPA, which differs from that of Cookie Monster. Moreover, we make sites appear explicitly in the data and query model. We take a set of sites $\mathcal{S}$ (*e.g.,* domain name). The same site can appear under different roles:

- impression site: site where an impression occurs (publisher in Cookie Monster)
- conversion site: site where a conversion occurs (advertiser in Cookie Monster)
- beneficiary site: site that receives the results of a DP query (querier in Cookie Monster)

## A.2 Modeling restrictions

We focus here on single-beneficiary settings, captured by the following assumptions, which we then relax in our study of the cross-report optimization in §D:

- The set of public events for a beneficiary site $b$ is the set of all conversions where $b$ appears as beneficiary site. In other words, we hardcode $P = C_b$. This prevents publisher reports or certain adtech optimization queries, that were technically supported by Cookie Monster.
- Each report has a single beneficiary site: either the conversion site (measurement report) or another site (optimization report). In this formalism, a conversion site can register the same conversion data multiple times with different beneficiary sites if it wants to execute both measurement and optimizations.

## A.3 Data model

A database $D$ is a set of device-epoch records where each record $x = (d, e, F) \in \mathcal{X} = \mathcal{D} \times \mathcal{E} \times \mathcal{P}(\mathcal{S} \times \mathcal{I} \cup \mathcal{S} \times \mathcal{S} \times C)$ contains a device $d$, an epoch $e$ and a set of impression and conversion events $F$. Each event $f \in F$ contains the site (impression site $i$ or conversion site $c$) where the event occurred: $f = (i, \text{imp}) \in \mathcal{S} \times \mathcal{I}$ or $f = (c, b, \text{conv}) \in \mathcal{S} \times \mathcal{S} \times C$. Additionally, conversions contain the beneficiary site $b$ that will receive the conversion report.

**Definition 1** (Filter ($\mathcal{F}_x$))**.** *For each device-epoch record $x = (d, e, F)$, we maintain several types of privacy filters:*

- $\mathcal{F}_x^{nc[b]}$*: Non-collusion filter for beneficiary site b, with capacity* $\epsilon_{nc[b]}$
- $\mathcal{F}_x^{c}$*: Collusion filter with capacity* $\epsilon_c$*.*
- $\mathcal{F}_x^{q\text{-}conv[c]}$*: Conversion site quota filter for site c, with capacity* $\epsilon_{conv\text{-}quota}$
- $\mathcal{F}_x^{q\text{-}imp[i]}$*: Impression site quota filter for site i, with capacity* $\epsilon_{imp\text{-}quota}$

*Each filter maintains a state of consumed budget and provides operations:*

- canConsume($\epsilon$)*: Returns* TRUE *if the filter can accommodate additional privacy loss $\epsilon$. In particular, letting $\mathbb{I}_{pass}$ be how it's defined in Lem. 1,*
  - *For a per-site filter, global filter, or* `conv-quota` *filter, return* TRUE *iff it satisfies,*

$$\epsilon_x^t \le \epsilon_{\text{initial}} - \sum_{k \in [t-1]} \epsilon_x^k \cdot \mathbb{I}_{\text{pass}}[k], \tag{1}$$

  *where $\epsilon_{\text{initial}}$ is the respective initialized privacy budget of the global filter and conversion-site quota filters in Line* **??** *of Alg. 3, and $\epsilon_x^t$ is the privacy loss at step k from the corresponding filter (epoch-level budget consumption is the same as "ComputeIndividualBudget" defined in appendix D of [26]).*
  - *For an impression-site quota filter, return* TRUE *iff it satisfies*

$$\epsilon_x^{\text{i},t}[i] \le \epsilon_{imp\text{-}quota} - \sum_{k \in [t-1]} \epsilon_x^{\text{i},k}[i] \cdot \mathbb{I}_{pass}[k], \tag{2}$$

  *where $\epsilon_x^{\text{i},t}[i]$ is the site-level privacy loss at step k from the* `imp-quota` *filter (site-level budget consumption in Alg. 5).*
- tryConsume($\epsilon$)*: Deducts privacy loss $\epsilon$ from the filter's remaining capacity iff* canConsume *on all relevant filters return* TRUE*, and the amount consumed is by basic compositions of pure DP filters.*

## A.4 Query model

**Definition 2** (Attribution function, adapted from Cookie Monster)**.** *Fix a set of relevant impression sites $\mathbf{i}_A \subset \mathcal{S}$ and a set of* impressions *relevant to the query $F_A \subset \mathbf{i}_A \times \mathcal{I}$. Fix $k, m \in \mathbb{N}^*$ where k is a number of epochs. An* attribution function *is a function $A : \mathcal{P}(\mathcal{I})^k \to \mathbb{R}^m$ that takes k event sets $F_1, \ldots, F_k$ from k epochs and outputs an m-dimensional vector $A(F_1, \ldots, F_k)$, such that only* relevant events *contribute to A. That is, for all $(F_1, \ldots, F_k) \in \mathcal{P}(\mathcal{I})^k$, we have:*

$$A(F_1, \ldots, F_k) = A(F_1 \cap F_A, \ldots, F_k \cap F_A). \tag{3}$$

**Definition 3** (Report identifier and attribution report, same as Cookie Monster)**.** *Fix a domain of report identifiers $\mathbb{Z}$. Consider a mapping $d(\cdot)$ from report identifiers R to devices $\mathcal{D}$ that gives the device $d_r$ that generated a report r.*

*Given an attribution function A, a set of epochs E and a report identifier $r \in \mathbb{Z}$, the* attribution report *$\rho_{r,A,E}$, or $\rho_r$ for short, is a function over the whole database D defined by:*

$$\rho_r : D \in \mathbb{D} \mapsto A(D_{d_r}^E). \tag{4}$$

**Definition 4** (Query, same as Cookie Monster)**.** *Consider a set of report identifiers $R \subset \mathbb{Z}$, and a set of attribution reports $(\rho_r)_{r\in R}$ each with output in $\mathbb{R}^m$. The* query *for $(\rho_r)_{r\in R}$ is the function $Q : \mathbb{D} \to \mathbb{R}^m$ is defined as $Q(D) := \sum_{r\in R} \rho_r(D)$ for $D \in \mathbb{D}$.*

## A.5 Sensitivity analyses

The Cookie Monster paper analyzes global and individual sensitivities of queries at device-epoch level (§C of [26]). In Gordon, we additionally need such analyses at device-epoch-site-level, as some of our quota systems operate at device-epoch-site granularity (notably the impression-site quotas)–see Alg. 5. This section provides analysis for these sensitivities: Thm. 3 and give the global sensitivities of generating reports and answering queries, respectively. Thm. 7 and Thm. 8 give the individual sensitivity versions.

We first start global sensitivity analysis of generating reports.

**Theorem 3** (Global sensitivity of reports per epoch-site)**.** *Fix a report identifier $r$, a device $d$, a set of epochs $E_r = \{e_1^{(r)}, \ldots, e_k^{(r)}\}$, and a set of sites $I_r^{(e)} = \{i_1^{(e)}, \ldots, i_{m_e}^{(e)}\}$ for each epoch $e \in E_r$. Let $A(\cdot)$ be the attribution function of interest, so that the corresponding report is $\rho : D \mapsto A(D_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}})$. We have $\Delta(\rho)$*

$$= \max_{\substack{i\in[k],e:=e_i^{(r)},j\in[m_e],e'=e_k^{(r)} \\ \mathcal{F}:=(F_{1,1},\ldots,F_{k,m_{e'}})\subseteq(\mathcal{S}\times\mathcal{S}\times\mathcal{C}\cup\mathcal{S}\times\mathcal{I})^k}} \|A(\mathcal{F} : F_{i,j} := \emptyset) - A(\mathcal{F})\|_1. \tag{5}$$

*Proof.* Fix a report $\rho$. Let $k$ be $|E_r|$. Then, for $i \in [k]$, we enumerate through the epochs in $E_r$, and call the $i$-th epoch $e_i$. By definition of global sensitivity:

$$\Delta(\rho) = \max_{D,D'\in\mathbb{D}:\exists x\in\mathcal{X},D'=D+x} \|\rho(D) - \rho(D)\|_1, \tag{6}$$

from which we expand what $\rho$ function does:

$$\Delta(\rho) = \max_{\substack{D,D'\in\mathbb{D}: \\ \exists x\in\mathcal{X},D'=D+x}} \|A(D_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}}) - A((D')_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}})\|_1 \tag{7}$$

$$= \max_{\substack{D,D'\in\mathbb{D}: \\ \exists x=(d,e,F)\in\mathcal{X}: \\ e\in E_r,D'=D+x, \\ \forall(i,imp)\in F:i\in I_r^{(e)}}} \|A(D_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}}) - A((D')_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}})\|_1, \tag{8}$$

because for $(d', e', F) \in \mathcal{X}$ with $d' \neq d$ or $e' \notin E_r$, or any $(i, imp) \in F$ where $i \notin I_r^{(e)}$, they would not be considered as in the computation of $A(\cdot)$ and $A(D_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}}) = A((D')_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}})$.

Next, we show that the following sets are equal:

- $\{(D_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}}, (D')_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}}) : D, D' \in \mathbb{D} : \exists x = (d, e, F) \in \mathcal{X} : e \in E_r, D' = D + x, \forall(i, imp) \in F : i \in I_r^{(e)}\}$.
- $\{(\mathcal{F} : F_{i,j} = \emptyset, \mathcal{F}) : i \in [k], e := e_i^{(r)}, j \in [m_e], e' = e_k^{(r)}, \mathcal{F} := (F_{1,1}, \ldots, F_{k,m_{e'}}) \subseteq (\mathcal{S} \times \mathcal{S} \times \mathcal{C} \cup \mathcal{S} \times \mathcal{I})^k\}$

We show that for all instances on the one side, there exists a corresponding instance on the other. In one direction, take any tuple from the first set, where $(D')_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}} = D_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}} + x$, for some $x = (d, e, F)$. Firstly, the $x$ satisfies $d = d_r$, $e \in E_r$, and $\forall f = (i, imp) \in F, i \in I_r^{(e)}$. Therefore, we construct $\mathcal{F}$ as follows:

$$F_{i,j} = \{f \in F : f = (j, imp) \text{ or } (c, q, conv) \text{ is relevant to site } j\}.$$

Note that every $F_{i,j}$ is set independently from $x$, except for the one that contains the events for site $j_0$ such that $F$ contains either $(j_0, imp)$ or $(c, q, conv)$ that is associated with $j_0$, in epoch $i_0$ such that $e_{i_0}^{(r)} = e \in E_r$. Since $x \notin D$, we know that $F_{i_0,j_0}$ is set to $\emptyset$ on the side corresponding to $D$, and it is set to contain all events related to $j_0$ in epoch $i_0$ on the side corresponding to $D'$. That is, the corresponding instance in the second set would be $(\mathcal{F} : F_{i_0,j_0} = \emptyset, \mathcal{F})$.

Conversely, let $(\mathcal{F} : F_{i,j} = \emptyset, \mathcal{F})$ be from the second set. Then, we know the set of events corresponding to site $j$ in epoch $i$ is empty for the first element. So, we let $x := (d, e, F) \in \mathcal{X}$, where $d$ is the same devices as was fixed, $e = e_i^{(r)}$, and $F$ contains events related to site $j$ in epoch $i$. Then, we let $(D, D') \in \mathbb{D} \times \mathbb{D}$, where $D' = D + x$ and everything in $D$ and $D'$ other than $x$ corresponds to the rest of the $F_{i,j}$ that are the same in both $\mathcal{F}$ and $\mathcal{F} : F_{i,j} = \emptyset$. As such, $(D_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}}, (D')_d^{E_r,\{I_r^{(e)}\}_{e\in E_r}})$ is the corresponding element in the first set.

Finally, we substitute the first set in equation line (8) by the second set accordingly, and the resulting max should be equal as the two sets are equivalent, and the resulting form will be the one claimed. □

Immediately, we can acquire the following corollary about global sensitivity of generating reports, with attribution functions that have $m$-dimensional outputs as they are in our setting.

**Corollary 4.** *Suppose the attribution function $A(\cdot)$ has an $m$-dimensional outputs, and each dimension of any $\mathcal{F} \subseteq (\mathcal{S} \times \mathcal{S} \times \mathcal{I} \cup \mathcal{S} \times \mathcal{C})^k$ satisfy $\forall i \in [m]$, $A(\mathcal{F})_i \in [0, A^{\max}]$, then we have $\Delta(\rho) \leq mA^{\max}$.*

*Proof.* Given the form in Thm. 3, we know $\Delta(\rho)$ equals the max over $\|A(\mathcal{F} : F_{i,j} = \emptyset) - A(\mathcal{F})\|_1 = \sum_{i=1}^m |A(\mathcal{F} : F_{i,j} = \emptyset)_i - A(\mathcal{F})_i|$. But, we know for each dimension, $A(\mathcal{F}_i) \in [0, A^{\max}]$, so the absolute value of the difference in each dimension is $\in [0, A^{\max}]$. Thus, the sum over $m$ dimensions is $\|A(\mathcal{F} : F_{i,j} = \emptyset) - A(\mathcal{F})\|_1 \leq mA^{\max}$. □

Next, we analyze the global sensitivity of answering queries.

**Theorem 5** (Global sensitivity of queries per epoch-site)**.** *Let $R$ be the set of report identifiers relevant to a query $Q$. Then, we write $(\rho_r, d_r, E_r)_{r\in R}$ as the reports, devices, and epoch windows corresponding to each $r \in R$. Then, we have*

$$\Delta(Q) \le \max_{(d,e,F)\in\mathcal{X}} \sum_{\substack{r\in R: d=d_r, e\in E_r, \\ \exists i\in I_r^{(e)}: F \text{ contains events related to } i}} \Delta(\rho_r).$$

*Proof.* By definition of global sensitivity,

$$\Delta(Q) = \max_{D,D'\in\mathbb{D}:\exists x\in\mathcal{X}, D'=D+x} \|Q(D) - Q(D')\|_1 \tag{9}$$

$$= \max_{x\in\mathcal{X}} \max_{D,D'\in\mathbb{D}:D'=D+x} \|Q(D) - Q(D')\|_1. \tag{10}$$

Let $x = (d, e, F) \in \mathcal{X}$, where $F$ contains events associated with site $j$. For $r \in R$ such that $d \neq d_r$ or $e \notin E_r$, or $F$ contains events related to site $i \notin I_r^{(e)}$, we have $\rho(D) = \rho(D')$. So, by applying triangle-inequality in the mean time:

$$\|Q(D) - Q(D')\|_1 \le \sum_{r\in R: d=d_r, e\in E_r, i\in I_r^{(e)}} \|\rho_r(D) - \rho_r(D')\|_1 \tag{11}$$

$$\le \sum_{r\in R: d=d_r, e\in E_r, i\in I_r^{(e)}} \Delta(\rho_r), \tag{12}$$

where the second inequality is by definition of $\Delta(\rho_r)$. Note that this bound is independent from the choice of $D$ and $D'$, so we take the max over $(d, e, i)$ and get

$$\Delta(Q) \le \max_{d,e,i} \sum_{r\in R: d=d_r, e\in E_r, i\in I_r^{(e)}} \Delta(\rho_r).$$

But, the choice of $i$ can be used to set $F$ accordingly, so, equivalently, this suffices to prove the theorem. □

We can tighten the upper bound in the preceding theorem if we additionally require each device-epoch-site to participate in at most one report.

**Corollary 6.** *If each device-epoch-site participates in at most one report, then $\Delta(Q) = \max_{r\in R} \Delta(\rho_r)$.*

*Proof.* Since each device-epoch-site participates in at most one report, $\sum_{r\in R: d=d_r, e\in E_r, i\in I_r^{(e)}} \Delta(\rho_r) = \Delta(\rho_r)$. So,

$$\Delta(Q) \le \max_{(d,e,F)\in\mathcal{X}} \Delta(\rho_r).$$

But then, $\Delta(Q)$ must match the sensitivity of one of the $r$'s in this case, so the inequality is tight: $\Delta(Q) = \max_{r\in R} \Delta(\rho_r)$. Particularly, $\forall r$, $\exists D \sim D'$, such that $\|\rho_r(D') - \rho_r(D)\|_1 = \Delta(\rho_r)$. □

Next, we shift our gears to individual sensitivity analyses. We first analyze the individual sensitivity of generating reports.

**Theorem 7** (Individual sensitivity of reports per epoch-site)**.** *Fix a report identifier $r$, a device $d_r$, a set of epochs $E_r = \{e_1^{(r)}, \ldots, e_k^{(r)}\}$, a set of sites $I_r^{(e)} = \{i_1^{(e)}, \ldots, i_{m_e}^{(e)}\}$ for each epoch $e \in E_r$ , an attribution function $A$ with relevant events $F_A$, and the corresponding report $\rho : D \mapsto A(D_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}})$. Fix a device-epoch record $x = (d, e, F) \in \mathcal{X}$, where $F \subseteq \mathcal{S}\times\mathcal{S}\times\mathcal{C} \cup \mathcal{S}\times\mathcal{I}$, so that $x_i = (d, e, F_i)$ is the projection where $F_i$ contains only events related to site $i$.*

*If the report requests a single epoch $E_r = \{e_r\}$ as well as a single site in the one epoch, $I_r^{(e_r)} = \{i_r\}$, then we have:*

$$\Delta_{x_i}(\rho) = \begin{cases} \|A(F_i) - A(\emptyset)\|_1 & \text{, if } d = d_r, e = e_r \text{ and } i = i_r \\ 0 & \text{, otherwise.} \end{cases} \tag{13}$$

*In particular, it should be intuitive to see why $\mathcal{F}$ is a single $F_i$ in the first case, because we only have one epoch which contains one site, so there should be only one set of events corresponding to the one site in the one epoch.*

*Otherwise, either $|E_r| \ge 2$ or $|I_r^{(e)}| \ge 2$ for some $e \in E_r$, or both, and so we have:*

$$\Delta_{x_i}(\rho) \le \begin{cases} \Delta(\rho) & \text{, if } d = d_r,\ e \in E_r,\ i \in I_r^{(e)} \text{ and } F_i \cap F_A \neq \emptyset \\ 0 & \text{, otherwise.} \end{cases} \tag{14}$$

*Proof.* Fix a report $\rho$ and $x_i = (d, e, F_i) \in \mathcal{X}$. Consider any $D, D' \in \mathbb{D}$ such that $D' = D + x_i$. We have $\rho(D) = A(D_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}})$ and $\rho(D') = A((D')_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}})$.

- First, if $d \neq d_r$, $e \notin E_r$, or $i \notin I_r^{(e)}$, then $(D')_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}} = D_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}}$. Hence, $\|\rho(D) - \rho(D')\|_1 = 0$ for all such $D, D'$, which implies $\Delta_{x_i}(\rho) = 0$.
- Next, suppose that the report requests a single epoch $E_r = \{e_r\}$ with a single site $I_r^{(e_r)} = \{i_r\}$:
  - If $d = d_r$, $e = e_r$, and $i = i_r$, then since $D + x_i = D'$, we must have $(d_r, e_r, F_i) \notin D$, and thus $D_{d_r}^{e_r, i_r} = \emptyset$. On the other hand, $(D')_{d_r}^{e_r, i_r} = F_i$ (restricted to events relevant to site $i_r$). Thus, $\|\rho(D) - \rho(D')\|_1 = \|A(F_i) - A(\emptyset)\|_1$.
  - If $d \neq d_r$, $e \neq e_r$, or $i \neq i_r$, then $(d, e, F_i)$ doesn't change the outcome and $(D')_{e_r}^{i_r} = D_{e_r}^{i_r}$. Hence, $\|\rho(D) - \rho(D')\|_1 = 0$.
- Now, suppose that the report requests either an arbitrary range of epochs $E_r$ each of whom has at least one site, or a single epoch that has multiple sites $I_r^{(e_r)}$:
  - If $d \neq d_r$, $e \notin E_r$, or $i \notin I_r^{(e)}$, then $A((D')_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}}) = A(D_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}})$, i.e., $\|\rho(D') - \rho(D)\|_1 = 0$.
  - If we have $d = d_r$, $e = e_j^{(r)} \in E_r$, and $i \in I_r^{(e)}$, but $F_i$ is simply not related to the attribution request, i.e. $F_i \cap F_A = \emptyset$. Then, by definition of $F_A$, we have $A((D')_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}}) = A(D_{d_r}^{E_r, \{I_r^{(e)}\}_{e\in E_r}})$, i.e., $\|\rho(D) - \rho(D')\|_1 = 0$.
  - Otherwise, it must be the case that $d = d_r$, $e = e_j^{(r)} \in E_r$, $i \in I_r^{(e)}$ and $F_i \cap F_A \neq \emptyset$ and there are events in

the intersection that is related to some site $i$ in epoch $e$, so we have:

$$\|\rho(D) - \rho(D')\|_1 = \|A(\mathcal{F} : F_{j,i} = \emptyset) - A(\mathcal{F})\|_1, \quad (15)$$

where $j$ is the index of epoch $e$ in $E_r$, and $F_{j,i}$ represents the relevant events for site $i$ in epoch $e_j^{(r)}$.

The first two cases are independent over choices of $D \sim D'$, so taking the max over such choices still gives $\Delta_{x_i}(\rho) = 0$. Unfortunately, the third identity does depend on the choice of $D \sim D'$, and taking the max only gives the general definition of global sensitivity, in the worst case. Particularly,

$$\Delta_{x_i}(\rho) = \max_{\mathcal{F}=\{F_{j,i}\}:F_{j,i}\subseteq\mathcal{S}\times\mathcal{S}\times\mathcal{C}\cup\mathcal{S}\times\mathcal{I}} \|\rho(D) - \rho(D')\|_1 \quad (16)$$

$$= \max_{\mathcal{F}=\{F_{j,i}\}:F_{j,i}\subseteq\mathcal{S}\times\mathcal{S}\times\mathcal{C}\cup\mathcal{S}\times\mathcal{I}} \|A(\mathcal{F} : F_{j,i} = \emptyset) - A(\mathcal{F})\|_1 \quad (17)$$

$$\leq \Delta(\rho), \quad (18)$$

where the last inequality is tight due to the definition of global sensitivity and Thm. 3, for the worst case choice of $x_i$ in general.

□

Finally, we analyze the individual sensitivity of answering queries.

**Theorem 8** (Individual sensitivity of queries per device-epoch-site)**.** *Fix a query $Q$ with corresponding report identifiers $R$ and reports $(\rho_r)_{r\in R}$. Fix a device-epoch-site record $x_i = (d, e, F_i) \in \mathcal{X}$, where $F_i$ contains only events related to site $i$. We have:*

$$\Delta_{x_i}(Q) \leq \sum_{r\in R} \Delta_{x_i}(\rho_r) \quad (19)$$

*In particular, if $x_i$ participates in at most one report $\rho_r$, then $\Delta_{x_i}(Q) = \Delta_{x_i}(\rho_r)$.*

*Proof.* Take $D, D' \in \mathcal{D}$ such that $D' = D + x_i$. By the triangle inequality:

$$\Delta_{x_i}(Q) = \max_{D'=D+x_i\in\mathcal{D}} \|Q(D) - Q(D')\|_1 \quad (20)$$

$$= \max_{D'=D+x_i\in\mathcal{D}} \left\| \sum_{r\in R} \rho_r(D) - \rho_r(D') \right\|_1 \quad (21)$$

$$\leq \sum_{r\in R} \max_{D'=D+x_i\in\mathcal{D}} \|\rho_r(D) - \rho_r(D')\|_1 \quad (22)$$

$$\leq \sum_{r\in R} \Delta_{x_i}(\rho_r), \quad (23)$$

where the last inequality is by the definition of individual sensitivity.

When $x_i$ participates in at most one report $\rho_{r_0}$, we have $\Delta_{x_i}(\rho_r) = 0$ for all $r \neq r_0$. Therefore, $\Delta_{x_i}(Q) \leq \Delta_{x_i}(\rho_{r_0})$. This inequality is tight because there exists a pair $(D^*, D'^*)$ with $D'^* = D^* + x_i$ such that $\|\rho_{r_0}(D'^*) - \rho_{r_0}(D^*)\|_1 = \Delta_{x_i}(\rho_{r_0})$, and for this pair, $\|Q(D^*) - Q(D'^*)\|_1 = \Delta_{x_i}(\rho_{r_0})$. □

# B Global Filter Management (Gap 2)

## B.1 Algorithm

Gordon manages per-site and global privacy filters using the quota mechanisms described in §4.2. Alg. 2 presents an abstract model of Gordon's operation, capturing how it answers beneficiary queries sequentially. The function `AnswerQuery` corresponds to a beneficiary collecting a batch of reports, submitting them to an aggregation service, and receiving a noisy result.

Alg. 3 depicts the functionality triggered on receiving a report request (i.e., `measureConversion()` and `getReport()`). Gordon checks and consumes budget from the relevant filters and prunes the resulting report based on filter status.

First, Gordon ensures all filters are initialized: the global filter, relevant per-site filter filters, and the impression-site quota and conversion-site quota filters. Second, it computes the privacy losses incurred—both at the epoch level (§C[26]) and at the site level (Alg. 5). Third, it checks whether all filters have sufficient budget and attempts to consume it. Because consumption must be atomic at the report level, Gordon uses a two-phase commit protocol to deduct privacy losses from multiple filters. Alg. 4 formalizes this: if any filter cannot afford its share of the privacy loss, the report is zeroed out, and no budget is consumed from any filter for that report.

Finally, Gordon returns the (possibly empty) report as the response to the query at the current time step.

## B.2 Privacy proofs

**Mechanisms.** Alg. 2 defines two types of interactive mechanisms. First, for each beneficiary site $b$ we can denote by $\mathcal{M}^b$ the interactive mechanism that only interacts with $b$. Second, $\mathcal{M}$ is the interactive mechanism that interacts with all the beneficiary sites concurrently. Remark that the database $D$ is fixed upfront for simplicity, but a reasoning identical to [26, Alg. 2] generalizes to adaptively generated data. Another simplification compared to [26] is that public events are never relevant events for our attribution functions (*i.e.,* , the output of a conversion report can only depend on impressions, not on other conversions). This is enforcing the constraint on queries mentioned in Case 1 of [26, Thm. 1].

**Levels of accounting.** EpochImpSiteBudget computes privacy loss at a different granularity than EpochBudget, using Def. 5.

**Definition 5** (Device-epoch-site neighborhood relation)**.** *Consider a device $d \in \mathcal{D}$, an epoch $e \in \mathcal{E}$, an impression site $i \in \mathcal{S}$ and a set of impression events happening on $i$: $F_i \in \{i\} \times \mathcal{I}$.*

*We say $D \sim_{d,e,i,F_i} D'$ if there exists $D_0$ and $F \in \mathcal{S} \setminus \{i\} \times \mathcal{I}$ such that:*

$$\{D, D'\} = \{D_0 + (d, e, F), D_0 + (d, e, F \cup F_i)\} \quad (24)$$

**Theorem 9.** *Consider $x \in \mathcal{X}$ on device $d$, with global filter*

**Algorithm 2** Formalism for DP analysis

**Input**
  Database $D$
  Stream of adaptively chosen queries
**function** $\mathcal{M}(D)$
  $(S_b)_{b\in\mathcal{S}} = (\emptyset)_{b\in\mathcal{S}}$
  **for** $(d, e, F) \in D$ **do**
    **for** $f \in F : f = (c, b, \text{conv})$ **do**
      Generate report identifier $r \xleftarrow{\$} U(\mathbb{Z})$
      *// Save mapping from r to the device that generated it*
      $d_r \leftarrow d$
      $S_b \leftarrow S_b \cup \{(r, f)\}$
  *// Each beneficiary receives its public events and corresponding report identifiers*
  **for** $b \in \mathcal{S}$ **do**
    **output** $S_b$ to $b$
  *// Beneficiaries ask queries interactively*
  **for** $t \in [t_{\max}]$ **do**
    **receive** $Q_t^{b_t}$ from beneficiary site $b_t$.
    **output** AnswerQuery$(Q_t^{b_t})$ to $b_t$
*// Collect, aggregate and noise reports to answer Q*
**function** AnswerQuery(Q)
  Get report identifiers $R$ and noise parameter $\sigma$ from $Q$
  **for** $r \in R$ **do**
    Read $Q$ to get conversion site $c$, beneficiary site $b$, impression sites $\mathbf{i}$, target epochs $E$, attribution function $A$ for report $r$.
    $\rho_r \leftarrow \text{GenerateReport}(d, c, b, \mathbf{i}, E, A, \sigma)$
  Sample $X \sim \mathcal{L}(\sigma)$
  **return** $\sum_{r\in R} \rho_r + X$

**Algorithm 3** Gordon algorithm (on-device)

**Input**
  Filter and quota capacities $\epsilon_{\text{global}}$, $\epsilon_{\text{per-site}}$, $\epsilon_{\text{imp-quota}}$, $\epsilon_{\text{conv-quota}}$
*// Generate report and update on-device budget*
**function** GenerateReport$(d, c, b, \mathbf{i}, E, A, \sigma)$
  **for** $e \in E$ **do**
    $x \leftarrow (d, e, D_d^e)$
    **if** $\mathcal{F}_x$ is not defined **then**
      $\mathcal{F}_x \leftarrow \text{InitializeFilters}(\epsilon_{\text{global}}, \epsilon_{\text{per-site}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$
    $F_e \leftarrow D_d^e$
    *// Compute individual device-epoch-level privacy losses, same as "ComputeIndividualBudget" defined in appendix D of [26]*
    $\epsilon_x^t \leftarrow \text{EpochBudget}(x, d, E, A, \mathcal{L}, \sigma)$
    $\epsilon_{\mathbf{x}}^{\mathbf{i},t} \leftarrow \{\}$ *// Initialize empty map*
    **for** $i \in \mathbf{i}$ **do**
      *// Compute individual device-epoch-site-level privacy losses, Alg. 5*
      $\epsilon_x^i \leftarrow \text{EpochImpSiteBudget}(x, i, d, E, A, \mathcal{L}, \sigma)$
      $\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i] \leftarrow \epsilon_x^i$
    *// Atomic filter check and update, Alg. 4*
    **if** AtomicFilterCheckAndConsume$(\mathcal{F}_x, b, c, \mathbf{i}, \epsilon_x^t, \epsilon_{\mathbf{x}}^{\mathbf{i},t}) =$ FALSE **then**
      $F_e \leftarrow \emptyset$ *// Empty the epoch if any filter check fails*
  $\rho \leftarrow A((F_e)_{e\in E})$ *// Clipped attribution report*
  **return** $\rho$

*capacity $\epsilon_{global}$ and per-site filter capacity $\epsilon_{per\text{-}site}$. The two following properties hold simultaneously:*

1. *$\mathcal{M}$ satisfies individual device-epoch $\epsilon_{global}$-DP for $x$ under public information $C$.*
2. *For each beneficiary site $b \in \mathcal{S}$, $\mathcal{M}^b$ satisfies individual device-epoch $\epsilon_{per\text{-}site}$-DP for $x$ under public information $C_b$.*

*Proof.* Take a device-epoch $x = (d, e, F) \in \mathcal{X}$ and a database $D$ that doesn't contain $(d, e)$. Denote by $x_C = (d, e, F \cap C)$ the device-epoch obtained by keeping only public events $C$ from $x$, where public events are the set of all conversions. Take $v \in \text{Range}(\mathcal{M})$.

**1. Our first goal is equivalent to showing that:**

$$\left|\ln\left(\frac{\Pr[\mathcal{M}(D + x_C) = v]}{\Pr[\mathcal{M}(D + x) = v]}\right)\right| \leq \epsilon_{\text{global}}. \tag{25}$$

Consider any database $D'$. $v$ is the vector of values returned at different points in Alg. 2. We split it into $v = (v_{\text{pub}}, v_1, \ldots, v_{t_{\max}})$ where $v_{\text{pub}}$ is a value for the output from Line 14 of Alg. 2, and $v_t$ is the output for query $t$ at Line 18. We denote by $\mathcal{M}_{\text{pub}}(D')$ the random variable of the output at Line 14, and $\mathcal{M}_t(D')$ the random variable of the output at Line 18. By conditioning over past outputs $(v_{\text{pub}}, v_1, \ldots, v_{t-1})$ at each time step $t \in [t_{\max}]$ we get:

$$\Pr[\mathcal{M}(D') = v] \tag{26}$$

$$= \Pr[\mathcal{M}_{\text{pub}}(D') = v_{\text{pub}}] \cdot \prod_{t=1}^{t_{\max}} \Pr[\mathcal{M}_t(D') = v_t | v_{<t}]. \tag{27}$$

Take $t \in [t_{\max}]$. By Algorithm 2 we have:

$$\Pr[\mathcal{M}_t(D') = v_t | v_{<t}] = \Pr[\text{AnswerQuery}(Q_t; D', \mathcal{F}_t) = v_t], \tag{28}$$

where the query $Q_t$ and the state of the privacy filters (across all device-epochs) $\mathcal{F}_t$ are functions of past results $v_{<t}$, and AnswerQuery is defined in Alg. 2. In particular, the behaviors of $Q_t$ and $\mathcal{F}$ are fixed given the views from previous steps, as $v_{<t}$.

Finally, if we denote by $\rho_r(D'; \mathcal{F})$ the filtered report returned by Alg. 3 we get:

**Algorithm 4** 2-Phase Commit Subroutine

**Input:**

$\epsilon_x^t$: epoch-level privacy loss for a particular report
$\epsilon_{\mathbf{x}}^{\mathbf{i},t}$: epoch-site-level privacy loss for a particular report
canConsume: function as is defined in Def. 1
tryConsume: function as is defined in Def. 1

**Output:**

Boolean function if all filters have enough budget for the privacy loss $\epsilon_x^t$ or not.

**function** AtomicFilterCheckAndConsume($\mathcal{F}_x, b, c, \mathbf{i}, \epsilon_x^t, \epsilon_{\mathbf{x}}^{\mathbf{i},t}$)
  *// Phase 1: Prepare - check if all filters can consume*
  **if** $\mathcal{F}_x^{\text{per-site filter}[b]}$.canConsume($\epsilon_x^t$) = FALSE **then**
    **return** FALSE
  **if** $\mathcal{F}_x^{\text{global filter}}$.canConsume($\epsilon_x^t$) = FALSE **then**
    **return** FALSE
  **if** $\mathcal{F}_x^{\epsilon_{\text{conv-quota}}[c]}$.canConsume($\epsilon_x^t$) = FALSE **then**
    **return** FALSE
  **for** $i \in \mathbf{i}$ **do**
    **if** $\mathcal{F}_x^{\epsilon_{\text{imp-quota}}[i]}$.canConsume($\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i]$) = FALSE **then**
      **return** FALSE
  *// Phase 2: Commit - consume from all filters*
  *// Privacy filters under no collusion*
  $\mathcal{F}_x^{\text{per-site filter}[b]}$.tryConsume($\epsilon_x^t$)
  *// Privacy filter under collusion*
  $\mathcal{F}_x^{\text{global filter}}$.tryConsume($\epsilon_x^t$)
  *// Quota filter for conversion site*
  $\mathcal{F}_x^{\epsilon_{\text{conv-quota}}[c]}$.tryConsume($\epsilon_x^t$)
  *// Privacy filters for impression sites*
  **for** $i \in \mathbf{i}$ **do**
    *// Invididual device-epoch-impression site loss.*
    $\mathcal{F}_x^{\epsilon_{\text{imp-quota}}[i]}$.tryConsume $\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i]$)
  **return** TRUE

$$\Pr[\mathcal{M}_t(D') = v_t | v_{<t}] = \Pr\left[\sum_{r \in R_t} \rho_r(D'; \mathcal{F}_{t,r}) + X_t = v_t\right], \tag{29}$$

where $X_t$ is the Laplace noise added at time $t$. This equality is a direct quantification of "QueryAnswer" in Alg. 2, as for each $r \in R_t$, "QueryAnswer" generates filtered reports and sums over these reports with the Laplace noise $X_t$ added to them.

Now, we instantiate $D'$ to be a neighboring dataset of $D$, specifically $D' = D + x_c$, then we have:

$$\Pr[\mathcal{M}(D + x_C) = v] = \prod_{t=1}^{t_{\max}} \Pr\left[\sum_{r \in R_t} \rho_r(D + x_C) + X_t = v_t\right] \tag{30}$$

$$= \prod_{t=1}^{t_{\max}} \Pr\left[\sum_{r \in R_t} \rho_r(D) + X_t = v_t\right], \tag{31}$$

**Algorithm 5** Compute epoch-site level privacy budget

**Input:**

$x$ : device-epoch record $(d, e, F)$
$i$ : impression site
$d$ : device
$E$ : set of epochs, from which we can extract set of impression sites in these epochs relevant to $A$
$A$ : attribution function
$\mathcal{L}$ : parameterized noise distribution
$\sigma$ : noise scale

**Output:**

Returns the epoch-site-level individual privacy loss for impression site $i$ at device-epoch $x$

**function** EpochImpSiteBudget($x, i, d, E, A, \mathcal{L}, \sigma$)
  Extract epoch $e$ from $x = (d, e, F)$
  $\mathbf{i}_x \leftarrow \{i \in S \mid x = (d, e, F) \ \wedge \ (i, \text{imp}) \in F\}$
  *// Relevant epoch-site level events for current epoch*
  $E_{\text{relevant}} \leftarrow \{f \in F | f = (i, \text{imp}) \ \wedge \ \|A(f)\|_1 > 0\}$
  *// Compute epoch-site-level privacy losses*
  **if** $E_{\text{relevant}} = \emptyset$ **then**
    *// Case 2 of equations (13) or (14): No relevant epoch-site events in epoch*
    $\Delta \leftarrow 0$
  *// Check if we have single or multiple epoch-sites in E*
  **if** $|E| = 1$ AND $|\{\text{site } i \text{ relevant to } A \text{ in epoch } e \in E\}| = 1$ **then**
    *// Case 1 of equation (13): Single epoch, and only one impression site with relevant events to A in this epoch*
    $\Delta \leftarrow \Delta_{x_i}(\rho)$, where $\Delta_{x_i}(\rho)$ is the upper bound on the value with the same name specified in theorem 7
  **else**
    *// Case 1 of equation (14): Request either touches multiple epochs or multiples epoch-sites with relevant events*
    $\Delta \leftarrow \Delta(\rho)$, where $\Delta(\rho)$ is the upper bound on the value with the same name specified in theorem 3
  **return** $\Delta$ / $\sigma$

where the last equality is by Def. 2 where $\rho_r(D + x_C) = \rho_r(D + x_C \cap F_A) = \rho_r(D \cap F_A)$ since $F_A \subset \mathcal{I}$ and $x_C \cap \mathcal{I} = \emptyset$. Since the filter is unambiguous, we also abused the notation by letting $\rho_r(\cdot) := \rho_r(\cdot, \mathcal{F}_{t,r})$,.

Similarly, we get:

$$\Pr[\mathcal{M}(D+x)=v] \tag{32}$$
$$=\prod_{t=1}^{t_{\max}}\Pr\left[\sum_{r\in R_t}\rho_r(D+x)+X_t=v_t\right] \tag{33}$$
$$=\prod_{t=1}^{t_{\max}}\Pr\left[\sum_{r\in R_t^x}\rho_r(D+x)+\sum_{r\in R_t\setminus R_t^x}\rho_r(D)+X_t=v_t\right], \tag{34}$$

where $R_t^x := \{r \in R_t : d_r = d, e \in E_r\}$ is the set of reports where $x$ is queried.

Take a report $r \in R_t^x$. Recall that $\text{pass}_r$ denotes whether $r$ passed the atomic filter check in Alg. 4.

- If $\text{pass}_r = 0$, we have $\rho_r(D+x) = \rho_r(D)$ because of Alg. 3, Line 19. Note that $\text{pass}_r = 0$ can happen even if $\mathcal{F}^{\text{global filter}}$ has enough budget, for instance if the per-site filter or a quota is out of budget.
- If $\text{pass}_r = 1$, we have $\|\rho_r(D+x) - \rho_r(D)\|_1 \le \Delta_x\rho_r \le \sigma\cdot\epsilon_r\text{pass}_r$ where $\epsilon_r\text{pass}_r$ is the individual device-epoch loss for $x$ successfully deducted from $\mathcal{F}^{\text{global filter}}$ for report $r$. Since $\text{pass}_r$ implies that $r$ passes $\mathcal{F}^{\text{global filter}}$, by definition of $\mathcal{F}^{\text{global filter}}$, the accumulated loss over all reports is below the filter capacity, i.e.,

$$\sum_{t=1}^{t_{\max}}\sum_{r\in R_t}\epsilon_r\text{pass}_r=\sum_{t=1}^{t_{\max}}\sum_{r\in R_t^x}\epsilon_r\text{pass}_r\le\epsilon_{\text{global}}, \tag{35}$$

  where the first equality is because $\Delta_{x'}\rho_{x'} = 0$ for $x \neq x' \in R_t^x$ not queried.

Finally, by property of the Laplace distribution, for $t \in [t_{\max}]$ we have:

$$\left|\ln\left(\frac{\Pr[\sum_{r\in R_t}\rho_r(D)+X_t=v_t]}{\Pr[\sum_{r\in R_t}\rho_r(D+x)+X_t=v_t]}\right)\right|\le\Delta_x Q_t/\sigma \tag{36}$$
$$\le\sum_{r\in R_t}\Delta_x\rho_r/\sigma \tag{37}$$
$$\le\sum_{r\in R_t}\epsilon_r\text{pass}_r, \tag{38}$$

where $\sigma$ is the noise scale for the Laplace distribution.

Let's put everything together. First, by definition of $x_C$, we observe that all the outputs $v_{\text{pub}}$ at Line 14 are identical across both worlds. For the query outputs, we apply the triangle inequality across all time steps (Ineq. (42)):

$$\left|\ln\left(\frac{\Pr[\mathcal{M}(D+x_C)=v]}{\Pr[\mathcal{M}(D+x)=v]}\right)\right| \tag{39}$$
$$=\left|\ln\left(\frac{\prod_{t=1}^{t_{\max}}\Pr[\sum_{r\in R_t}\rho_r(D)+X_t=v_t]}{\prod_{t=1}^{t_{\max}}\Pr[\sum_{r\in R_t}\rho_r(D+x)+X_t=v_t]}\right)\right| \tag{40}$$
$$=\left|\sum_{t=1}^{t_{\max}}\ln\left(\frac{\Pr[\sum_{r\in R_t}\rho_r(D)+X_t=v_t]}{\Pr[\sum_{r\in R_t}\rho_r(D+x)+X_t=v_t]}\right)\right| \tag{41}$$
$$\le\sum_{t=1}^{t_{\max}}\left|\ln\left(\frac{\Pr[\sum_{r\in R_t}\rho_r(D)+X_t=v_t]}{\Pr[\sum_{r\in R_t}\rho_r(D+x)+X_t=v_t]}\right)\right| \tag{42}$$
$$\le\sum_{t=1}^{t_{\max}}\sum_{r\in R_t}\epsilon_r\text{pass}_r=\sum_{t=1}^{t_{\max}}\sum_{r\in R_t^x}\epsilon_r\text{pass}_r \tag{43}$$
$$\le\epsilon_{\text{global}}, \tag{44}$$

where Ineq. (43) follows from Ineq. (38) and Ineq. (44) follows from Ineq. (35). This establishes the first property: $\mathcal{M}$ satisfies individual device-epoch $\epsilon_{\text{global}}$-DP for $x$ under public information $C$.

**2. We now consider the view of a single beneficiary $b \in \mathcal{S}$.** Our goal is equivalent to showing that:

$$\left|\ln\left(\frac{\Pr[\mathcal{M}_b(D+x_C)=v]}{\Pr[\mathcal{M}_b(D+x)=v]}\right)\right|\le\epsilon_{\text{per-site}}. \tag{45}$$

Let $\mathcal{M}$ be the global mechanism that processes queries from all beneficiaries. Partition its queries into those from the single beneficiary $b$ and those from other parties (the "adv"):

$$Q=Q_b\cup Q_{\text{adv}}\quad\text{where}\quad Q_b=\{t\in[t_{\max}]:b_t=b\}. \tag{46}$$

Define

$$\mathcal{M}_b(Q_b)=(\mathcal{M}(Q_b\cup Q_{\text{adv}}))\big|_{\text{indices in }Q_b}, \tag{47}$$

such that $\mathcal{M}_b$ behaves exactly like $\mathcal{M}$ on the queries issued by $b$, but treats the adversary's queries as fixed. Concretely, from $b$'s viewpoint, the queries in $Q_{\text{adv}}$ and their outputs are independent arbitrary events that $b$ has no control over (i.e. the adversary sees and makes the same queries in both worlds).

Hence, for any database $D \in \mathbb{D}$ and row $x \in \mathcal{X}$, we have the equality of distributions w.r.t. the outputs that $b$ observes:

$$\Pr\left[\mathcal{M}_b(D+x)=v\right]=\Pr\left[(\mathcal{M}(D+x))\big|_{Q_b}=v\right]. \tag{48}$$

This justifies focusing on $\mathcal{M}_b$ rather than the full mechanism $\mathcal{M}$ when bounding the privacy loss for $b$.

Recall that, as before, the public information is identical across both worlds, so we can focus on queries only —more specifically, queries that are observed by $b$:

$$\Pr[\mathcal{M}_b(D+x)=v] \tag{49}$$
$$=\prod_{t\in[t_{\max}]:b_t=b}\Pr\left[\sum_{r\in R_t}\rho_r(D+x;\mathcal{F}_{t,r})+X_t=v_t\right]. \tag{50}$$

Secondly, we can bound the privacy loss $\epsilon_t$ at any given

time $t \in [t_{\max}]$ by the property of Laplace distribution:

$$\left|\ln\left(\frac{\Pr\left[\sum_{r\in R_t}\rho_r(D)+X_t=v_t\right]}{\Pr\left[\sum_{r\in R_t}\rho_r(D+x)+X_t=v_t\right]}\right)\right| \le \sum_{r\in R_t}\Delta_x\rho^t_{b_t}/\sigma \tag{51}$$

$$\le \sum_{r\in R_t}\epsilon_r \text{pass}_r \tag{52}$$

Finally, we use the fact that $\text{pass}_r$ implies that $r$ passes $\mathcal{F}^{\text{per-site filter}}$ succesfully, and thus:

$$\sum_{t\in[t_{\max}]:b_t=b}\sum_{r\in R_t}\epsilon_r\text{pass}_r \le \epsilon_{\text{per-site}}. \tag{53}$$

Analogous to the first part, we conclude with a triangle inequality:

$$\left|\ln\left(\frac{\Pr\left[\mathcal{M}_b(D+x_C)=v\right]}{\Pr\left[\mathcal{M}_b(D+x)=v\right]}\right)\right| \tag{54}$$

$$\le \sum_{t\in[t_{\max}]:b_t=b}\left|\ln\left(\frac{\Pr\left[\sum_{r\in R_t}\rho_r(D)+X_t=v_t\right]}{\Pr\left[\sum_{r\in R_t}\rho_r(D+x)+X_t=v_t\right]}\right)\right| \tag{55}$$

$$\le \sum_{t\in[t_{\max}]:b_t=b}\epsilon_t \le \sum_{t\in[t_{\max}]:b_t=b}\sum_{r\in R_t}\epsilon_r\text{pass}_r \tag{56}$$

$$\le \epsilon_{\text{per-site}}. \tag{57}$$

□

### B.3 DoS resilience proofs

This section proves our main resilience result for Gordon's quota-based online algorithm: Thm. 1. First, Lemma 1 shows that the 2-PC check (Alg. 4) ensures (1) atomic consumption across all filters relevant to a query, and (2) when all filters have sufficient budget, each consumes an amount proportional to its level-specific sensitivity—either at epoch or at epoch-site level. Next, Lemma 2 bounds the total privacy budget the adversary can consume from global filter at any qualified time, using the atomic consumption guarantees of Lemma 1. This final bound directly implies Thm. 1.

**Definition 6.** *We define the following notations for privacy loss accounting:*

- $\epsilon^{\le t}_{global}$*: the cumulative privacy loss charged to the global filter up to step* $t$ *before 2PC is triggered at step* $t$*.*
- $D^{\le e}$*: The subset of database* $D$ *containing records with epochs up to and including* $e$*.*

We first formalize the atomicity property of the 2-PC algorithm for consuming privacy budgets from relevant filters when Gordon answers a query at any time step $k$ (Alg. 4)

**Lemma 1** (2-phase commit filter guarantees)**.** *For query* $k$*, let:*

$$pass(k) \triangleq \begin{cases} 1 & \text{if AtomicFilterCheckAndConsume returns } \textsf{TRUE} \text{ for query } k \\ 0 & \text{otherwise} \end{cases} \tag{58}$$

*The* AtomicFilterCheckAndConsume *function in Algorithm 4 guarantees the following properties:*

*For any query* $k$ *processed by* AtomicFilterCheckAndConsume*, if* $pass(k) = 1$*, then*

1. ***Epoch-level Consistency Property:*** *the per-site filter, global filter, and conversion-site quota-filter all consume exactly the same amount of budget* $\epsilon^t_x$ *for that query.*
2. ***Epoch-site-level Consistency Property:*** *the impression-site quota filter consumes exactly* $\epsilon^i_x[i]$*, which represents the device-epoch-impressionsite-level individual privacy loss.*

*Proof.* We can prove both properties at the same time. Fix an arbitrary query $k$, for which $\text{pass}(k) = 1$. From Algorithm 4, we observe that AtomicFilterCheckAndConsume returns TRUE if and only if: (1) all canConsume checks in Phase 1 pass, and (2) All tryConsume operations in Phase 2 are executed. For a query from conversion site $c$ with beneficiary site $b$ and impression sites $\mathbf{i}$, the function calls:

- $\mathcal{F}_x^{\text{per-site filter}[b]}.\text{tryConsume}(\epsilon^t_x)$
- $\mathcal{F}_x^{\text{global filter}}.\text{tryConsume}(\epsilon^t_x)$
- $\mathcal{F}_x^{\text{conversion-site quota}[c]}.\text{tryConsume}(\epsilon^t_x)$
- For each $i \in \mathbf{i}$: $\mathcal{F}_x^{\epsilon_{\text{imp-quota}}[i]}.\text{tryConsume}(\epsilon^{\mathbf{i},t}_{\mathbf{x}}[i])$

Note that $\epsilon^t_x$ is computed once at EpochBudget in Line 11 of Algorithm 3 and represents the device-epoch-level individual privacy loss. Similarly, each $\epsilon^{\mathbf{i},t}_{\mathbf{x}}[i]$ is computed once via EpochImpSiteBudget and represents the device-epoch-impressionSite-level individual privacy loss. Therefore, when $\text{pass}(k) = 1$, the conversion-site quota, per-site filter, and global filter all consume exactly the same amount $\epsilon^t_x$, while each impression-site quota filter consumes its specific amount $\epsilon^{\mathbf{i},t}_{\mathbf{x}}[i]$, which is proportional to its sensitivity at the impression site level. □

With such atomic guarantees for every step $k$ up to some time $t$, we can show the following upper bounds for how much an adversary can deplete the global filter budget by the end of time $t$. For notations, at step $t$ in Line 16, suppose that beneficiary site $b$ requests a report $\rho_{r,E,A}$ with noise $\sigma$ through conversion site $c$ for impression sites $\mathbf{i}$. Consider a device-epoch $x$, with individual budget $\epsilon^t_x$ computed at Line 11 in Alg. 3. Denote by $N^{\le t,\text{adv}}$ the number of conversion sites in $\mathbf{bad_c}$ with respect to $x$ that were queried with non-zero budget by step $t$. Denote by $M^{\le t,\text{adv}}$ the number of impression sites in $\mathbf{bad_i}$ with respect to $x$ that were queried with non-zero budget by step $t$.

**Lemma 2.** *At the end of time* $t$*, given that the adversary has created at most* $M^{adv}$ *and* $N^{adv}$ `imp-quota` *and* `conv-quota` *filters, respectively, we have the following upper bounds on the global filter filter budget that the adversary can consume:*

- *The adversary consumes at most* $M^{\text{adv}}\epsilon_{imp\text{-}quota}$ *budget from the global filter.*

- *The adversary consumes at most $N^{\text{adv}}\epsilon_{\textit{conv-quota}}$ budget from the global filter.*

*Proof (first upper bound by $N^{\text{adv}}$).* The total privacy loss in the global filter incurred by the attackers for the global filter by step $t$, not inclusive, is:

$$\epsilon_{\text{used}}^{\mathbf{bad}} = \epsilon_{\text{global}}^{\leq t-1,\mathbf{bad}} = \sum_{k=[t-1]:c_k\in\mathbf{bad_c}} \epsilon_{\text{global}}^{k} \cdot \text{pass}(k). \quad (59)$$

By basic composition under a pure DP filter and the definition of individual privacy loss, the total privacy loss equals:

$$= \sum_{c\in\mathbf{bad_c}} \sum_{k<t:c_k=c} \Delta_x \rho_{b_k}^k \cdot \text{pass}(k), \quad (60)$$

where $\Delta_x\rho_{b_k}^k$ represents how much the attribution report generated for query $k$ from beneficiary $b_k$ can change with respect to $x$.

By the consistency property of lemma 1, Alg. 4 ensures that privacy loss is only incurred on $k$ where $\text{pass}(k) = 1$. So, for each conversion site $c$, the filter, $\epsilon_{\text{conv-quota}}[c]$, precisely tracks the privacy losses incurred, so

$$\epsilon_{\text{conv-quota}}[c]^{\leq t-1} = \sum_{k<t:c_k=c} \Delta_x \rho_{b_k}^k \cdot \text{pass}(k). \quad (61)$$

Substitute this equality into equation (60), we get:

$$\epsilon_{\text{global}}^{\leq t-1,\mathbf{bad}} = \sum_{c\in\mathbf{bad_c}} \epsilon_{\text{conv-quota}}[c]^{\leq t-1}. \quad (62)$$

This sum can be restricted to conversion sites with non-zero privacy loss, i.e.:

$$= \sum_{c\in\mathbf{bad_c}:\epsilon_{\text{conv-quota}}[c]^{\leq t-1}>0} \epsilon_{\text{conv-quota}}[c]^{\leq t-1} \quad (63)$$

$$\leq \sum_{c\in\mathbf{bad_c}:\epsilon_{\text{conv-quota}}[c]^{\leq t-1}>0} \epsilon_{\text{conv-quota}}, \quad (64)$$

where $\epsilon_{\text{conv-quota}}$ is the capacity of each $\epsilon_{\text{conv-quota}}$ filter. It follows that the number of conversion sites with non-zero privacy loss is precisely $N^{\leq t,\text{adv}}$, so:

$$\leq \left|\left\{c \in \mathbf{bad_c} : \epsilon_{\text{conv-quota}}[c]^{\leq t-1} > 0\right\}\right| \cdot \epsilon_{\text{conv-quota}} \quad (65)$$

$$= N^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{conv-quota}}. \quad (66)$$

Now, during the 2-PC for time $t$, we have the following cases:

- Suppose $\epsilon_x^t$ is a reasonable value, in the sense that it's bounded by the capacity $\epsilon_{\text{conv-quota}}$. Then,

$$\epsilon_{\text{global}}^{\leq t,\mathbf{bad}} = \epsilon_{\text{used}}^{\mathbf{bad}} + \epsilon_x^t \leq N^{\leq t,\text{adv}} \cdot \epsilon_{\text{conv-quota}}. \quad (67)$$

- Otherwise, $\epsilon_x^t$ is unreasonable, in which case $\epsilon_x^t$ exceeds the capacity $\epsilon_{\text{conv-quota}}$. In this case,

$$\epsilon_{\text{conv-quota}}[c_t]^{\leq t-1} + \epsilon_x^t \geq \epsilon_{\text{conv-quota}}, \quad (68)$$

causing $\mathcal{F}_x^{\epsilon_{\text{conv-quota}}[c]}.\text{canConsume}(\epsilon_x^t)$ to return FALSE by definition, so no budget is spent at all. In such a case,

$$\epsilon_{\text{global}}^{\leq t,\mathbf{bad}} = \epsilon_{\text{used}}^{\mathbf{bad}} + 0 = \epsilon_{\text{used}}^{\mathbf{bad}} \leq N^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{conv-quota}}, \quad (69)$$

by equation (66).

Since the adversary has created at most $N^{\text{adv}}$ by the end of time $t$, it must be the case that $N^{\leq t-1,\text{adv}} \leq N^{\leq t,\text{adv}} \leq N^{\text{adv}}$. This means that, in either case, the attackers can consume at most $N^{\leq t,\text{adv}}\epsilon_{\text{conv-quota}} \leq N^{\text{adv}}\epsilon_{\text{conv-quota}}$ of the global filter budget by the end of time $t$, as desired □

*Proof (second upper bound by $M^{\text{adv}}$).* By basic composition under a pure DP filter, we know $\epsilon_{\text{global}}$ is by definition the sum of global filter consumption at each time up to the end of time $t-1$:

$$\epsilon_{\text{global}}^{\leq t-1,\mathbf{bad}} = \sum_{k\in[t-1]:c_k\in\mathbf{bad_c}} \epsilon_x^k \cdot \text{pass}(k) \quad (70)$$

$$= \sum_{k\in[t-1]:c_k\in\mathbf{bad_c}} \Delta_x \rho_{b_k}^k \cdot \text{pass}(k), \quad (71)$$

where $\Delta_x\rho_{b_k}^k$ is how much the attribution report generated for query $k$ from beneficiary $b_k$ can change with respect to $x$, which we substitute next. Let $\vec{x} = (x_{i_1}, \ldots, x_{i_m})$ be the vector of $|\mathbf{i}_k| = m$ device-epoch-sites, where $x_i = (d, e, F_i)$ has all its events on site $i$. Then, we let the corresponding neighboring dataset of $D \in \mathbb{D}$ be $D + \vec{x}$, so that:

$$\rho(D+\vec{x}) - \rho(D) = \sum_{j=1}^{m} \rho(D + x_{i_j}) - \rho(D + x_{i_{j-1}}), \quad (72)$$

where $x_{i_0} = 0$. Thus, we substitute by the following decomposition of a query's sensitivity across the impression sites relevant to that query, by how it is assigned in Alg. 3 Line 15:

$$\Delta_x\rho_{b_k}^k = \max_{D,D'\in\mathcal{D}'=D+x} \left\|\rho_{b_k}^k(D') - \rho_{b_k}^k(D)\right\|_1 \quad (73)$$

$$= \max_{D\in\mathcal{D}} \left\|\rho_{b_k}^k(D+x) - \rho_{b_k}^k(D)\right\|_1 \quad (74)$$

$$\leq \max_{D\in\mathcal{D}} \sum_{j\in[m]:i_j\in\mathbf{bad_i}} \|\rho_{b_k}^k(D + x_1 + \ldots + x_i) - \quad (75)$$

$$\rho_{b_k}^k(D + x_1 + \ldots + x_{i-1})\|_1 \quad (76)$$

$$\leq \sum_{j\in[m]:i_j\in\mathbf{bad_i}} \max_{\hat{D}\in\mathcal{D}} \left\|\rho_{b_k}^k(\hat{D} + x_{i_j}) - \rho_{b_k}^k(\hat{D})\right\|_1 \quad (77)$$

$$= \sum_{j\in[m]:i_j\in\mathbf{bad_i}} \Delta_x^{i_j}\rho_{b_k}^k = \sum_{i\in\mathbf{bad_i}} \Delta_x^i \rho_{b_k}^k \quad (78)$$

$$= \sum_{i\in\mathbf{bad_i}} \epsilon_{\mathbf{x}}^{\mathbf{i},k}[i], \quad (79)$$

where the inequality in equation 75 is because of the following. First, by the restriction in the sum, we know $k$ satisfies $c_k \in \mathbf{bad_c}$. Second, recall that, for a conversion site to incur privacy losses on impression sites, the conversion site must register these impression sites, meaning that if $c_k \in \mathbf{bad_c}$, then $\mathbf{i}_k \subseteq \mathbf{bad_i}$. Now, plug the the inequality

$\Delta_x \rho^k_{b_k} \leq \sum_{i \in \mathbf{bad_i}} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i]$ in, we get

$$\epsilon^{\leq t-1,\mathbf{bad}}_{\text{global}} \leq \sum_{k\in[t-1]:c_k\in\mathbf{bad_c}} \sum_{i\in\mathbf{bad_i}} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k) \quad (80)$$

$$= \sum_{i\in\mathbf{bad_i}} \sum_{k\in[t-1]:c_k\in\mathbf{bad_c}, i\in\mathbf{i}_k} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k), \quad (81)$$

$$\leq \sum_{i\in\mathbf{bad_i}} \sum_{k\in[t-1]:i\in\mathbf{i}_k} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k), \quad (82)$$

by changing order of summation, and the last inequality by relaxing the "$c_k \in \mathbf{bad_c}$" condition. But note that

$$\epsilon^{\leq t-1}_{\text{imp-quota}}[i] = \sum_{k\in[t-1]:i\in\mathbf{i}_k} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k), \quad (83)$$

because, by epoch-site-level consistency property in lemma 1, we know that only relevant site $i$ at time $k$, where every filter has enough budget to pass the 2-PC check, will have epoch-site level privacy losses incurred. Substituting this equality into equation (82), we get:

$$\epsilon^{\leq t-1,\mathbf{bad}}_{\text{global}} \leq \sum_{i\in\mathbf{bad_i}} \epsilon^{\leq t-1}_{\text{imp-quota}}[i] \quad (84)$$

$$= \sum_{i\in\mathbf{bad_i}:\epsilon^{\leq t-1}_{\text{imp-quota}}[i]>0} \epsilon^{\leq t-1}_{\text{imp-quota}}[i] \quad (85)$$

$$\leq \sum_{i\in\mathbf{bad_i}:\epsilon^{\leq t-1}_{\text{imp-quota}}[i]>0} \epsilon_{\text{imp-quota}} \quad (86)$$

$$= \left|\left\{i \in \mathbf{bad_i} : \epsilon^{\leq t-1}_{\text{imp-quota}}[i] > 0\right\}\right| \cdot \epsilon_{\text{imp-quota}}, \quad (87)$$

because only non-zero privacy losses that were incurred contribute meaningfully to the composition. Finally, we note that $\left|\left\{i \in \mathbf{bad_i} : \epsilon^{\leq t-1}_{\text{imp-quota}}[i] > 0\right\}\right| \leq M^{\leq t-1,\text{adv}}$ by definition and:

$$\epsilon^{\leq t-1,\mathbf{bad}}_{\text{global}} \leq M^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{imp-quota}}. \quad (88)$$

Following this result, similar to the proof for part 1:

- Suppose $\epsilon^t_x \leq \epsilon_{\text{imp-quota}}$,

$$\epsilon^{\leq t,\mathbf{bad}}_{\text{global}} \leq M^{\leq t,\text{adv}} \cdot \epsilon_{\text{imp-quota}}. \quad (89)$$

- Else, $\epsilon^t_x > \epsilon_{\text{imp-quota}}$, then $\epsilon_{\text{imp-quota}}$ will be exceeded, causing canConsume to return FALSE, so,

$$\epsilon^{\leq t,\mathbf{bad}}_{\text{global}} = \epsilon^{\leq t-1,\mathbf{bad}}_{\text{global}} + 0 = \epsilon^{\leq t-1,\mathbf{bad}}_{\text{global}} \leq M^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{imp-quota}}, \quad (90)$$

by equation (88).

Since by the end of time $t$, the adversary has created at most $M^{\text{adv}}$ `imp-quota` filters, we know $M^{\leq t-1,\text{adv}} \leq M^{\leq t,\text{adv}} \leq M^{\text{adv}}$, which means that in both cases we have:

$$\epsilon^{\leq t,\mathbf{bad}}_{\text{global}} \leq M^{\leq t,\text{adv}} \epsilon_{\text{imp-quota}} \leq M^{\text{adv}} \epsilon_{\text{imp-quota}}, \quad (91)$$

as desired. □

Finally, we use the preceding lemma to show the overall DoS depletion resilience guarantee over Gordon's lifetime. Consider an execution of Alg. 2. A report at time step $k$ concerns with *one* conversion site $c_k$, and some subset of impression sites $\mathbf{i}_k \subseteq S$. The union of attackers, in particular, can control an arbitrary subset of conversion sites $\mathbf{bad_c} \subseteq S$, which may or may not contain $c_k$ at a given $k$. We denote by $N^{\text{adv}}$ the size of $|\mathbf{bad_c}|$ over the entire lifetime. Similarly, the adversary can control an arbitrary subset of impression sites $\mathbf{bad_i} \subseteq S$, which may or may not intersect with $\mathbf{i}_k$. We denote by $N^{\text{adv}}$ the size of $|\mathbf{bad_i}|$ over the entire lifetime. We let $\mathbf{bad} = \mathbf{bad_c} \cup \mathbf{bad_i}$ and $\mathbf{good} = S \backslash \mathbf{bad}$.

**Theorem 1 (Resilience to DoS depletion).** *Consider an adversary who manages to create $M^{adv}$ and $N^{adv}$ `imp-quota` and `conv-quota` filters, respectively. The maximum budget $\epsilon^{adv}_{global}$ that the adversary can consume from the global filter on a device $d$ is such that:*

$$\epsilon^{adv}_{global} \leq \min(M^{adv}\epsilon_{imp\text{-}quota},\ N^{adv}\epsilon_{conv\text{-}quota}).$$

*Proof.* The result follows directly from Lemma 2. At any time $t$, if the adversary controls at most $M^{\leq t,\text{adv}} \leq M^{\text{adv}}$ `imp-quota` and $N^{\leq t,\text{adv}} \leq N^{\text{adv}}$ `conv-quota` filters, then by Lemma 2:

$$\epsilon^{\text{adv}}_{\text{global}} \leq M^{\leq t,\text{adv}} \epsilon_{\text{imp-quota}} \leq M^{\text{adv}} \epsilon_{\text{imp-quota}} \quad (92)$$

$$\epsilon^{\text{adv}}_{\text{global}} \leq N^{\leq t,\text{adv}} \epsilon_{\text{conv-quota}} \leq N^{\text{adv}} \epsilon_{\text{conv-quota}}. \quad (93)$$

Therefore:

$$\epsilon^{\text{adv}}_{\text{global}} \leq \min(M^{\text{adv}}\epsilon_{\text{imp-quota}},\ N^{\text{adv}}\epsilon_{\text{conv-quota}}).$$

□

# C Batched Algorithm to Improve Utilization

## C.1 Algorithm

Alg. 6 describes the batched algorithm on a single device. Instead of executing GenerateReport as soon as a request comes, as in Alg. 3, requests are accumulated in a batch. Each epoch is divided into $k$ scheduling intervals, and since a request can request older epochs (up to a maximum attribution window length, *a.k.a.* data lifetime) we release budget progressively over $T$ intervals. For instance, if requests have attribution window of at most 2 epochs, we can divide this data lifetime into $T = 4$ releases, with $k = 2$ releases happening inside each epoch. We can also do $T = 2$ releases with $k = 1$ interval per epoch.

Budget release and unlocked budget semantics are defined as in [18]. $\mathcal{F}^{\text{global}}$.unlock becomes a no-op after $T$ releases, when the unlocked budget reaches the filter capacity $\epsilon_{\text{global}}$.

We define $\mathcal{A}, \mathcal{U} \leftarrow \text{TryAllocate}(\mathcal{R})$ as follows. TryAllocate takes a set of report requests $\mathcal{R}$. For each request, it executes a heuristic that estimates whether Alg. 3's GenerateReport will successfully allocate budget for the request (*i.e.,* the whether the filters will return TRUE at Line 18). It then calls GenerateReports on the requests that were predicted to be allocatable, and returns two sets: $\mathcal{A}$ the reports for requests that were executed, and $\mathcal{U}$ the remaining unallocated requests. TryAllocateOne behaves like TryAllocate, except

**Algorithm 6** Batched Algorithm (On-Device)

**Input:**

$\epsilon_{\text{global}}$, $\epsilon_{\text{per-site}}$, $\epsilon_{\text{imp-quota}}$, $\epsilon_{\text{conv-quota}}$: same parameters as Alg. 3.
$k$: number of scheduling intervals per epoch.
$T$: number of scheduling intervals to release the full budget.

**function** MAIN
  **for** $e \in \mathbb{N}$ **do**
    *// Initialize new epoch with its own filters*
    $\mathcal{F}_e \leftarrow \text{InitializeFilters}(\epsilon_{\text{global}}, \epsilon_{\text{per-site}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$
    *// Initially no global budget available*
    $\mathcal{F}_e^{\text{global}}.\text{unlocked} \leftarrow 0$
    $\mathcal{R}_{batch} \leftarrow \emptyset$ *// Requests for the batch phase*
    **for** $t \in [k]$ **do**
      $\mathcal{R}_{new} \leftarrow \text{ReceiveNewRequests}()$
      $\mathcal{A}, \mathcal{R}_{batch} \leftarrow \text{ScheduleBatch}(\mathcal{R}_{new}, \mathcal{R}_{batch})$
      $\text{SendReportsForRelease}(\mathcal{A})$

**function** SCHEDULEBATCH($\mathcal{R}_{new}, \mathcal{R}_{batch}$)
  *// 1. Initialization phase*
  **for** $e' \in [e]$ **do**
    $\mathcal{F}_e^{\text{global}}.\text{unlocked} \leftarrow \mathcal{F}_e^{\text{global}}.\text{unlocked} + \epsilon_{\text{global}}/T$
    **for** $i \in \mathcal{S}$ **do**
      *// impression-site quota on (only accepts requests within remaining budget).*
      $\mathcal{F}_e^{\text{imp-quota}}[i].\text{on} = \text{True}$
  $\mathcal{A}_{init}, \mathcal{U}_{init} \leftarrow \text{TryAllocate}(\mathcal{R}_{batch})$
  $\mathcal{A} \leftarrow \mathcal{A}_{init}$
  *// 2. Online phase*
  $a_{online}, u_{online} \leftarrow \text{TryAllocate}(\mathcal{R}_{new})$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_{online}$
  *// 3. Batch phase*
  **for** $e' \in [e], i \in \mathcal{S}$ **do**
    *// impression-site quota off (accepts all requests regardless of impression-site quota; requests still decrease filter budget).*
    $\mathcal{F}_e^{\text{imp-quota}}[i].\text{on} = \text{False}$
  $batch \leftarrow \mathcal{U}_{init} \cup \mathcal{U}_{online}$
  **do**
    $sorted \leftarrow \text{SortBatch}(batch)$
    $(a, u) \leftarrow \text{TryAllocateOne}(sorted)$
    $batch \leftarrow u$
    $\mathcal{A} \leftarrow \mathcal{A} \cup a$
  **while** $a \neq \emptyset$
  **return** $\mathcal{A}, batch$

that it stops after the first executed request. Our heuristic is a variant of GenerateReport that relies purely on public information — using proxy filters $\tilde{\mathcal{F}}$ with IDP optimizations turned off — allowing the scheduler to make decisions without leaking privacy across epochs.

SortBatch attaches a weight $(b_r, \epsilon_r)$ to each request $r$ in a batch, and then sorts by smallest weight first (in lexicographic order). The weights are defined as follows, using the proxy filters $\tilde{\mathcal{F}}$ define for TryAllocate. $\epsilon_r$ is the global epsilon requested by $r$ (either available as a request parameter, or computed as $\epsilon_r = \Delta\rho_r/\sigma$ for a Laplace noise scale $\sigma$). $b_r$ is the smallest budget consumed by any impression site $i \in \mathbf{i}_r$ requested by $r$, where the budget consumed by $i$ over the set of epochs $E$ considered in the queue is defined by the maximum budget consumed by $i$ over any epoch:

$$b_r := \min_{i \in \mathbf{i}_r} \max_{e \in E} \tilde{\mathcal{F}}^{\text{imp-quota}}[i].\text{consumed} \tag{94}$$

Finally, SendReportsForRelease prepares the reports from allocated requests to be sent at the right time, depending on the duration specified by each request.

## C.2 DoS resilience under batching

Under our mixed online and batch algorithm, the effort in terms of on device user interactions $U^{\text{adv}}$ required from an adversary to consume global filter budget depends on the overall workload of the system. As we saw in §4.3, in the worst case (*e.g.,* when there are no legitimate queries in the system) this can lead to a weaker upper-bound on the budget consumption by an adversary compared to Thm. 2, with the following result:

$$\epsilon_{\text{global}}^{\text{adv}} \leq (1+r)\epsilon_{\text{per-site}} \times \texttt{quota-count} \times (U^{\text{adv}} - 1). \tag{95}$$

Intuitively, this is because the attacker can batch conversion queries that all request the same impression: user interactions are only needed to create one impression under the adversary's control, as well as $U^{\text{adv}} - 1$ conversions that can be used to deplete global filter budget when the imp-quota filters are disabled (l.9 in Algorithm 1).

In practice however, we expect the benign workload to contain online queries (configured to return instantly, with no batching). To deny service to those queries requires a higher number of on device interactions for the adversary, so it is relevant to ask for a lower-bound on user interactions $U^{\text{adv}}$ required by an adversary to prevent a specific set of legitimate online queries from being allocated. Intuitively, even in the best case, even in the easiest case an attacker will need to cause $\epsilon_{\text{global}}^{\text{adv}} \geq \epsilon_{\text{global}}^{\text{good}}$ of global filter consumption to deny service to $\epsilon_{\text{global}}^{\text{good}}$ worth of legitimate online requests. Such denial comes at the higher $U^{\text{adv}}$ cost from Thm. 2. Formally, we have the following result:

**Theorem 10** (Graceful degradation for online queries under the batch algorithm)**.** *Consider a set of legitimate target queries, with total requested budget summing to $\epsilon_{global}^{good}$. To deny service to those target queries, an attacker requires the*

*following lower-bound on user interactions $U^{adv}$:*

$$\frac{1+n}{n}\frac{\epsilon^{good}_{global}}{(1+r)\epsilon_{per\text{-}site} \times \texttt{quota-count}} \leq U^{adv}.$$

*Proof.* The best case for the attacker is when online queries consume $\frac{\epsilon_{\text{global}}}{T}$ in this period and target queries arrive last, so that all budget consumed by the attacker is denied to target queries. This yields a lower-bound on the DoS attack budget consumption: $\epsilon^{\text{good}}_{\text{global}} \leq \epsilon^{\text{adv}}_{\text{global}}$.

This consumption has to apply to newly released $\frac{\epsilon_{\text{global}}}{T}$ global filter budget, which can happen lines 5 and 7 in Algorithm 1. In both cases all quota filters apply. By Thm. 2:

$$\begin{aligned}\epsilon^{\text{good}}_{\text{global}} &\leq \epsilon^{\text{adv}}_{\text{global}} \\ &\leq (1+r)\epsilon_{\text{per-site}} \times \frac{n}{1+n}\left(\texttt{quota-count} \times U^{\text{adv}}\right).\end{aligned}$$

Reorganizing the terms concludes the proof. □

In addition, during the batch allocation phase (lines 9 to 14 in Algorithm 1), the adversary would still need overcome the scheduler's sorting mechanism, to be scheduled before waiting legitimate requests. Since the sorting mechanism favors low-budget and underrepresented impression sites, the adversary would likely require more than one user interaction ($U^{\text{adv}} \gg 1$) to mount an attack, making Equation 95 pessimistic. The time dynamics and workload dependency of the batching phases make the analysis of such guarantees challenging though, and we leave a proper formal treatment of any guarantees related to sorting for future work.

# D Cross-report Privacy Loss Optimization

## D.1 Example from Fig. 2

We illustrate Gordon's *cross-report* privacy loss optimization using our running example. This optimization is orthogonal to Cookie Monster's *per-report* individual-DP-based strategies (§2.3), and instead leverage structure *across* reports, often requested by different intermediaries for the same conversion.

In Fig. 2, *r1.ex*, *r2.ex*, and *r3.ex* request single-advertiser reports from `attributionObject` (a), all on behalf of client *shoes.ex*; separately, *r1.ex* and *r2.ex* request cross-advertiser reports from (b) for their own purposes. All five reports operate on the same attribution histogram, assigning $30 to each of two impressions (epochs `e1`, `e2`). Cookie Monster computes a base epoch-level privacy loss of 0.3 per report (§2.3). Naïvely, one would expect a cumulative deduction of 0.9 from *shoes.ex*'s filters (three reports) and 1.5 from the global filter (five reports). Yet the Privacy Filters table shows only deductions of 0.6 and 0.9, respectively.

The discrepancy arises because some reports *shard* the histogram into non-overlapping pieces—enabling sensitivity-based optimizations. *r1.ex* and *r2.ex*'s single-advertiser reports from (a) each include a disjoint portion: `{1:30}` and `{2:30}`, respectively. Since both are funded by the same per-site filter (of *shoes.ex*), their combined release leaks no more than a single full histogram toward *shoes.ex*, incurring only 0.3 privacy loss. They likewise count as one deduction against the shared global filter. In contrast, *r3.ex*'s report includes the full histogram (to give *shoes.ex* a complete view across intermediaries; see §3.2), overlapping with both *r1.ex* and *r2.ex* and adding another 0.3 of loss to both *shoes.ex*'s filter and the global filter. A similar optimization applies to cross-advertiser reports from (b). These are funded from separate filters (those of *r1.ex* and *r2.ex*), so each incurs 0.3 loss. But against the global filter, they again count as one, bringing the total global filter deduction to 0.9 instead of the unoptimized 1.5.

We next formalize this optimization, whose logic we encapsulate in the `attributionObject`. This object dynamically optimizes budget deduction across the per-site, global, and quota filters on each `getReport()` call, on the basis of prior invocations and deductions.

## D.2 Privacy proof

We formalize the cross-report optimization with a series of definitions, after which we analyze its sensitivity properties through a set of intermediary lemmas that ultimately lead to our main correctness result in Thm. 11.

**Definition 7** (Histogram attribution function)**.** *Let $A$ be an attribution function $A : \mathbf{F} \in \mathcal{P}(\mathcal{I})^k \to A(\mathbf{F}) \in \mathbb{R}^m$, for a fixed $k$ and $m$. $A$ is a histogram attribution function, if for any vector of sets of impressions* $\mathbf{F}$*, the following conditions all hold:*

1. *There exists $a_{\mathbf{F}} : \mathcal{I} \to \mathbb{R}_+$ a function that attributes a value to each impression $f \in \mathcal{I}$ depending on all the other impressions* $\mathbf{F}$.
2. *There exists a one-hot encoding function $H$ that maps each event $f$ to one of $m$ buckets. That is, $H : \mathcal{I} \to \{0,1\}^m$ such that $\forall f \in \mathcal{I}, ||H(f)||_1 = 1$.*
3. *$A(\mathbf{F}) = \sum_{f \in \mathbf{F}} a_{\mathbf{F}}(f) \cdot H(f)$*

**Definition 8** (Query partitions for beneficiary sites)**.** *Let $A$ be a histogram attribution function $A : \mathcal{P}(\mathcal{I})^k \to \mathbb{R}^m$, for a fixed $k$ and $m$ and with an associated one-hot encoding function $H : \mathcal{I} \to \{0,1\}^m$. Fix $n \in \mathbb{N}_+$. Take $\sum_{i\in[n]} m_i = m$ and an isomorphism $\Psi : \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_n} \to \mathbb{R}^m$ that divides the histogram of size $m$ into $n$ histograms of size $m_j$ each. For instance, $\Psi$ can be the concatenation map:*

$$\begin{aligned}&\Psi((x_{1,1}, ..., x_{1,m_1}), ..., (x_{n,1}, ..., x_{n,m_n})) \\ &\quad = (x_{1,1}, ..., x_{1,m_1}, ..., x_{n,1}, ..., x_{n,m_n})\end{aligned}$$

*We define the following:*

- *A set of one-hot encoding functions $H_1, H_2, ..., H_n$ where $H_j : S_j \subset \mathcal{I} \to \{0,1\}^{m_j}$ is a* valid partition set *of encoding $H$ if $\Psi(H_1, H_2, ..., H_n) = H$ and $S_1 \sqcup \cdots \sqcup S_n$. That is, each impression $f \in \mathcal{I}$ appears in $S_j$ for exactly one $j \in [n]$, and gets mapped to $j$'s portion of the $m$-sized histogram.*
- *$A_j$ is the* attribution partition *of $A$ for some partition*

$j \in [n]$ *where*

$$A_j : \mathbf{F} \in \mathcal{P}(\mathcal{I})^k \mapsto \sum_{f \in \mathbf{F}} a_{\mathbf{F}}(f) \cdot \mathbb{1}[f \in S_j] \cdot H_j(f) \tag{96}$$

*Note that $A_j$ is a histogram attribution function as defined in Def. 7.*

- $\rho^j$ *is the* report partition *for $j \in [n]$ for a fixed report identifier $r \in \mathbb{Z}$ where*

$$\rho_r^j(D) = A_j(D_d^E) \tag{97}$$

*Let $H_1, H_2, ..., H_n$ be a valid encoding partition of encoding $H$. Let $B = \{b_1, \ldots, b_n\}$ be a set of beneficiary sites, where each $b_j$ has a corresponding encoding partition $H_j$. Consider a set of report identifiers $R$. We define $Q_j$ as the* query partition *that results by using the attribution partition $A_j$ and the corresponding report partitions $\rho^j$. We have:*

$$Q_j(D) := \sum_{r \in R} \rho_r^j(D) \tag{98}$$

**Lemma 3** (Recovering query partitions from a concatenated query)**.** *Let $H_1, H_2, ..., H_n$ be a valid encoding partition of $H$, with concatenation map $\Psi$. By definition, each $H_j$ has a corresponding subset of impressions $S_j \subseteq \mathcal{I}$. Given $S_1, S_2, ..., S_n$ and $H$, we can recover their corresponding encoding partitions and*

$$\Psi^{-1}(H) = (H_1, H_2, ..., H_n) \tag{99}$$

*Proof.* By definition, each $H_j$ can be constructed by generating $H(f)$ for each $f \in S_j$. Hence, given the subsets $S_1, \ldots, S_n$ and the complete encoding $H$, we can recover $H_1, \ldots, H_n$. □

**Lemma 4** (Individual sensitivity of a report partition)**.** *Fix a report identifier $r$, a device $d_r$, a set of epochs $E_r$, a beneficiary $b_j \in B$, an attribution partition function $A_j$ with encoding partition $H_j$, impression set $S_j$ and the corresponding report partition $\rho^j : D \to A_j(D_{d_r}^{E_r})$. We define*

$$A^{\max} := \max_{\mathbf{F} \in \mathcal{P}(\mathcal{I})^k} \sum_{f \in \boldsymbol{F}} a_{\mathbf{F}}(f) \cdot \mathbb{1}[f \in S_j] \cdot H_j(f) \tag{100}$$

*For a fixed device-epoch record $x = (d, e, F) \in \mathcal{X}$, we have that the individual sensitivity of $\rho_j$ is*

$$\Delta_x(\rho^j) \leq \begin{cases} 0 & \text{if } d \neq d_r, e \notin E_r \text{ or } F \cap S_j = \emptyset \\ \|A_j(F)\|_1 & \text{if } d = d_r \text{ and } E_r = \{e\} \\ 2A^{\max} & \text{if } d = d_r, e \in E_r \text{ and } F \cap S_j \neq \emptyset \end{cases}$$

*Proof.* Follows directly from theorem 4 of [26], with the upper bound on histogram report sensitivity from theorem 18 of [26]. □

**Lemma 5** (Individual sensitivity of a concatenated report)**.** *Fix a report identifier $r$, a device $d$, a set of epochs $E_r$, a histogram attribution function $A$ and a set of beneficiaries $B = \{b_1, b_2, ..., b_n\}$. Suppose each beneficiary has a corresponding report partition that results from histogram attribution function $A$ with valid encoding partitions $H_1, H_2, ..., H_n$ (i.e. $\rho_j$ is the report partition for beneficiary $b_j$ with encoding partition $H_j$). For a fixed device-epoch record $x = (d, e, F) \in \mathcal{X}$, the individual sensitivity of the concatenated report $\rho$ is*

$$\Delta_x(\rho) \leq \begin{cases} 0 & \text{if } d \neq d_r, e \notin E_r \\ \|A(F)\|_1 & \text{if } d = d_r \text{ and } E_r = \{e\} \\ 2A^{\max} & \text{if } d = d_r, e \in E_r \end{cases}$$

*Proof.* Follows directly from theorem 4 of [26], with the upper bound on histogram report sensitivity from theorem 18 of [26]. □

**Theorem 11** (Cross-report privacy loss optimization)**.** *Let $H_1, H_2, ..., H_n$ be a valid encoding partition of $H$. Let $B = \{b_1, \ldots, b_n\}$ be a set of beneficiary sites, where each $b_j$ has a corresponding encoding partition $H_j$. Consider a set of report identifiers $R$. Let each beneficiary site $b_j$ also have a corresponding query partition $Q_j$ and the corresponding report partitions $\rho_r^j$ for $r \in R$ and attribution partition $A_j$. Let $Q_1, Q_2, ..., Q_n$ be histogram query partitions and let $\Psi$ be the concatenation mapping as defined in Def. 8. Let $Q = \Psi(Q_1, Q_2, ...Q_n)$ be the query that results from jointly processing all $n$ query partitions. Let $\mathcal{M}$ be a mechanism that adds Laplace noise to a query, such that $\mathcal{M}(Q) = Q + X$ where $X \sim \text{Lap}(b)$ is drawn from a multi-dimensional Laplace mechanism. $\mathcal{M}$ costs $\Delta_Q^{\max} b$ privacy budget, where $\Delta_Q^{\max}$ is the maximum sensitivity of the query. We then have that running $\mathcal{M}(Q)$ and then repartitioning into $n$ noised queries using $\Psi^{-1}$ is equivalent to running $\mathcal{M}(Q_1), \mathcal{M}(Q_2), ..., \mathcal{M}(Q_n)$ sequentially. Furthermore, running $\mathcal{M}$ on the concatenated query $Q$ costs $\frac{1}{n}$-times the cost of running $\mathcal{M}$ on each query partition $Q_1, ..., Q_n$ separately. We show that $\mathcal{M}(Q)$ costs $2 \cdot |R| \cdot A^{\max} \cdot b$ privacy budget, whereas running $\mathcal{M}(Q_1), \mathcal{M}(Q_2), ..., \mathcal{M}(Q_n)$ costs a total of $2 \cdot n \cdot |R| \cdot A^{\max} \cdot b$ privacy budget.*

*Proof.* Suppose that $n$ query partitions are first concatenated into one query $Q$, using concatenation mapping $\Psi$ as defined in Def. 8. Then, mechanism $\mathcal{M}$ is applied to $Q$, and the resulting noised queries are partitioned again into $n$ noised queries $Q_1, ..., Q_n$, using $\Psi^{-1}$. We first show that this is equivalent to running $\mathcal{M}$ on each query partition individually and that this is equally differentially private (i.e. if $\mathcal{M}(Q)$ is $\epsilon$-differentially private, then so is $\mathcal{M}(Q_1), \mathcal{M}(Q_2), ..., \mathcal{M}(Q_n)$). By Lem. 3, we can then recover each query partition $Q_1, Q_2, ..., Q_n$ from $Q$ and $\Psi^{-1}(Q) = Q_1, Q_2, ..., Q_n$. Now consider $\mathcal{M}(Q) = Q + X$ for $X \sim \text{Lap}\, b$. Since $\Psi$ is an isomorphism, it commutes with addition. It follows that

$$\begin{aligned} \Psi^{-1}(\mathcal{M}(Q)) &= \Psi^{-1}(Q + X) && (101) \\ &= \Psi^{-1}(Q) + \Psi^{-1}(X) && (102) \\ &= (Q_1, Q_2, ..., Q_n) + (X_1, X_2, ..., X_n) && (103) \\ &= Q_1 + X_1, Q_2 + X_2, \ldots, Q_n + X_n && (104) \end{aligned}$$

where we define $X := X_1, \ldots, X_n$, where each $X_j$ corresponds to noise in $Q_j$, since additive noise applied to the concatenated query remains additive when the query is decomposed. By

Thm. 12 of [26], we have that the sensitivity of $Q$ is such that

$$\Delta_x(Q) \leq \sum_{r \in R} \Delta_x(\rho_r) \tag{105}$$

Similarly, the sensitivity of query partitions is such that

$$\Delta_x(Q_j) \leq \sum_{r \in R} \Delta_x(\rho_r^j) \tag{106}$$

Furthermore, from Lem. 4 and Lem. 5, we have that both $\Delta_x(\rho_r^j)$ and $\Delta_x(\rho_r)$ have an upper bound of $2A^{\max}$ and therefore have the same maximum sensitivity. Therefore, each $X_j \sim \text{Lap}(b)$, which is exactly what we would expect for $\mathcal{M}(Q_j)$. Therefore, each noised query $Q_j + X_j$ can be recovered as part of $\Psi^{-1}(\mathcal{M}(Q))$ and

$$\Psi^{-1}(\mathcal{M}(Q)) = \mathcal{M}(Q_1), \mathcal{M}(Q_2), \ldots, \mathcal{M}(Q_n) \tag{107}$$

Next, we can apply the post-processing property of differential privacy, because $\Psi$ is a deterministic function. Therefore, if $\mathcal{M}(Q)$ is $\epsilon$-differentially private, then so is $\mathcal{M}(Q_1)$, $\mathcal{M}(Q_2), \ldots, \mathcal{M}(Q_n)$.

We have therefore shown that running $\mathcal{M}(Q)$ and then repartitioning it using $\Psi^{-1}$ is equivalent to running $\mathcal{M}(Q_1)$, $\mathcal{M}(Q_2), \ldots, \mathcal{M}(Q_n)$.

We now show that the privacy budget cost $\epsilon_Q$of $\mathcal{M}(Q)$ is $\frac{1}{n}$ times the privacy budget cost $\epsilon_{Q_1,\ldots Q_n}$ of $\mathcal{M}(Q_1)$, $\mathcal{M}(Q_2)$, $\ldots, \mathcal{M}(Q_n)$. We use the individual sensitivity of $Q$ as defined in Thm. 12 of [26] to get the privacy budget cost $\epsilon_Q$ of $\mathcal{M}(Q)$. Let $\Delta_x^{\max}(\rho_r)$ denote the maximum sensitivity of a report. We get the maximum sensitivity using Lem. 5 as follows:

$$\Delta_Q^{\max} = \sum_{r \in R} \Delta_x^{\max}(\rho_r) \tag{108}$$

$$= 2 \cdot |R| \cdot A^{max} \tag{109}$$

We then get a privacy budget cost of

$$\epsilon_Q = 2 \cdot |R| \cdot A^{max} \cdot b \tag{110}$$

Suppose now that $\mathcal{M}$ is run on each query partition $Q_1, Q_2, ..., Q_n$ sequentially in isolation. The privacy cost would use the maximum individual sensitivity of each query partition as defined in Thm. 12 of [26]. By Lem. 4, we get the maximum sensitivity as follows

$$\Delta_{Q_j}^{\max} = \sum_{r \in R} \Delta_x^{\max}(\rho_r) \tag{111}$$

$$= 2 \cdot |R| \cdot A^{max} \tag{112}$$

Running $\mathcal{M}$ on each query $Q_j$ would therefore cost:

$$\epsilon_j = 2 \cdot |R| \cdot A^{\max} \cdot b \tag{113}$$

By sequential composition, running $\mathcal{M}$ on all queries would the following cost privacy budget $\epsilon_{Q_1,\ldots,Q_n}$

$$\epsilon_{Q_1,\ldots,Q_n} = \sum_{j \in [n]} \epsilon_j \tag{114}$$

$$= 2 \cdot n \cdot |R| \cdot A^{\max} \cdot b \tag{115}$$

$$= n\epsilon_Q \tag{116}$$

□

# E Prototype Screeenshot

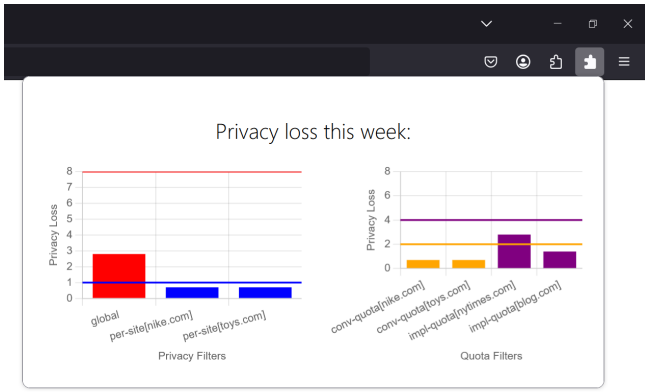



**Fig. 5. Firefox privacy loss dashboard**.

Fig. 5 shows a screenshot of our Firefox extension that serves as a dashboard for visualizing privacy loss across the different filters and quotas Gordon maintains. The screenshot follows a scenario of user visits and purchases, which we emulate programmatically on our local browsers, since no site currently invokes the PPA API. The scenario is as follows: A user visits many websites that display ads on them, such as nytimes.com and blog.com. These websites store every ad view as an event using `saveImpression()`. The user then purchases products for which they have seen ads, including on nike.com and toys.com. At time of purchase, these websites call `measureConversion()` to generate and send a report, consuming privacy in the process. The user wants to check how much of their privacy budget has been spent using the dashboard in Fig. 5.