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 filter 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 [5], introduced the first formal framework based on individual differential privacy (individual DP) [1] to systematically analyze and optimize these APIs—a framework later adopted by Google in privacy analysis of its ARA API [**ara-paper**]. That same Cookie Monster framework now underpins Privacy-Preserving Attribution (PPA) [**ppa**], the API standard being drafted by Private Advertising Technology Working Group (PATWG), a W3C working group that includes representatives from all browsers [**patwg**]. 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 foundation to drive solutions.

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-share scheduling [**drf**, 2], in this new domain. First, 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. Second, recognizing

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

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

that quotas can underutilize the global budget in benign conditions, we explore *batched operation*—periodically collecting and scheduling report requests—to improve utilization while maintaining DoS protection. We use a fair-share-inspired algorithm, showing how OS scheduling research can address foundational gaps in emerging privacy solutions. 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. Results show that: TODO(Pierre): (1) to occupy the paragraph for space budgeting, I add some lorem ipsum; (2) Lorem ipsum dolor sit amet, consectetur adipiscing elit; (3) Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua; and (4) Ut enim ad minim veniam.

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.

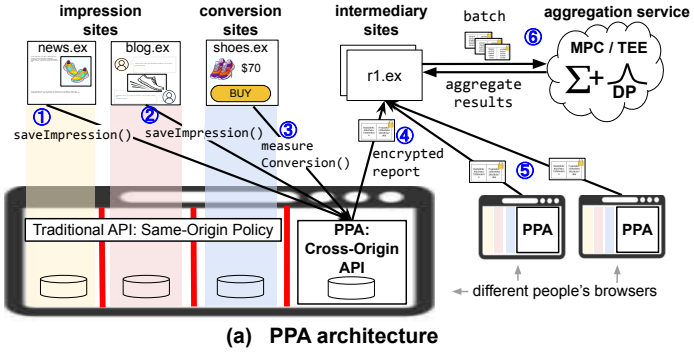
2.2 Example workflow

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 `attribution-Object.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



(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)`

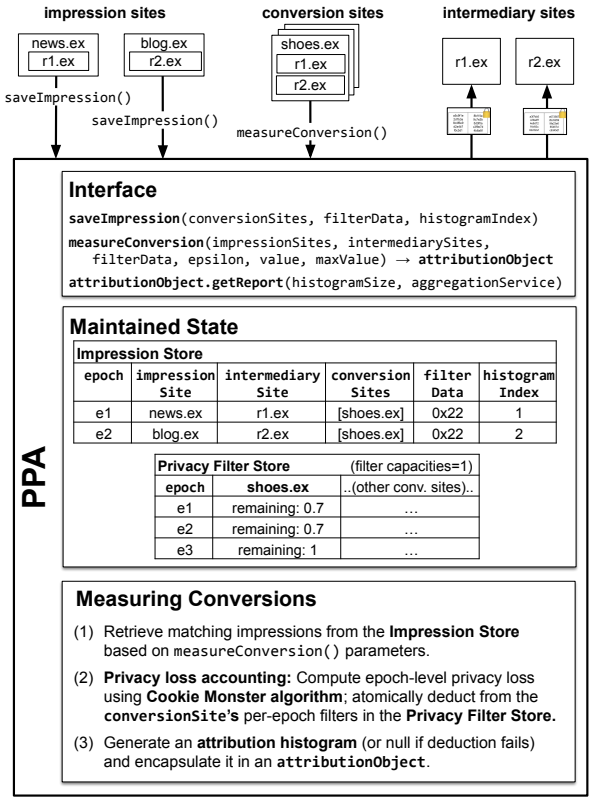
③ **Epoch e3:** User visits *shoes.ex* and purchases $60 of advertised shoes. *Shoes.ex* invokes:
`measureConversion(impressionSites=[blog.ex,news.ex], intermediarySites=[r1.ex,r2.ex], filterData=0x22, epsilon=0.5, value=60, maxValue=100) → attributionObject`
AttributionObject contains the full attribution 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.

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 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 [**cm-paper**], 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) attempt atomic deduction of 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 [3]. 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" [**ppa**].

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 and for two settings: the current PPA design (§4.2) and an extended version that affords more efficient mechanisms (§4.3).

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.

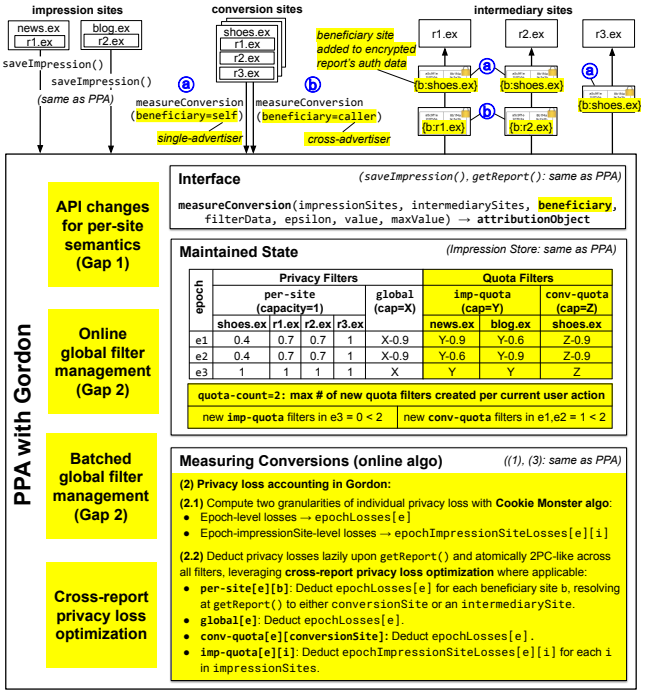


Fig. 2. Gordon architecture. Changes vs. PPA (Fig. 1(c)) in yellow. XXX

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): 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.

Online global filter management (Gap 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 global filter management (Gap 2): While effective, quotas can underutilize the global filter under benign workloads—much like static resource partitioning in scheduling trades fairness for efficiency. In PPA's current online model, immediate request resolution requires strict quotas to ensure isolation, but relaxing them risks new attacks. To improve utilization without compromising protection, we propose a batched mode: report requests are collected and scheduled periodically (e.g., daily), allowing a scheduler to allocate more of the global budget across pending requests. This recovers efficiency while maintaining DoS resilience, addressing Gap 2 under a modified operational model of PPA.

Cross-report privacy loss optimization: Gordon formalizes a new optimization across filters, building on PATWG insights. When multiple reports stem from the same conversion but have disjoint impression sets—e.g., across intermediaries—they do not increase privacy loss against a shared filter, as they reveal no more than the original attribution histogram toward the entity associated with that shared filter. Gordon exploits this to optimize accounting across both privacy and quota filters. Appendix D formalizes the approach; §D.1 explains how it determines the filter states in Fig. 2.

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)—all of which we assume honest-but-curious sites avoid.

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`, `conv-quota`

, 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 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()` call, Gordon attempts to deduct losses across all relevant filters in an atomic transaction per epoch: success only alters state if all checks pass. A non-null report is returned only if all checks succeed across all epochs. For each epoch `e`, 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.

"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.

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 ϵ^{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_{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^{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 ϵ^{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 `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 $M^{\text{adv}} = \frac{1}{n+1}$`quota-count` $\times\, U^{\text{adv}}$ and $N^{\text{adv}} = \frac{n}{n+1}$`quota-count` $\times\, U^{\text{adv}}$. Applying Thm. 1 concludes the proof. □

4.3 Batched global filter management (Gap 2)

As in classic scheduling, static resource partitioning—created by our quota system—can lead to underutilization of the global filter, even in benign cases. Consider a device that visits only two impression sites: *news.ex*, which serves ads for

many conversion sites, and *blog.ex*, which serves just one. Due to the per-site filter, *blog.ex* 's lone conversion site can only consume a small fraction of the global budget ($\epsilon_{\text{per-site}}$). Meanwhile, conversions tied to impressions from *news.ex* are capped at its quota ($\epsilon_{\text{imp-quota}}$), even if they support many conversion sites. If these are the only impression sites visited, much of the global budget remains unused—despite no added privacy risk in letting *news.ex* consume more. This underutilization especially impacts major sites.

In PPA's current online mode, where each report request must be answered immediately, quotas are essential for isolation. But we observe that if PPA supported *batched operation*—collecting requests over time—a scheduler could sort them using a metric that preserves isolation while striving to fully utilize available budget.

Scheduling interval and response time. We divide each epoch's data lifetime into T *scheduling intervals* (e.g., one week each). We extend PPA's API to allow requests to specify a *response time*—the interval after which the response is returned. For privacy, responses are delivered only at that time, regardless of when (or whether) they were scheduled. Requests not scheduled in time get encrypted null reports.

Algorithm. Algorithm 1 outlines the three phases of a scheduling interval: initialization, online, and batch.

(1) Initialization phase: We reset quotas (line 3) and release a portion of the global filter's budget ($\frac{\epsilon_{\text{global}}}{T}$, line 4). This, plus any leftover from the prior interval, is used to schedule as many pending requests as possible (line 5). $TryAllocate(r)$ allocates budget—if successful, it removes the request from the queue; otherwise, the request stays. Default is to check for sufficient remaining budget across all filters and quotas.

Algorithm 1 Batched algorithm

```
1:  function ScheduleBatch(queue)
2:    // (1) Initialization phase:
3:    ResetQuotas()
4:    ReleaseGlobalFilter()
5:    for ∀r ∈ queue do TryAllocate(r)
6:    // (2) Online phase:
7:    // r is newly arriving request
8:    if !TryAllocate(r) then queue ← r
9:    // (3) Batch phase:
10:   // TryAllocateOne ignores quotas
11:   while do
12:     Sort(queue)
13:     flag ← TryAllocateOne(queue)
14:     if flag then Next
15:     else Break
```

(2) Online phase. In the online phase of each interval, requests consume budget by the order of arrival (line 8). If they cannot be allocated budget, they are added to a *queue* and may get another chance at being scheduled in the batch phase, assuming there is still time remaining in their response time.

(3) Batch phase. At the end of the scheduling interval, Gordon tries to utilize all the unused global budget that has been released so far, by scheduling from the remaining requests in the queue that have not already been allocated budget. The queue gets sorted in order to optimize for isolation (or "fairness") across impression sites. Gordon schedules outstanding requests one by one from the queue ($TryAllocateOne$ in line 13 in Algorithm 1, which returns 1 if it allocates one request and 0 if it fails). Each time it schedules a request from the queue, it resorts the queue and tries to allocate a single request again. If it can't schedule any requests from the entire queue, the algorithm is finished, and the scheduling interval is done.

Queue sorting. Gordon by default sorts the requests in the queue based on the impression site that received the least budget so far, and among the requests that belong to the same impression site, it sorts the requests based on their requested privacy budget (from small to large). If a request contains multiple sites we sort based on the smallest budget consumed by any impression site it requests (see Appendix C.1).

Resilience to DoS depletion.TODO(Mathias).

Pareto efficiency. The batched scheduling algorithm aims to maximize global-filter utilization. A key theoretical property supports this goal: the algorithm is Pareto efficient with respect to global filter budget allocation within a device epoch. That is, no request can be additionally scheduled without reducing the allocation of another already-scheduled request. This property applies solely to the batch phase of the batched algorithm, however in §6.5 shows that utilization benefits extend in practice to the end-to-end Gordon system.

Theorem 3 (Pareto efficiency (proof in C.2))**.** *The batched global filter scheduling algorithm is Pareto efficient with regards to the global filter budget within a device epoch.*

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

Roxana: WIP. Do not read.

We implement Gordon in two components: (1) `pdslib`, a generic on-device individual DP library in Rust and (2) its integration into Mozilla Firefox's Private Attribution, a minimal PPA implementation. **`pdslib`:** Mark: New version WIP, do not read... The Rust implementation of the pdslib, which stands for Private Data Service Library, is fully generic. The main functionality of the library, "compute_report", aggregates information over events and consumes privacy losses accordingly to generate a report. We were able to make the two main data structures generic: (1) filters encapsulate filter ID, budget and capacities information that is of Rust generic types, and (2) events contain ID and URI information also of Rust generic types. Then, "compute_report" takes input of a request with associated generic event, epoch ID, and URI types that match the events and associated generic budget type that matches the filters, so that we could implement the logic to select relevant events, compute privacy losses and consume them from the filters, and filter those reports by filter statuses without instantiating any concrete data types at all. Finally, the report that "compute_report" generates is also kept generic. Furthermore, based on these generic data structures and functions, we implemented cross-report privacy loss optimization across reports with the same beneficiary URI (*e.g.,* shoes.ex) but different intermediary URIs (*e.g.,* r1.ex and r2.ex), with an example of this optimization illustrated in figure §3.2. For the demo, we instantiated concrete event and query request data types to be consistent with the PPA proposal [**ppa**] for the purpose of releasing pdslib as a reference for W3C (§1), but the pdslib is implemented to support more general privacy-preserving aggregation use cases beyond advertising.

Firefox integration: TODO(Giorgio).

6 Evaluation

Pierre: WIP. Do not read.

We seek to answer the following questions:

- **Q1:** What parameters define "normal" operation in the Criteo workload?
- **Q2:** In the online setting, how do query error rates vary with different quota capacities?
- **Q3:** Can quotas preserve low error rates for benign queries under DoS attacks?
- **Q4:** Do quotas lead to under-utilization, and can batching mitigate this?

6.1 Methodology

Dataset. We evaluate Gordon on CriteoPrivateAd [4], a dataset released by Criteo specifically to evaluate private advertising systems. The dataset is a sample of 30 days production data collected by Criteo using third-party cookies. It contains 100M impressions displayed on various impression sites (publisher_id feature) for different conversion sites (campaign_id, which is a good proxy for advertiser domain according to [4]). It only involves a single intermediary (Criteo itself). Each impression contains rich features, including a device identifier that is reset every day, contextual and user features, and attribution information such as whether an impression led to a click, a visit or a sale on a conversion site. This attribution information allows us to generate a list of conversions for each device. Since the Criteo dataset is subsampled at the impression level and not device level, most devices contain only a single impression. Criteo provides a device-level distribution of their real traffic, along with a resampling methodology to extract a smaller dataset from CriteoPrivateAd that follows the device-level distribution. We follow their methodology to obtain a dataset with XXX impressions and XXX conversions. We use the first 10 days of the dataset to design our algorithms and calibrate parameters, and we report results on the previously-unseen last 20 days.

Evaluation scenario. Pierre: Each advertiser runs single-advertiser measurements. Histogram queries.

Benign workload process. Pierre: Generate conversions. Keep the 73 conversion sites that have more than 100 reports per day on average. Define the histogram buckets. Calibrate the batch size.

Attack workload process. Pierre: ...

Metric. Pierre: RMSRE, histogram generalization from ARA paper.

Roxana: Move a very short version of the below to the first section that studies error attribution. We further analyze how much each type of filter contributes to the overall error in a query, by introducing an "error cause" metric defined as follows. For each report, we compute how many and which filters were out-of-budget while computing the report. If any per-site filter filter is out-of-budget, we label the

whole report as potentially biased and attribute the error to the per-site filter filter. We look at global filter filters next, then conversion-site quota filters and finally impression-site quota filters. Next, for each query we compute the number of reports in each category, and divide it by the number of reports in the query. This gives us the fraction of reports in the query affected by the per-site filter (resp. global filter, impression-site quota, conversion-site quota) filter. Finally, we average the fractions over all the queries in the workload.

Pierre: Expand on other sources of error, DP error and RMSRE calibration. Bias variance.

Baselines. We compare Gordon against two baselines. The first baseline is **Cookie Monster**, which only maintains per-site filter filters. The second baseline, which we call **PPA**, is Cookie Monster augmented by a global filter filter, which corresponds to the current state of the PPA draft specification with the global filter filter as a safety limit.

Defaults. $\epsilon_{\text{per-site}} = 1$.

6.2 "Normal" workload parameters in Criteo (Q1)

In §4.2 we established sufficient conditions under which filters do not harm normal operation, thanks to parameters r, N, M and n. In our Criteo workload we have $r = 0$ since we only consider measurement queries, and can thus take $\epsilon_{\text{conv-quota}} = \epsilon_{\text{per-site}}$. However, determining the values of N, M and n is not immediate since it would require executing a benign workload to count how many non-zero reports each device-epoch sent to or from various parties. Pierre: TODO: do that, it's tighter and easier to explain in the text. Keeping rough statistics as a placeholder, but they are overestimations (*e.g.,* count of "authorized" conversion sites for each impression site, instead of actual conversions that happen). Instead, we can derive upper bounds $\tilde{N} \geq N, \tilde{M} \geq M, \tilde{n} \geq n$ by computing simple statistics from a dataset of impressions and conversions, as detailed next. For $\tilde{N}$, we compute a percentile of the distribution of number of unique conversion sites in each device-epoch, and multiply by the maximum attribution window length (2 in Criteo). This gives a crude upper bound on the maximum number of times most devices are queried, which is itself an upper bound on the maximum number of times most devices are queried *with non-zero loss*. For $\tilde{M}$, we take a percentile of the number of unique impressions sites across device-epochs. For $\tilde{n}$, we take a percentile of the number of unique conversion sites across impression sites and device-epochs, and multiply by the maximum attribution window length.

Percentile	$\tilde{N}$	$\tilde{M}$	$\tilde{n}$	ϵ_{global}	$\epsilon_{\text{imp-quota}}$
50	2	1	2	2	2
90	4	2	4	8	4
95	4	2	4	8	4
99	6	3	6	18	6
100	12	7	14	98	14

Tab. 2. Percentile values for $\tilde{N}, \tilde{M}$, and $\tilde{n}$.

Finally, we take $\epsilon_{\text{conv-quota}} = \epsilon_{\text{per-site}}$, $\epsilon_{\text{imp-quota}} = \tilde{n}\epsilon_{\text{per-site}}$ and $\epsilon_{\text{global}} = \max(\tilde{N}, \tilde{n} \cdot \tilde{M})\epsilon_{\text{per-site}}$. For instance, taking values at the 95th percentile gives $\epsilon_{\text{global}} = 8$, $\epsilon_{\text{imp-quota}} = 4$, $\epsilon_{\text{conv-quota}} = 1$.

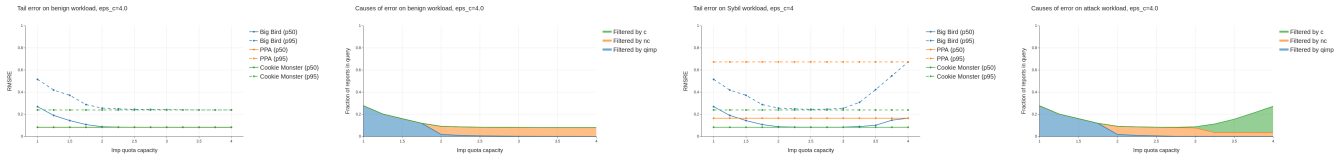
6.3 Query errors under normal workload (Q2)

We fix $\epsilon_{\text{global}} = 4$, a value that is large enough to accommodate all the queries without triggering the global filter filter. Pierre: TODO: rerun experiments with $\epsilon_{\text{global}} = 8$ or whatever we get from Q1. We vary $\epsilon_{\text{imp-quota}}$ and measure its impact on query error. Fig. 3a reports both median and 95th percentile RMSRE. Since Cookie Monster and PPA lack a impression-site quota filter, their errors remain constant across $\epsilon_{\text{imp-quota}}$ values. Moreover, Cookie Monster and PPA exhibit identical error, as we set ϵ_{global} high enough to deactivate PPA's global filter filter—effectively reducing it to Cookie Monster. In contrast, Gordon 's error increases at low $\epsilon_{\text{imp-quota}}$, as the impression-site quota filter blocks some reports. For $\epsilon_{\text{imp-quota}} \geq 2$, the filter no longer affects benign-query error, suggesting that reasonably-configured quotas can preserve utility.

Fig. 3b attributes the error to each filter type, for Gordon. Roxana: I had initially misunderstood the y axis. Explain this graph and the process of attributing error here, not in methodology, as people lose track. Mention just extremely briefly that errors in queries can come from multiple sources in Gordon: DP itself, but also filters and quotas. Remind them how quotas/filters OOB add bias into the query results (the may not know this as we never reminded them of bias etc.! Don't say too much, just find a very simple, intuitive way of explaining it in one short sentence). Say that the graph is showing attribution of errors that come from the filters or quotas. Pierre: Add Cookie Monster and PPA for reference as numbers in the text? They only have per-site filter reports, the same as Gordon. For $\epsilon_{\text{imp-quota}} = 1$, almost a third of reports in each query are filtered out by the impression-site quota filter. This explains the high error seen at low values of $\epsilon_{\text{imp-quota}}$ in Fig. 3a. For $\epsilon_{\text{imp-quota}} = 2$ and higher, almost all the biased reports come from per-site filter filters. These filters also exist in Cookie Monster and PPA, which explains why Gordon's error converges to Cookie Monster's error in Fig. 3a.

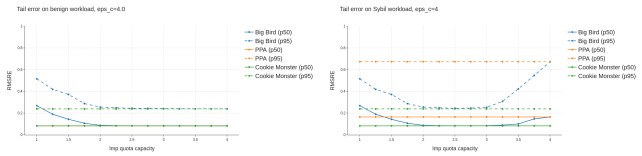
6.4 Query errors under DoS attack (Q3)

We augment the benign scenario with a DoS depletion attack akin to Attacks 2 and 3 (since our dataset does not permit cross-advertiser queries for more than one intermediary site). First, we create a malicious impression site X, by duplicating the impression site X^* that had the most impressions in Criteo. This ensures that a non-negligible fraction of the devices visit X at some point, thus enabling the attack to have visible effect on query accuracy. Next, we create new malicious conversion sites $Y_1^1, \ldots, Y_1^s, \ldots, Y_k^1, \ldots, Y_k^s$ by duplicating s times each of the $k = 10$ conversion sites $Y_1^*, \ldots, Y_k^*$ with most conversions attributed to X^* in Criteo. Finally, we create impressions for X by copying all the impressions X^* has



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

Fig. 3. Online 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. Roxana: Changes to graphs: (1) Move the legends inside the graphs (there's space on top given the [0,1] y axis (which you should keep!). You can order the legend entries 2x2 to fit atop each graph. (2) Enlarge all fonts and line widths and point sizes significantly so they become visible when areas are squeezed. (3) Remove graph titles. (4) Turn everything into black-and-white – no color differentiations pls! You can differentiate based on point types (consistent across the colors hence system types). For area graphs, you can use different textures in place of colors. (5) Make the PPA points much larger than the Cookie Monster points so you can see them from behind CM. (6) Rename Big Bird to Gordon. (7) Label Cookie Monster as "Cookie Monster $\epsilon_{\text{global}} = \infty$", PPA as "PPA $\epsilon_{\text{global}} = 4$, Gordon as "Gordon $\epsilon_{\text{global}} = 4$". (8) x axis label: "Impression site quota capacity ($\epsilon_{\text{imp-quota}}$). x axis label: RMSRE becomes "RMSRE (error, lower is better)"; "Fraction of reports in query" becomes "% of reports in query impacted by a filter." For (b) and (d), legend should say: "per-site filter, "global filter," "quota filter". (9) Please transform y axis for (b) and (d) to 0-100%, to not confuse it with the other 0-1 RMSRE axes. Pierre: Also probably combine all the 4 graphs as a single pdf on the Python side, to share captions etc.



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

Fig. 4. Batched system evaluation. Roxana: BOGUS FIGURES, DO NOT LOOK.

for $Y_1^*, \ldots, Y_s^*$, and we create conversions for Y_i^j by copying all the conversions that Y_i^* has for X. $s = 1$ corresponds to Attack 3, since conversions happen over time, following the same pattern as the benign conversions $Y_1^*, \ldots, Y_k^*$ have for X^*. A larger value for s corresponds to Attack 2, since we have a chain of s conversion sites $Y_i^1, \ldots, Y_i^s$ (using automatic redirection or tricking the user into clicking) instead of a single conversion site Y_i^*.

Fig. 3c shows that impression-site quota limits the impact X has on the c-filter, for an attack with $s = 10$. We do not include X's queries in the error. For a well-configured impression-site quota filter, honest queriers are perfectly isolated from X, and are still able to run their own queries without being perturbed by the impression-site quota.

Fig. 3d further drills down into the causes for high error at both ends of the graph in Fig. 3c. For low $\epsilon_{\text{imp-quota}}$, the error comes from the impression-site quota filter, as in Fig. 3b. The attack has no direct impact on error, but impression-site quota is hurting even honest queries. For high $\epsilon_{\text{imp-quota}}$, the error comes from the global filter filter. This is because the impression-site quota filter is too loose and lets X deplete the global filter filter, thereby denying reports for honest queriers.

6.5 Batched system evaluation (Q4)

7 Related Work

Roxana: WIP. Do not read.

8 Conclusions

References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

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, to better align with PPA. 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. [1]
- 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. [2]

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* ϵ_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(ϵ)*: Returns* TRUE *if the filter can accommodate additional privacy loss ϵ. In particular, letting $\mathbb{I}_{pass}$ be how it's defined in lemma 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 8 of Alg. 3, and ϵ_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 [5]).
  - *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(ϵ)*: Deducts privacy loss ϵ 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 [3] *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.* [4]

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 ρ_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 $\mathbb{R}^m$.[5] 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 [5]). 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. 4 and give the global sensitivities of generating reports and answering queries, respectively. Thm. 8 and Thm. 9 give the individual sensitivity versions.

We first start global sensitivity analysis of generating reports.

Theorem 4 (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 ρ. 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 ρ 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 5. *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. 4, 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 6 (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 7. *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 8 (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 ρ 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. 4, for the worst case choice of x_i in general.

□

Finally, we analyze the individual sensitivity of answering queries.

Theorem 9 (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 ρ_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 ρ_{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 Online Global Filter Management (Gap 2)

B.1 Algorithm

Mark: New paragraph added The Cookie Monster handles on-device privacy budget management on the epoch level and sets up only the per-site filters that cannot safeguard the various DoS attacks (§3.1). Gordon refines budget management to the epoch-site level firstly for better utilities and deals with the DoS attacks through its additional privacy filters (Alg. 2). At a given time upon the reception of a query, Gordon checks and consumes from its privacy filters, and filters its report for that query based on the privacy filter budget statuses (Alg. 3). Particularly, it first makes sure all filters have been initialized, namely the global filter, relevant per-site filters, conversion-site quota filters, and impression-site quota filters. Second, it computes epoch-level privacy losses (§C[5]) and site-level privacy losses (Alg. 5) relevant to the current query and reports. Third, it checks and consumes if all filters have sufficient budget for consuming the respective privacy losses (different filters either consume epoch-level or site-level privacy losses according to their types). This third step needs to be done atomically on the report level according to the 2-phase commit (Alg. 4), since, when generating a report for answering the query eventually, the report will be made empty and incur no privacy losses on all the relevant privacy filters, if any of the relevant filters cannot consume its corresponding privacy loss. Finally, Gordon answers the query at the current time step with the filtered reports.

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 [5, Alg. 2] generalizes to adaptively generated data. Another simplification compared to [5] 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 conversion). This is enforcing the constraint on queries mentioned in Case 1 of [5, 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: $F_i \in \{i\} \times \mathcal{I}$.* [8]

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:

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

Algorithm 2 Gordon Notations and Setup

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

Algorithm 3 Gordon Algorithm (on device)

1: **Input**
2: Filter and quota capacities ϵ_{global}, $\epsilon_{\text{per-site}}$, $\epsilon_{\text{imp-quota}}$, $\epsilon_{\text{conv-quota}}$
3: *// Generate report and update on-device budget*
4: **function** GenerateReport$(d, c, b, \mathbf{i}, E, A, \sigma)$
5: **for** $e \in E$ **do**
6: $x \leftarrow (d, e, D_d^e)$
7: **if** $\mathcal{F}_x$ is not defined **then**
8: $\epsilon_{\text{global}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}} \leftarrow$ Capacities$(\epsilon_{\text{per-site}}, N, M, r)$
9: $\mathcal{F}_x \leftarrow$ InitializeFilters$(\epsilon_{\text{global}}, \epsilon_{\text{per-site}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$
10: $F_e \leftarrow D_d^e$
11: *// Compute individual device-epoch-level privacy losses, same as "ComputeIndividualBudget" defined in appendix D of [5]*
12: $\epsilon_x^t \leftarrow$ EpochBudget$(x, d, E, A, \mathcal{L}, \sigma)$
13: $\epsilon_{\mathbf{x}}^{\mathbf{i},t} \leftarrow \{\}$ *// Initialize empty map*
14: **for** $i \in \mathbf{i}$ **do**
15: *// Compute individual device-epoch-site-level privacy losses, Alg. 5*
16: $\epsilon_x^i \leftarrow$ EpochImpSiteBudget$(x, i, d, E, A, \mathcal{L}, \sigma)$
17: $\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i] \leftarrow \epsilon_x^i$
18: *// Atomic filter check and update, Alg. 4*
19: **if** AtomicFilterCheckAndConsume$(\mathcal{F}_x, b, c, \mathbf{i}, \epsilon_x^t, \epsilon_{\mathbf{x}}^{\mathbf{i},t}) =$ FALSE **then**
20: $F_e \leftarrow \emptyset$ *// Empty the report if any filter check fails*
21: $\rho \leftarrow A((F_e)_{e\in E})$ *// Clipped attribution report*
22: **return** ρ

Theorem 10. *Consider $x \in \mathcal{X}$ on device d, with global filter capacity ϵ_{global} and per-site filter capacity $\epsilon_{per\text{-}site}$. The two following properties hold simultaneously:*

1. *$\mathcal{M}$ satisfies individual device-epoch ϵ_{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. Pierre: WIP: proof sketch for now. 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 to show 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_{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:

$$\Pr[\mathcal{M}(D') = v] = \Pr[\mathcal{M}_{\text{pub}}(D') = v_{\text{pub}}] \cdot \prod_{t=1}^{t_{\max}} \Pr[\mathcal{M}_t(D') = v_t | v_{<t}] \tag{26}$$

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

$$\Pr[\mathcal{M}_t(D') = v_t | v_{<t}] = \Pr[\text{AnswerQuery}(Q_t; D', \mathcal{F}_t) = v_t] \tag{27}$$

where the query Q_t and the state of the privacy filters (across all device-epochs) $\mathcal{F}_t$ are functions of past results $v_{<t}$.

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:

ϵ_x^t: epoch-level privacy loss for a particular query
$\epsilon_{\mathbf{x}}^{\mathbf{i},t}$: epoch-site-level privacy loss for a particular query
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 ϵ_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 [6]
  $\mathcal{F}_x^{\epsilon_{\text{conv-quota}}[c]}$.tryConsume($\epsilon_x^t$)
  // Privacy filters for impression sites
  for $i \in \mathbf{i}$ **do**
    // Invididual device-epoch-impression site loss.[7]
    $\mathcal{F}_x^{\epsilon_{\text{imp-quota}}[i]}$.tryConsume $\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i]$)
  return TRUE

$$\Pr[\mathcal{M}_t(D') = v_t | v_{<t}] = \Pr[\sum_{r \in R_t} \rho_r(D; \mathcal{F}_{t,r}) + X_t = v_t] \quad (28)$$

Now we are ready to replace D' by $D + x_C$ on one hand and $D + x$ on the other hand.

$$\Pr[\mathcal{M}(D + x_C) = v] \quad (29)$$

$$= \prod_{t=1}^{t_{\max}} \Pr[\sum_{r \in R_t} \rho_r(D + x_C) + X_t = v_t] \quad (30)$$

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

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 $x \cap C \cap \mathcal{I} = \emptyset$. Pierre: Add filters above, needs some explicit notation maybe, *e.g.*, filtered report $\rho_r(D; \mathcal{F})$ pass could also take $\mathcal{F}$ and the report as arguments instead of an index that refers to some implicit state.

First, by definition of x_C, we observe that all the outputs v_{pub} at Line are identical across both worlds. Pierre: A bit more notation.

Similarly, we get:

$$\Pr[\mathcal{M}(D + x) = v] = \prod_{t=1}^{t_{\max}} \Pr[\sum_{r \in R_t} \rho_r(D + x) + X_t = v] \quad (32)$$

$$= \prod_{t=1}^{t_{\max}} \Pr[\sum_{r \in R_t^x} \rho_r(D + x) + \sum_{R_t \setminus R_t^x} \rho_r(D) + X_t = v] \quad (33)$$

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

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
σ : 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 8
  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 4

  return Δ / σ

Take a report $r \in R_t^x$. Recall that 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 20. Note that pass_r can happen even if $\mathcal{F}^{\text{global filter}}$ has enough budget, for instance if per-site filter or a quota is out of budget.
- If $\text{pass}_r = 1$, we have $\|\rho_r(D + x) - \rho_r(D)\| \leq \Delta_x \rho_r \leq \sigma \cdot \epsilon_r \text{pass}_r$ 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 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.,*

$$\sum_{t=1}^{t_{\max}} \sum_{r \in R_t} \epsilon_r \text{pass}_r \leq \epsilon_{\text{global}} \tag{34}$$

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

$$\left| \ln \left(\frac{\Pr[\sum_{r \in R_t} \rho_r(D) + X_t = v_t]}{\Pr[\sum_{r \in R_t} \rho_r(D + x) + X_t = v_t]} \right) \right| \leq \Delta_x Q_t / \sigma \tag{35}$$

$$\leq \sum_{r \in R_t} \Delta_x \rho_r / \sigma \tag{36}$$

$$\leq \sum_{r \in R_t} \epsilon_r \text{pass}_r \tag{37}$$

We conclude by triangle inequality. Pierre: Clean up and reorder things, I'm not so happy with this proof.

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

$$\left| \ln \left(\frac{\Pr[\mathcal{M}_b(D + x_C) = v]}{\Pr[\mathcal{M}_b(D + x) = v]} \right) \right| \leq \epsilon_{\text{per-site}} \tag{38}$$

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:

$$\Pr[\mathcal{M}_b(D + x_C) = v] \tag{39}$$

$$= \prod_{t \in [t_{\max}]: b_t = b} \Pr[\sum_{r \in R_t} \rho_r(D + x_C; \mathcal{F}_t) + X_t = v_t] \tag{40}$$

Pierre: I might be overthinking this, but here the weird thing is that the filter depends on events that are not visible to a particular beneficiary (*e.g.,* queries from $b' \neq b$ that deplete the global filter in the world where x is present), so we even after conditioning on past events we might end up having different $\mathcal{F}_t$? Even if other queriers ask the exact same queries in both worlds, the state of the filters will be different because in one world x might pay. At least all the filters for $x' \neq x$ are identical in both worlds. Luckily it should all work out, because having different filters can only create more zeros, still within the individual sensitivity. It might help to mention that the per-site filter for b is the same in both worlds.

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

$$\sum_{t \in [t_{\max}]: b_t = b} \sum_{r \in R_t} \epsilon_r \text{pass}_r \leq \epsilon_{\text{per-site}} \tag{41}$$

We conclude with a triangle inequality as before.

□

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:*

- $\epsilon_{global}^{\leq t}$*: the cumulative privacy loss charged to the global filter up to step* t *before 2PC is triggered at step* t*.*
- $D^{\leq e}$*: 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:*

$$pass(k) \triangleq \begin{cases} 1 & \text{if AtomicFilterCheckAndConsume returns} \\ & \quad \textsf{TRUE} \text{ for query } k \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

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* ϵ_x^t *for that query.*
2. ***Epoch-site-level Consistency Property:*** *the impression-site quota filter consumes exactly* $\epsilon_x^i[i]$*, 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 benefi-

ciary site b and impression sites $\mathbf{i}$, the function calls:

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

Note that ϵ_x^t is computed once at EpochBudget in Line 12 of Algorithm 3 and represents the device-epoch-level individual privacy loss. Similarly, each $\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i]$ is computed once via EpochImpSiteBudget and represents the device-epoch-impressionSite-level individual privacy loss. Therefore, when $\text{pass}(k) = 1$, the per-site filter, global filter, and conversion-site quota filters all consume exactly the same amount ϵ_x^t, while each impression-site quota filter consumes its specific amount $\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i]$, 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 σ through conversion site c for impression sites $\mathbf{i}$. Consider a device-epoch x, with individual budget ϵ_x^t computed at Line 12 in Alg. 3. Denote by $N^{\leq t,\text{adv}}$ 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 $M^{\leq t,\text{adv}}$ 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 $N^{\text{adv}}\epsilon_{conv\text{-}quota}$ budget from the global filter.*

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

$$\epsilon_{\text{used}}^{\mathbf{bad}} = \epsilon_{\text{global}}^{\leq t-1,\mathbf{bad}} = \sum_{k=[t-1]:c_k\in\mathbf{bad_c}} \epsilon_{\text{global}}^k \cdot \text{pass}(k). \tag{43}$$

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

$$= \sum_{c\in\mathbf{bad_c}} \sum_{k<t:c_k=c} \Delta_x \rho_{b_k}^k \cdot \text{pass}(k), \tag{44}$$

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

$$\epsilon_{\text{conv-quota}}[c]^{\leq t-1} = \sum_{k<t:c_k=c} \Delta_x \rho_{b_k}^k \cdot \text{pass}(k). \tag{45}$$

Substitute this equality into equation (44), we get:

$$\epsilon_{\text{global}}^{\leq t-1,\mathbf{bad}} = \sum_{c\in\mathbf{bad_c}} \epsilon_{\text{conv-quota}}[c]^{\leq t-1}. \tag{46}$$

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

$$= \sum_{c\in\mathbf{bad_c}:\epsilon_{\text{conv-quota}}[c]^{\leq t-1}>0} \epsilon_{\text{conv-quota}}[c]^{\leq t-1} \tag{47}$$

$$\leq \sum_{c\in\mathbf{bad_c}:\epsilon_{\text{conv-quota}}[c]^{\leq t-1}>0} \epsilon_{\text{conv-quota}}, \tag{48}$$

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:

$$\leq \|\{c \in \mathbf{bad_c} : \epsilon_{\text{conv-quota}}[c]^{\leq t-1} > 0\}\| \cdot \epsilon_{\text{conv-quota}} \tag{49}$$

$$= N^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{conv-quota}}. \tag{50}$$

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

- Suppose ϵ_x^t is a reasonable value, in the sense that it's bounded by the capacity $\epsilon_{\text{conv-quota}}$. Then,
$$\epsilon_{\text{global}}^{\leq t,\mathbf{bad}} = \epsilon_{\text{used}}^{\mathbf{bad}} + \epsilon_x^t \leq N^{\leq t,\text{adv}} \cdot \epsilon_{\text{conv-quota}}. \tag{51}$$
- Otherwise, ϵ_x^t is unreasonable, in which case ϵ_x^t exceeds the capacity $\epsilon_{\text{conv-quota}}$. In this case,
$$\epsilon_{\text{conv-quota}}[c_t]^{\leq t-1} + \epsilon_x^t \geq \epsilon_{\text{conv-quota}}, \tag{52}$$
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,
$$\epsilon_{\text{global}}^{\leq t,\mathbf{bad}} = \epsilon_{\text{used}}^{\mathbf{bad}} + 0 = \epsilon_{\text{used}}^{\mathbf{bad}} \leq N^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{conv-quota}}, \tag{53}$$
by equation (50).

Since the adversary has created at most N^{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^{adv}). By basic composition under a pure DP filter, we know ϵ_{global} is by definition the sum of global filter consumption at each time up to the end of time $t-1$:

$$\epsilon_{\text{global}}^{\leq t-1,\mathbf{bad}} = \sum_{k\in[t-1]:c_k\in\mathbf{bad_c}} \epsilon_x^k \cdot \text{pass}(k) \tag{54}$$

$$= \sum_{k\in[t-1]:c_k\in\mathbf{bad_c}} \Delta_x \rho_{b_k}^k \cdot \text{pass}(k), \tag{55}$$

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:

$$\rho(D + \vec{x}) - \rho(D) = \sum_{j=1}^{m} \rho(D + x_{i_j}) - \rho(D + x_{i_{j-1}}), \quad (56)$$

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 16:

$$\begin{aligned}
\Delta_x \rho^k_{b_k} &= \max_{D, D' \in \mathcal{D}' = D + x} \|\rho^k_{b_k}(D') - \rho^k_{b_k}(D)\|_1 && (57)\\
&= \max_{D \in \mathcal{D}} \|\rho^k_{b_k}(D + x) - \rho^k_{b_k}(D)\|_1 && (58)\\
&\leq \max_{D \in \mathcal{D}} \sum_{j \in [m]: i_j \in \mathbf{bad_i}} \|\rho^k_{b_k}(D + x_1 + \ldots + x_i) - && (59)\\
&\qquad \rho^k_{b_k}(D + x_1 + \ldots + x_{i-1})\|_1 && (60)\\
&\leq \sum_{j \in [m]: i_j \in \mathbf{bad_i}} \max_{\hat{D} \in \mathcal{D}} \|\rho^k_{b_k}(\hat{D} + x_{i_j}) - \rho^k_{b_k}(\hat{D})\|_1 && (61)\\
&= \sum_{j \in [m]: i_j \in \mathbf{bad_i}} \Delta^{i_j}_x \rho^k_{b_k} = \sum_{i \in \mathbf{bad_i}} \Delta^i_x \rho^k_{b_k} && (62)\\
&= \sum_{i \in \mathbf{bad_i}} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i], && (63)
\end{aligned}$$

where the inequality in equation 59 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

$$\begin{aligned}
\epsilon^{\leq t-1, \mathbf{bad}}_{\text{global}} &\leq \sum_{k \in [t-1]: c_k \in \mathbf{bad_c}} \sum_{i \in \mathbf{bad_i}} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k) && (64)\\
&= \sum_{i \in \mathbf{bad_i}} \sum_{k \in [t-1]: c_k \in \mathbf{bad_c}, i \in \mathbf{i}_k} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k), && (65)\\
&\leq \sum_{i \in \mathbf{bad_i}} \sum_{k \in [t-1]: i \in \mathbf{i}_k} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k), && (66)
\end{aligned}$$

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

$$\epsilon^{\leq t-1}_{\text{imp-quota}}[i] = \sum_{k \in [t-1]: i \in \mathbf{i}_k} \epsilon^{\mathbf{i},k}_{\mathbf{x}}[i] \cdot \text{pass}(k), \quad (67)$$

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 (66), we get:

$$\begin{aligned}
\epsilon^{\leq t-1, \mathbf{bad}}_{\text{global}} &\leq \sum_{i \in \mathbf{bad_i}} \epsilon^{\leq t-1}_{\text{imp-quota}}[i] && (68)\\
&= \sum_{i \in \mathbf{bad_i}: \epsilon^{\leq t-1}_{\text{imp-quota}}[i] > 0} \epsilon^{\leq t-1}_{\text{imp-quota}}[i] && (69)\\
&\leq \sum_{i \in \mathbf{bad_i}: \epsilon^{\leq t-1}_{\text{imp-quota}}[i] > 0} \epsilon_{\text{imp-quota}} && (70)\\
&= \left|\left\{i \in \mathbf{bad_i} : \epsilon^{\leq t-1}_{\text{imp-quota}}[i] > 0\right\}\right| \cdot \epsilon_{\text{imp-quota}}, && (71)
\end{aligned}$$

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:

$$\epsilon^{\leq t-1, \mathbf{bad}}_{\text{global}} \leq M^{\leq t-1, \text{adv}} \cdot \epsilon_{\text{imp-quota}}. \quad (72)$$

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

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

$$\epsilon^{\leq t, \mathbf{bad}}_{\text{global}} \leq M^{\leq t, \text{adv}} \cdot \epsilon_{\text{imp-quota}}. \quad (73)$$

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

$$\epsilon^{\leq t, \mathbf{bad}}_{\text{global}} = \epsilon^{\leq t-1, \mathbf{bad}}_{\text{global}} + 0 = \epsilon^{\leq t-1, \mathbf{bad}}_{\text{global}} \leq M^{\leq t-1, \text{adv}} \cdot \epsilon_{\text{imp-quota}}, \quad (74)$$

by equation (72).

Since by the end of time t, the adversary has created at most M^{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:

$$\epsilon^{\leq t, \mathbf{bad}}_{\text{global}} \leq M^{\leq t, \text{adv}} \epsilon_{\text{imp-quota}} \leq M^{\text{adv}} \epsilon_{\text{imp-quota}}, \quad (75)$$

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^{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^{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 ϵ^{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 $M^{\leq t,\text{adv}} \leq M^{\text{adv}}_{\texttt{imp-quota}}$ and $N^{\leq t,\text{adv}} \leq N^{\text{adv}}_{\texttt{conv-quota}}$ filters, then by Lemma 2:

$$\epsilon^{\text{adv}}_{\text{global}} \leq M^{\leq t,\text{adv}} \epsilon_{\text{imp-quota}} \leq M^{\text{adv}} \epsilon_{\text{imp-quota}} \tag{76}$$

$$\epsilon^{\text{adv}}_{\text{global}} \leq N^{\leq t,\text{adv}} \epsilon_{\text{conv-quota}} \leq N^{\text{adv}} \epsilon_{\text{conv-quota}}. \tag{77}$$

Therefore:

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

□

C Batched Global Filter Management (Gap 2)

C.1 Algorithm

Alg. 6 describes the batched algorithm for a single epoch.

Pierre: TODO: update wording to be accurate for cross-epoch. We define $\mathcal{A}, \mathcal{U} \leftarrow \text{TryAllocate}(\mathcal{R})$ as follows. TryAllocate takes a set of report requests $\mathcal{R}$, and executes Alg. 3's GenerateReport function for each request. It then returns two sets: $\mathcal{A}$ the reports for requests that were allocated successfully (*i.e.,* the filters returned TRUE at line Line 19), and $\mathcal{U}$ the remaining unallocated requests. TryAllocateOne behaves like TryAllocate, except that it stops after the first successfully allocated request.

SortBatch attaches a weight (b_r, ϵ_r) to each request r in a batch, given access to filters $\mathcal{F}$, and then sorts by smallest weight first (in lexicographic order). The weights are defined as follows. ϵ_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 σ). b_r is the smallest budget consumed by any impression site $i \in \mathbf{i}_r$ requested by r:

$$b_r := \min_{i \in \mathbf{i}_r} \mathcal{F}^{\text{impression-site quota}}[i].\mathit{consumed} \tag{78}$$

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 Pareto efficiency proof

We prove the batched algorithm's pareto efficiency property under a restricted setting. First, we only consider requests that request budget from a single device epoch. Second, we prove pareto efficiency of the batched phase of the Gordon batched algorithm (Algorithm 6), which operates on the global filter. We do not prove Pareto efficiency for Gordon as a whole, which also manages per-site filters and quotas. Similar to prior work [2], we also assume all-or-nothing utility for DP requests, wherein their utility is equal to one if their entire global filter requested budget is granted, and is equal to zero if it is not.

Theorem 3 [Pareto efficiency. *] The batched global filter scheduling algorithm is Pareto efficient with regards to the global filter budget within a device epoch.*

Algorithm 6 Batched algorithm

Input:

$\epsilon_{\text{per-site}}, N, M, r, n$: same parameters as Alg. 3.

T: number of scheduling intervals in the epoch's data lifetime.

function MAIN
  $\epsilon_{\text{global}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}} \leftarrow \text{Capacities}(\epsilon_{\text{per-site}}, N, M, r, n)$
  $\mathcal{F} \leftarrow \text{InitializeFilters}(\epsilon_{\text{global}}, \epsilon_{\text{per-site}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$
  $\mathcal{F}^{global}.capacity \leftarrow 0$ *// Initially no global budget available*
  $\mathcal{R}_{batch} \leftarrow \emptyset$ *// Requests for the batch phase*
  for $t \in [T]$ **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
  $\mathcal{F}^{global}.capacity \leftarrow \mathcal{F}^{\text{global filter}}.capacity + \epsilon_{\text{global}}/T$
  $\mathcal{F}^{imp-quota} \leftarrow \text{InitializeFilter}(\epsilon_{\text{imp-quota}}/T)$
  $\mathcal{A}_{init}, \mathcal{U}_{init} \leftarrow \text{TryAllocate}(\mathcal{R}_{batch})$
  $\mathcal{A} \leftarrow \mathcal{A}_{init}$
  // 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
  $\mathcal{F}^{imp-quota}.capacity \leftarrow \infty$
  $batch \leftarrow \mathcal{U}_{init} \cup \mathcal{U}_{online}$
  do
    $sorted \leftarrow \text{SortBatch}(batch)$
    $(a, u) \leftarrow \text{TryAllocateOne}(sorted)$
    $batch \leftarrow u$
    $\mathcal{A} \leftarrow \mathcal{A} \cup a$
  while $\mathcal{A} \neq \emptyset$
  return $\mathcal{A}, batch$

Proof. Consider the batch phase with requests that request budget from a single device-epoch. We prove that no allocation from global budget can increase a request's utility without decreasing another request's utility. Consider request i that is in the queue of the batch phase. If i is allocated by Algorithm 6, its utility is one and cannot be further improved. If i has not yet been scheduled after the batch stage, this means it cannot be allocated from any released global filter budget, as Algorithm 6 grants requests until no requests can be allocated. Allocating request i would require extra global filter budget, which can only come from another allocated request. Since each allocated request has exactly its requested global filter budget this would decrease its utility from one to zero, which is not Pareto-improving. □

C.3 Resilience to DoS depletion proof

TODO(Mathias?): Can we prove something under some assumptions? Make the assumptions clear here. Like, single-epoch attribution maybe?

Theorem ?? Resilience to DoS depletion. *] XXX*

Proof. XXX. □

C.4 Response time tradeoff proof

Theorem ?? [Response time tradeoff. *] The longer the requested response time, the higher the probability the request will get allocated privacy budget.*

Proof. We prove by induction that the longer the response time for a request, the higher the probability the request will get allocated privacy budget. The base case ($0 \leq n < l$, where l is the length of a single schedule interval, and where without loss of generality the request is submitted at the beginning of a scheduling interval) is a response time that results in a response within the same scheduling interval as the request. In this case, the request is only granted the opportunity to be scheduled during the online phase. If the response time is one scheduling interval longer ($l \leq n < 2l$) than the base case, the request has the opportunity to be scheduled either: in the online phase of the scheduling interval it was requested in, in the batch phase of that scheduling interval, and in the initialization phase of the next scheduling interval (see Algorithm 6), thereby increasing its probability of getting scheduled. The same property holds true for any response time n that falls within the k next intervals ($k \cdot l \leq n < k \cdot l+1$), where if we increase the response time by one interval ($k \cdot l+1 \leq n < k \cdot l+2$), the request would have two additional opportunities (at the batch phase of scheduling interval k and the initialization phase of scheduling interval $k+1$) to get scheduled, thereby increasing its probability of getting allocated budget. □

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

TODO(Alison): Please capitalize names of things you are referencing: "proposition 1" becomes "Proposition 1" (it's viewed as its name).Pierre: This is automated if you use macros like \Thm, Prop, etc. And actually let's be consistent across appendix sections, we've been using Lem. instead of Prop in previous sections.

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

Definition 7 (Histogram attribution function)**.** *Let A be an attribution function $A : \mathcal{P}(\mathcal{I})^k \to \mathbb{R}^m$, for a fixed k and m. A is a histogram attribution function, if for any vector of a set 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 $H : \mathbb{R}^m \to \{0,1\}^m$. Fix $n \in \mathbb{N}_+$. Let $\mathbf{F}$ be a fixed vector of event sets. We define the following:*

- *H_j is an* encoding partition *of H for some $j \in [n]$ if $H_j : \mathcal{I} \to \{0,1\}^m$ and for some subset of impressions*

$S_j \subseteq \mathcal{I}$ *we have that*

$$H_j(f) = \begin{cases} H(f) & \text{if } f \in S_j \\ 0 & \text{if } f \in \mathcal{I} \setminus S_j \end{cases}$$

- *Pierre: We don't need to support any possible isomorphism, so I'd define Ψ once and for all, and call it "the" concatenation isomorphism, or just "the concatenation map" (we'll use the isomorphism later but it sounds complicated when we mention this upfront).* $\Psi : \{0,1\}^m \times \cdots \times \{0,1\}^m \to \{0,1\}^m$ *is a* concatenation map *for a set of encoding partitions H_j of H if*

$$\Psi(H_1(f), H_2(f), ..., H_n(f)) = H(f) \tag{79}$$

for all $f \in \mathcal{I}$ and we have that $\sum_{j\in[n]} H_j(f) = H(f)$
- *A set of encoding partitions $H_1, H_2, ..., H_n$ is a* valid partition set *of encoding H if there exists a concatenation isomorphism Ψ such that $\Psi(H_1, H_2, ..., H_n) = H$ and for all $f \in \mathcal{I}$ we have that $||H_j(f)||_1 = 1$ for exactly one $j \in [n]$.*
- *A_j is the* attribution partition *of A for some $j \in [n]$ where*

$$A_j(\mathbf{F}) := \sum_{f\in\mathbf{F}} a_{\mathbf{F}}(f) \cdot H_j(f) \tag{80}$$

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

$$\rho^j_r(D) = A_j(D^E_d) \tag{81}$$

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 ρ^j. We have:*

$$Q_j(D) := \sum_{r\in R} \rho^j_r(D) \tag{82}$$

Proposition 1 (Recovering encoding partitions from a concatenated encoding)**.** *Let $H_1, H_2, ..., H_n$ be a valid encoding partition of encoding H, with concatenation map Ψ. 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*

$$\Psi^{-1}(H) = (H_1, H_2, ..., H_n) \tag{83}$$

Pierre: So here we need one more step saying that given noisy results for Q, we can recover noisy results for each Q_i, when the noise is the multidimensional Laplace mechanism with some parameter Lap b*. This is where we need Ψ to be an isomorphism and not just a bijection: we want the additive noise to go through (would be nice to be a bit extra verbose here about how the noise also gets decomposed into coordinates). Then, add another point saying that the mechanism $\mathcal{M}_1, \ldots, \mathcal{M}_n$ that jointly releases the n queries can be obtained by post-processing $\mathcal{M}$ with Ψ^{-1}. And by post-processing property of DP, if $\mathcal{M}$ is ϵ-DP, then $\Psi^{-1}(\mathcal{M}) = \mathcal{M}_1, \ldots, \mathcal{M}_n$ is also ϵ-DP.*

Proof. By definition, each H_j can be constructed by outputing $H(f)$ for each $f \in S_j$ and 0 for all $f \in \mathcal{I} \setminus S_j$. □

Proposition 2 (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 $\rho^j : D \to A_j(D^{E_r}_{d_r})$. We define*

$$A^{\max} := \max_{\mathbf{F}\in\mathcal{P}(\mathcal{I})^k} \sum_{i=1}^{k} \sum_{f\in\boldsymbol{F}} a_{\mathbf{F}}(f) \cdot H(f) \tag{84}$$

For a fixed device-epoch record $x = (d, e, F) \in \mathcal{X}$, we have that the individual sensitivity of ρ_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 [5], with the upper bound on histogram report sensitivity from theorem 18 of [5]. □

Proposition 3 (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. ρ_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 ρ 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 [5], with the upper bound on histogram report sensitivity from theorem 18 of [5]. □

Proposition 4 (Individual sensitivity of query partitions Q_j)**.** *Fix an impression set $S_j \subseteq \mathcal{I}$. Let Q_j be the query partition of beneficiary site $j \in [n]$ with partition encoding H_j. Fix a device-epoch record $x = (d, e, F) \in \mathcal{X}$.*

$$\Delta_x(Q_j) \leq \sum_{r\in R} \Delta_x(\rho^j_r) \tag{85}$$

Proof. Take $D, D' \in \mathcal{D}$ such that $D' = D + x$. We have that

$$\Delta_x(Q_j) = \max_{D'=D+x_i \in \mathcal{D}} \|Q_j(D) - Q_j(D')\|_1 \tag{86}$$

$$= \max_{D'=D+x \in \mathcal{D}} \left\| \sum_{r \in R} \rho_r^j(D) - \rho_r^j(D') \right\|_1 \tag{87}$$

$$\leq \sum_{r \in R} \max_{D'=D+x \in \mathcal{D}} \left\| \rho_r^j(D) - \rho_r^j(D') \right\|_1 \tag{88}$$

$$\leq \sum_{r \in R} \Delta_x(\rho_r^j) \tag{89}$$

by the definition of individual sensitivity and the triangle inequality. □

Proposition 5 (Sensitivity of concatenated query Q)**.** *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 ρ_r^j for $r \in R$ and attribution partition A_j. Let $Q = Q_1, Q_2, ..., Q_n$ be the query that results from jointly processing all queries of each beneficiary site. Fix a device-epoch record $x = (d, e, F) \in \mathcal{X}$. We have that the sensitivity of Q is such that*

$$\Delta_x(Q) \leq \sum_{r \in R} \Delta_x(\rho_r) \tag{90}$$

Proof. We first notice that for a fixed r we have that $\sum_{j=1}^n \rho_r^j(D) = \rho_r(D)$. This follows directly from our definition of a valid encoding partition where $H_j(f) = H(f)$ for all $f \in \mathbf{F}$ and for all $j \in [n]$.

$$\sum_{j=1}^n \rho_r^j(D) = \sum_{j=1}^n A_j(D_d^E) \tag{91}$$

$$= \sum_{j=1}^n \sum_{f \in \mathbf{F}} a_{\mathbf{F}}(f) \cdot H_j(f) \tag{92}$$

$$= \sum_{f \in \mathbf{F}} a_{\mathbf{F}}(f) \cdot H(f) \tag{93}$$

$$= \rho_r(D) \tag{94}$$

Now, take $D, D' \in \mathcal{D}$, such that $D' = D + x$. We have that

$$\Delta_x(Q) = \max_{D'=D+x \in \mathcal{D}} \|Q(D) - Q(D')\|_1 \tag{95}$$

$$= \max_{D'=D+x \in \mathcal{D}} \| \sum_{j=1}^n Q_j(D) - Q_j(D')\|_1 \tag{96}$$

$$= \max_{D'=D+x \in \mathcal{D}} \| \sum_{j=1}^n \sum_{r \in R} \rho_r^j(D) - \rho_r^j(D')\|_1 \tag{97}$$

$$= \max_{D'=D+x \in \mathcal{D}} \| \sum_{r \in R} \sum_{j=1}^n \rho_r^j(D) - \sum_{j=1}^n \rho_r^j(D')\|_1 \tag{98}$$

$$= \max_{D'=D+x \in \mathcal{D}} \| \sum_{r \in R} \rho_r(D) - \rho_r(D')\|_1 \tag{99}$$

$$\leq \sum_{r \in R} \max_{D'=D+x \in \mathcal{D}} \|\rho_r(D) - \rho_r(D')\|_1 \tag{100}$$

$$\leq \sum_{r \in R} \Delta_x(\rho_r) \tag{101}$$

□

Theorem 11 (Cross-report privacy loss optimization)**.** *Consider the setting described in Proposition 5. Pierre: Re-emphasize that we are considering the specific class of queries defined in Def. .... Also give the actual value in the theorem instead of just saying that it's "less". Processing query partitions $Q_1, Q_2, ..., Q_n$ jointly deducts less global filter budget than processing them individually. Pierre: Here we need to talk about DP mechanisms $\mathcal{M}_1$ instead of the underlying queries. Also would be good to be consistent, I don't think we use "processing" too much elsewhere, maybe we've been saying "answering" a query, to check with other propositions and algorithms.*

Proof. Suppose queries partitions $Q_1, Q_2, ..., Q_n$ were processed sequentially in isolation by algorithm 3. The global filter deduction would use the individual sensitivity of each query partition as defined in proposition 4. Given a standard deviation σ of the Laplace distribution $\mathcal{L}$, each query Q_j would deduct from the budget:

$$\epsilon_j = \sum_{r \in R} \Delta_x(\rho_r) \cdot \frac{\sqrt{2}}{\sigma} \tag{102}$$

~~In total~~By sequential composition, processing all queries would deduct the following. We use proposition 2 to show an upper bound.

$$\epsilon_{Q_1,...,Q_n} = \sum_{j \in [n]} \epsilon_j \cdot \frac{\sqrt{2}}{\sigma} \tag{103}$$

$$\leq n \sum_{r \in R} 2A^{\max} \cdot \frac{\sqrt{2}}{\sigma} \tag{104}$$

$$\leq 2n|R|A^{\max} \cdot \frac{\sqrt{2}}{\sigma} \tag{105}$$

Suppose now that query partitions are first jointly processed

all query partitions as one query Q. By proposition 1, we can then recover each query partition $Q_1, Q_2, ..., Q_n$ and distribute them among the beneficiary sites. Pierre: Woud be good to plug the beefed up version of Prop 1 with the formal post-processing argument I sketched. Alison: Is it clear how you can do this? Should I provide more details?. Roxana: I don't know, didn't get to read. But giving all the details on a thing that we do that's new is important. The global filter deduction would use individual sensitivity concatenated query sensitivity as defined in proposition 5 as follows:

$$\epsilon_Q = \sum_{r \in R} \cdot \frac{\sqrt{2}}{\sigma} \Delta_x(\rho_r) \tag{106}$$

By proposition 3, we can bound this as follows

$$\epsilon_Q \leq 2|R|A^{\max} \cdot \frac{\sqrt{2}}{\sigma} \tag{107}$$

It clearly follows that $\epsilon_{Q_1,...,Q_n} \leq \epsilon_Q$.

□

E Prototype Details

TODO(Mark, Giorgio): Add here any further details that you think PATWG should know about your components.

TODO(Nikos/Giorgio): Add here the screenshot. Describe it in text – scenario, what it shows etc.

XXX It would be good if this section didn't only have the screenshot but more details from implementation in which the screenshot is embedded.

F Evaluation Methodology Details

§6.1 gives a high-level overview of our methodology, covering the necessary details for result assessment. Here, we give further details that may be necessary for reproduction. We also intend to release our evaluation repository as part of artifact release.