Big Bird: Privacy Budget Management for W3C's Privacy-Preserving Attribution API

Pierre Tholoniat
Columbia University

Alison Caulfield*
Columbia University

Giorgio Cavicchioli
Columbia University

Mark Chen
Columbia University

Nikos Goutzoulias
Columbia University

Benjamin Case
Meta Platforms, Inc.

Asaf Cidon
Columbia University

Roxana Geambasu†
Columbia University

Mathias Lécuyer
University of British Columbia

Martin Thomson
Mozilla

Abstract

Privacy-preserving advertising APIs like Privacy-Preserving Attribution (PPA) are designed to enhance web privacy while enabling effective ad measurement. PPA offers an alternative to 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 *Big Bird*, a privacy budget manager for PPA that clarifies per-site budget semantics and introduces a global budgeting system grounded in resource isolation principles. Big Bird enforces utility-preserving limits via quota budgets and improves global budget utilization through a novel batched scheduling algorithm. Together, these mechanisms establish a robust foundation for enforcing privacy protections in adversarial environments. We implement Big Bird in 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, which we presented at SOSP last year [28], introduced the first formal framework based on individual differential privacy (individual DP) [8] to systematically analyze and optimize these APIs—a framework later adopted by Google in privacy analysis of its ARA API [11]. That same Cookie Monster framework now underpins Privacy-Preserving Attribution (PPA) [20], the API standard being drafted by Private Advertising Technology Working Group (PATWG), a W3C working group that includes representatives from all browsers [22]. We are active participants in PATWG, tackling technical challenges from a scientific perspective to help advance the APIs' practicality under strong privacy guarantees.

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

The absence of a principled approach to privacy budget management has led to unresolved questions within PATWG, creating uncertainty in key design decisions. For instance, should some sites get budget while others do not—and if so,

*Also affiliated with Microsoft, but work done in the context of Columbia University graduate program.

†Temporarily affiliated with Meta, but work done in the context of Columbia University research.

Contact authors: Pierre Tholoniat, Roxana Geambasu, Mathias Lécuyer ({pierre,roxana}@cs.columbia.edu, mathias.lecuyer@ubc.ca).

based on what criteria?[1] Should there be a cap on how many sites are allocated budget, and if so, how can we prevent a denial-of-service attack where one entity exhausts it?[2] Should API invocations be rate-limited to prevent privacy or DoS attacks? To date, there is no consensus, largely due to the lack of a principled foundation to drive the design.

We describe *Big Bird*, 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 add clarity to the semantics of PPA's ambiguous per-site budgeting—often muddied by shifting roles of third parties in the advertising ecosystem—we propose changes to the PPA interface, protocol, and terms of use. These changes allow per-site budgets to provably satisfy individual DP, but only under non-adaptive behavior across sites. This limitation adds to the rationale for a well-managed global budget in achieving end-to-end privacy in PPA.

For the global budget, the challenge is configuring and managing it to support benign workloads while resisting depletion by malicious actors. Our insight is to treat the global privacy budget as a *shared resource*—analogous to traditional computing resources but governed by privacy constraints—and to apply classic resource isolation techniques, such as quotas and fair scheduling [18, 12], in this new domain. Beyond per-site and global budgets, Big Bird introduces *quota budgets* that regulate global-budget consumption, ensuring graceful utility degradation for benign sites under attack. It does so by forcing adversaries to operate within expected workload bounds—which they can currently evade to wreak havoc on PPA's global budget. Further, to address underutilization of the global privacy budget caused by static quota partitioning, we propose a scheduling algorithm that reallocates unused capacity to otherwise-blocked requests. Together, these mechanisms establish a principled, practical foundation for PPA and give browsers a basis for enforceable defenses, along with guidance on where to focus.

We implement Big Bird in two components: (1) pdslib, a generic on-device individual DP library that subsumes Cookie Monster and extends it with Big Bird'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 Big Bird on a dataset from the Criteo ad-tech company, showing that: (1) well-chosen quotas preserve high utility for benign workloads, (2) quotas isolate benign sites under attack, and (3) batched scheduling boosts utilization without sacrificing isolation. We make our source code available via several repositories: pdslib at https://github.com/columbia/pdslib and Firefox integration at https://github.com/columbia/pdslib-firefox. We also plan to release our experimental infrastructure, which will be posted at https://github.com/columbia/big-bird in the near future.

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

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

2 PPA Overview and Gaps

2.1 PPA architecture

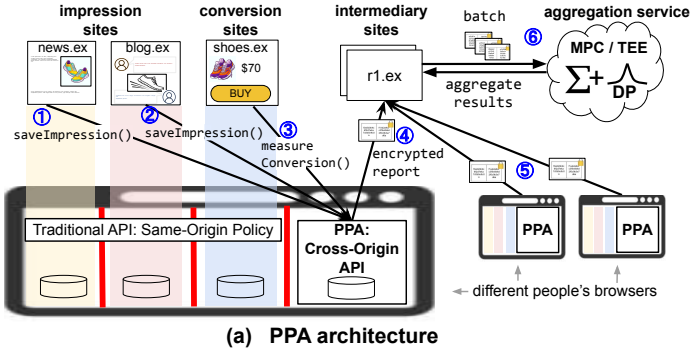
Fig. 1(a) illustrates the architecture of *Privacy-Preserving Attribution (PPA)*, W3C's browser-based API that enables *conversion attribution measurement* while preserving user privacy. Traditionally, browsers enforce a *same-origin policy*, while conversion attribution–the process of determining whether 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) 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`

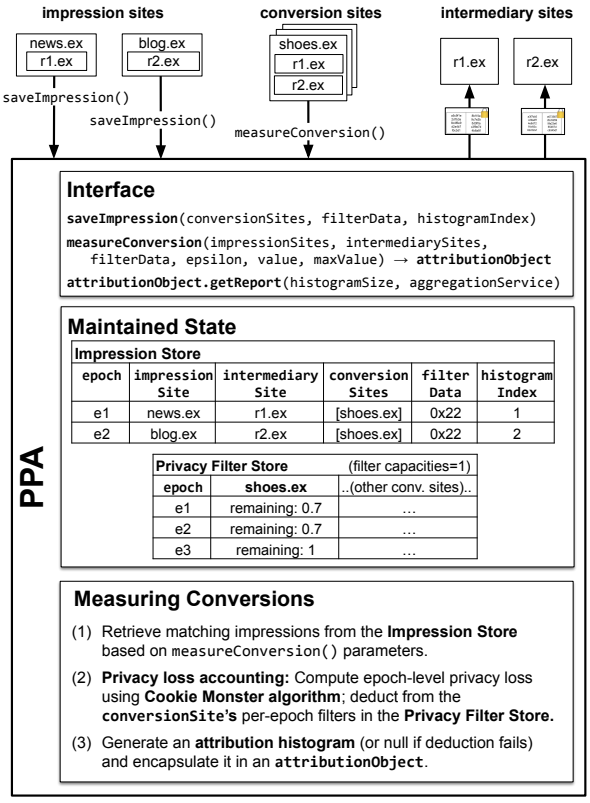
AttributionObject contains the full attribution histogram `{r1.ex:1=30, r2.ex:2=30}` and privacy loss tracking. Privacy loss is applied when `getReport(.)` is called.

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.

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

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

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

2.4 Stock-and-flow pattern

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

2.5 Global privacy filter

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

2.6 Foundational gaps

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

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. This paper proposes changes to the PPA API, protocol, and terms of use to add some level of clarity to per-site semantics and highlights the additional assumptions needed for them to remain provable (§4.1). The assumptions underscore the need for a well-configured, well-managed global budget to achieve end-to-end privacy without relying on them—an area where PPA currently lacks clear guidance, as we next discuss.

Gap 2: Lack of mechanisms to manage the global filter. The global filter—shared across all parties requesting reports from a browser—is a critical yet under-specified component of PPA. It introduces two challenges: (1) how to set its capacity to support benign workloads and (2) how to prevent malicious actors from depleting it—either to boost their own

utility or to deny service to others (e.g., competitors). While per-site budgets cap consumption per domain, they offer weak protection, as domain names are cheap and easily acquired. PATWG-discussed mitigations range from requiring sites to register with a trusted authority to browser-side heuristics for identifying illegitimate use of the API. But site registration faces resistance from some industry participants for undermining the API's open nature while heuristics rely on notions of "legitimacy" that are hard to define, especially for a nascent API with no deployment history and potentially valuable, unforeseen use cases. For instance, should the number of invocations be limited? Over what period and to what value? Should access to device-side budgets be restricted? On what grounds? While discussion in PATWG continues, we argue that the group lacks a foundation—a minimal set of principled mechanisms with well-defined properties under clear assumptions—to guide browsers toward targeted, defense-in-depth strategies that are both protective and not over-constraining for the API. This paper contributes such a foundation, from the vantage point of PPA's internal privacy budget management (§4.3).

3 Big Bird 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 Big Bird to both add clarity to 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 Big Bird have similar threat models. Users trust the 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. Big Bird 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. §4.3 gives example attacks to which PPA is currently vulnerable.

Big Bird 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, Big Bird 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, Big Bird 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, Big Bird still upholds its privacy guarantees, but its DoS resilience will diminish.

3.2 Running example

We update the *shoes.ex* example to support *cross-advertiser queries*, a feature PPA plans to add soon. Our Big Bird 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

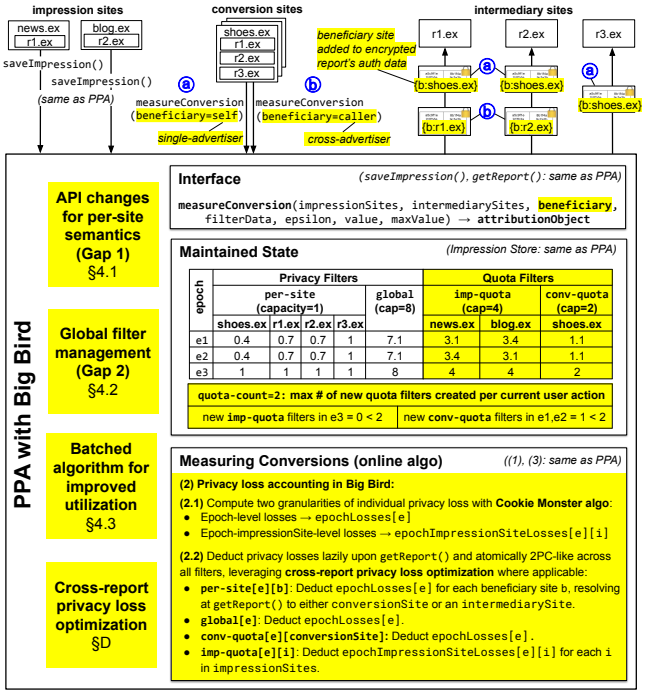


Fig. 2. Big Bird architecture. Changes vs. PPA (Fig. 1(c)) in yellow.

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 Big Bird architecture

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

API changes for per-site semantics (Gap 1, §4.1). We modify the API, protocol, and terms of use to eliminate ambiguity in budget attribution. Specifically, we introduce a *beneficiary site* parameter, authenticate it to the aggregation service, and enforce its use—both technically and contractually—so that reports can only support the intended site's DP queries. These changes prevent intermediaries from misusing reports funded by conversion sites, adding some level of clarity to per-site semantics for parties that comply with the protocol and its terms. This helps address PPA's Gap 1 from §2.6, but proving per-site DP properties still requires additional assumptions that the protocol and terms cannot enforce. This heightens the need for a well-configured, well-managed global filter whose formal guarantees do not depend on them.

Cross-report privacy loss optimization (§4.2). Big Bird introduces an optimization that reduces overcounting when multiple reports for the same conversion involve disjoint impression sets (e.g., across intermediaries). Since such reports reveal no more than the original attribution histogram to shared filters, Big Bird accounts for them jointly across privacy and quota filters.

Global filter management (Gap 2, §4.3). Without changing the API or protocol, we rework PPA's internal state to restore its intended stock-and-flow model of privacy loss, where user actions create "stock" at impression sites and trigger "flow" at conversion sites. Depletion attacks break this structure by automating flows or collapsing domain roles, draining the global filter without real user input. To defend against this, we introduce three quotas: one limits how much stock an impression site can create, another caps how much flow a conversion site can trigger, and a third bounds how many new sites can participate per user action. These quotas don't just limit indirect proxies (like API calls or intermediaries); they act directly on the core protected resource–the global filter–enforcing a privacy budget flow tightly coupled to actual user behavior and curbing adversarial misuse. We show that this leads to graceful degradation of utility for benign workloads under attack. This addresses PPA's Gap 2.

Batched algorithm to improve utilization (§4.4). Static quotas can underutilize the global budget, especially when some impression sites see little demand while others face heavy load. To address this inefficiency, Big Bird introduces a batched scheduling algorithm that collects unserved requests over a time interval and reallocates unused impression-site quota toward them at the end of each interval. This improves utilization without sacrificing resilience in our evaluation.

4 Detailed Design

We detail Big Bird's four core components, grounding each in our running example and the right side of Fig. 2.

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.

Big Bird 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. Pri-

vacy loss is then charged to the beneficiary's per-epoch filters, which are created as needed. Under the honest-but-curious model, we permit unrestricted filter creation. In the **protocol**, the beneficiary is included in the report's authenticated data, and aggregators are required to reject any batch with inconsistent beneficiaries. This blocks intermediaries from reusing reports charged to other clients' budgets. In the **terms of use**, we prohibit using DP results tied to one `beneficiarySite` to benefit another. Reports and results must remain siloed by beneficiary, even across shared infrastructure. This prohibits report-sharing among sharded identities (e.g., *shoes-cart.ex*, *shoes-purchase.ex*), cross-company collusion, and Sybil behavior (§3.1). Honest-but-curious sites will avoid these.

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).

Per-site guarantees and their limitations. The preceding PPA changes bring much-needed clarity to per-site privacy loss accounting, and we plan to propose them to PPA.

However, our formal analysis (Appendix F) shows that proving per-site DP guarantees still require an additional assumption: the exclusion of data-driven adaptivity across sites. This assumption, to our knowledge, is novel in the DP literature, yet we suspect that it applies broadly to systems enforcing sharded (non-global) semantics—including PPA, Big Bird, Cookie Monster, and ARA. We believe that this gap has gone unrecognized in prior work due to incomplete system modeling, particularly the tendency to model the system's behavior for each site in isolation, as done in Cookie Monster [28] and ARA [11]. In contrast, Appendix F explicitly models cross-site behavior for Big Bird, revealing the necessity of this assumption for formally establishing granular DP.

We leave to future work a deeper exploration of how broadly this assumption applies and what it means in practice. Still, we emphasize that its presence does not diminish the importance of per-site budgeting in PPA. These budgets help constrain privacy loss against individual domains and can be configured far more tightly than a global budget realistically can. We thus urge PATWG to continue rigorously enforcing per-site budgets—especially with our proposed changes to clarify them—while also advancing toward effective enforcement of a well-configured, well-managed global budget, a topic this paper also addresses.

4.2 Cross-report privacy loss optimization

We illustrate Big Bird's *cross-report* privacy loss optimization using our running example, deferring a general treatment to Appendix D. 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 parallel-composition-like 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.

Appendix D formalizes the 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.

4.3 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, Big Bird 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.

Big Bird quota system. Fig. 2 (yellow background) highlights the internal state maintained by Big Bird to manage global privacy filters in PPA. Appendix B formalizes the system's behavior and proves its privacy and resilience properties. We use two types of quotas: (1) *quota filters*, `imp-quota` and `conv-quota`, are implemented as DP filters—not for privacy accounting, but to regulate global filter consumption, a novel use in DP literature; (2) a standard *count-based quota* limits 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`, Big Bird creates `imp-quota[e][i]`, with capacity set to a share of the global filter. This quota is consumed only if a

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

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

Tab. 1. Big Bird filter configurations.

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

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

Fig. 2 sketches Big Bird'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()`, Big Bird attempts to deduct losses across relevant filters in an atomic transaction per epoch: success only alters state if all checks pass. A non-null report is returned only if checks succeed in all epochs. Relevant filters include the beneficiary's `per-site` filter, the `global` filter, the conversion site's `conv-quota`, and an `imp-quota` for each impression site with non-zero loss. To efficiently enforce impression-site quotas, we compute loss at the (epoch, impression site) level and charge it to the corresponding `imp-quota`. Cross-report optimizations eliminate redundant charges. Although total quota capacities may exceed the global budget at any moment, Big Bird's atomic checks ensure global filter is never breached.

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

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

Resilience to DoS depletion. We prove the following:

Theorem 1 (Resilience to DoS depletion (proof in B.4))**.** *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 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. □

4.4 Batched scheduling to improve utilization

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

Dynamically adjusting quotas based on observed demand would invite attacks, but we observe that if PPA supports *batched mode*—collecting requests over a period of time and

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

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

Algorithm 1 Batched algorithm

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

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

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

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

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

4.5 Recommendations for PATWG

Big Bird 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 Big Bird 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 Big Bird, 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 Big Bird and by first parties managing their own quotas—intermediaries do not impact privacy or resilience guarantees. While Big Bird 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, Big Bird'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 Big Bird.

5 Prototype

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

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

6 Evaluation

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

6.1 Methodology

Dataset. We evaluate Big Bird on CriteoPrivateAd [27], a dataset released by the Criteo ad-tech company, a PATWG participant, for the purpose of benchmarking private advertising systems. The dataset samples 30 days of production traffic using third-party cookies, with 104M impressions across 220k publisher sites (`publisher_id`) and 10k conversion sites (`campaign_id`, a good proxy for advertiser domains [27]). The data involves a *single intermediary*–Criteo itself.

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

The dataset is impression-subsampled, not device-subsampled, so most devices have only one impression. Criteo provides the true device-level impression distribution and a resampling method to match it [27]. Using this, we construct a dataset with 4.6M impressions and 5.6M conversions across 1.4M devices, in which the median (resp. 90th percentile) device has 2 (resp. 6) impressions and 4 (resp. 16) conversions. We tune our algorithms and workload parameters on the first 10 days and report results on the remaining 20 days that we explicitly hold out for this evaluation.

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

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

Attack workload process. To evaluate Big Bird's resilience to DoS depletion attacks (specifically, Attacks 2 and 3 from §4.3), we inject a synthetic adversarial workload into real benign traffic. (Attack 1 is excluded, as the dataset only includes a single intermediary.) Our setup simulates an attacker who controls 10 popular impression sites and 10 popular conversion sites, that each redirects to 7 new Sybil domains per real user action. This corresponds to a highly permissive configuration of quota-count = 8 to ensure a strong attack since most devices in Criteo convert only once. This attack evaluation methodology is still preliminary and we will explore other shapes of attacker traffic. We instantiate the attack as follows.

We first create 10 attacker impression sites, created by

duplicating the most active impression sites to ensure the attacker interacts with many devices, since most devices in Criteo see only one impression site. For each impression, the attacker registers all its domains as target conversion sites. Then, we identify the top 10 real sites by number of conversions, and duplicate them to create 10 attacker conversion sites. For each user action on an attacker conversion site, we request a report with maximum $\epsilon = \epsilon_{\text{per-site}}$, where all the attacker impression sites are marked as relevant. We then redirect 7 times to new attacker conversion sites that request reports in the same way.

Since each series of attacker events (one impression or conversion event followed by 7 redirections) reuses device ids and timestamps from a real event, we need to break ties to decide in which order to inject attacker events on top of real events. We flip an unbiased coin to either let the attacker run first or the real event run first. This gives a fair chance for real sites to run their queries without being systematically front-runned by the attacker.

Baselines. We compare Big Bird against two baselines. The first is **PPA w/o global filter**, which enforces only per-site filters. This baseline is how Cookie Monster [28] itself would behave. The second, **PPA w/ global filter**, extends PPA w/o global budget by adding a global filter, aligning with the current PPA draft specification.

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

6.2 "Normal" workload parameters in Criteo (Q1)

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

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

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

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

Tab. 2. Criteo "normal" workload.

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

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

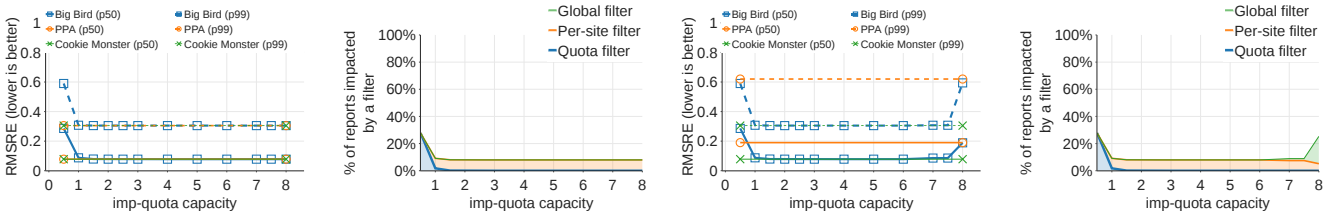
6.3 Query errors under normal workload (Q2)

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

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

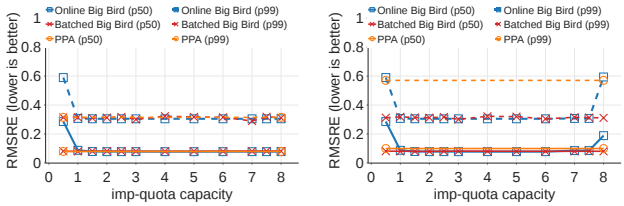
6.4 Query errors under DoS attack (Q3)

We evaluate Big Bird's resilience under X's attack against the global filter. Fig. 3c plots median and tail error for benign queries as a function of imp-quota capacity. PPA w/o global filter is unaffected (but also lacks a global privacy guarantee).



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

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



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

Fig. 4. Batched algorithm evaluation.

PPA w/ global filter lacks protection from X, so its tail error rises sharply under attack. Big Bird with a well-configured quota (e.g., $\epsilon_{\text{imp-quota}} = 4$ per p95 in Tab. 2) fully isolates honest queriers, matching the error levels of PPA w/o global filter.

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

6.5 Batched algorithm evaluation (Q4)

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

7 Related Work

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

Among prior work on PPA and related APIs, the most relevant is Cookie Monster [28], which tracks privacy loss using per-epoch filters tied to individual queriers. Big Bird goes further by managing these filters alongside the global filter and introducing a cross-report optimization that Cookie Monster lacks. Other foundational work includes Google's ARA papers [2, 6, 11], research on the MPC components of these APIs [4, 5, 3], now-retired proposals like IPA [14] and PAM [21], or Hybrid [13], which proposed several planned extensions to PPA. More broadly, there is related work on privacy-preserving ad targeting [31, 32].

Beyond ad measurement, this paper contributes to the broader challenge of **privacy budget management**, a crucial but understudied area in DP. While budget allocation *within* a single query is well-studied [17, 1], budget management *across* queries from mutually distrustful parties is less explored. Notably, [23, 24] balance utility across analysts sharing a single global budget, but without per-analyst guarantees. Most similar to us, [30] simultaneously enforces per-analyst guarantees and global DP guarantees in case of collusion. However, they do not consider adaptively chosen data and budgets, which sidesteps the fundamental challenges we identify in Big Bird. Relevant systems include those for global budget scheduling [18, 29, 15], which inspire our batched scheduling approach but differ in key ways. [18] proposes a max-min-fair algorithm for allocating global budget across epochs, similar in spirit to our scheduler. Big Bird departs from these systems in two ways. First, it operates under epoch-level *individual DP*, which rules out relying on cross-epoch budget information. Second, it supports adaptive, multi-task queriers and defends against DoS depletion—gaps unaddressed in prior work.

Big Bird builds on the literature on privacy filters [25],

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

8 Conclusions

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

10.1145/1807085.1807104. URL: https://doi.org/10.1145/1807085.1807104.

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

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

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

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

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

[23] David Pujol et al. "Budget sharing for multi-analyst differential privacy". In: *Proc. VLDB Endow.* 14.10 (June 2021), pp. 1805–1817. ISSN: 2150-8097. DOI: 10.14778/3467861.3467870. URL: https://doi.org/10.14778/3467861.3467870.

[24] David Pujol et al. "Multi-Analyst Differential Privacy for Online Query Answering". In: *Proc. VLDB Endow.* 16.4 (Dec. 1, 2022), pp. 816–828. ISSN: 2150-8097. DOI: 10.14778/3574245.3574265.

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

[26] Ryan M Rogers et al. "Privacy Odometers and Filters: Pay-as-you-go Composition". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016.

[27] 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.

[28] 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.

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

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

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

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

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

This section formalizes API changes to clarify the per-site semantics. Starting from Cookie Monster's formalism, we adapt it to capture Big Bird'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 Big Bird's data and query model (§A.1 and §A.2). This allows us to set up the formal framework for our DP analysis in Alg. 2, which defines a single mechanism that answers to different beneficiaries simultaneously. We conclude with the sensitivity analyses needed in subsequent sections (§A.3).

A.1 Data model

We align with the terminology of PPA and also 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)

Definition 1 (Database with per-site semantics)**.** *A database D is a set of device-epoch records where each record $x = (d, e, F) \in \mathcal{X} = \mathcal{D} \times \mathcal{E} \times \mathcal{P}(\mathcal{S} \times \mathcal{I} \cup \mathcal{S} \times \mathcal{P}(\mathcal{S}) \times \mathcal{C})$ 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: $f = (i, imp) \in \mathcal{S} \times \mathcal{I}$ or $f = (c, \mathbf{b}, conv) \in \mathcal{S} \times \mathcal{P}(\mathcal{S}) \times \mathcal{C}$. Additionally, conversions contain the set of beneficiary sites $\mathbf{b}$ that will receive the conversion report ($\mathbf{b} = \{b\}$ without the cross-report privacy loss optimization from 4.2).*

A.2 Query model

We now define queries and reports, with a slight adaptation of Cookie Monster's definitions to our per-site semantics. While Cookie Monster comes with an arbitrary set of public events P for each beneficiary b, here for simplicity we assume that all the conversions for a beneficiary are public. Using in Cookie Monster's terminology, that means we set $P = C_b := \{(c, b, \text{conv}), c \in C, \text{conv} \in C\}$, where C is the set of all conversions. Also, while Cookie Monster defines a set of *relevant events*, potentially including conversions, in Def. 2 we only consider *relevant impressions* for simplicity. In particular, this hardcodes "Case 1" from [28, Thm. 1].

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

$$A(F_1, \ldots, F_k) = A(F_1 \cap F_A, \ldots, F_k \cap F_A). \tag{1}$$

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

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

$$\rho_r : D \in \mathbb{D} \mapsto A(D^E_{d_r}). \tag{2}$$

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

A.3 Sensitivity analyses

The Cookie Monster paper analyzes global and individual sensitivities of queries at device-epoch level. In Big Bird, we additionally need such analyses at the device-epoch-site level, as our impression-site quota operates at this granularity. This section provides the necessary sensitivity definitions and analyses.

Definition 5 (Per-Epoch Sensitivity, same as [28])**.** *Fix a report $\rho : \mathbb{D} \to \mathbb{R}^m$ for some m. We define the* per-epoch individual L_1 sensitivity of ρ for a device-epoch $x \in \mathcal{X}$ *as follows:*

$$\Delta(\rho) := \max_{D, D' \in \mathbb{D} : \exists x \in \mathcal{X}, D' \sim_x D} \|\rho(D) - \rho(D')\|_1, \tag{3}$$

where $D' \sim_x D$ means D' and D differ by the single record x.

We also define the per-epoch global L_1 sensitivity of ρ *as follows:*

$$\Delta(\rho) := \max_{x \in \mathcal{X}} \Delta_x(\rho) \tag{4}$$

Definition 6 (Per-Epoch-Site Sensitivity)**.** *Fix a report $\rho : \mathbb{D} \to \mathbb{R}^m$ for some m, and an impression site $i \in \mathcal{S}$. We define the* per-epoch-site individual L_1 sensitivity of ρ for a device-epoch-site $x \in \mathcal{X}, i \in \mathcal{S}$ *as follows:*

$$\Delta_{x,i}(\rho) := \max_{D, D' \in \mathbb{D} : D' \sim_{x,i} D} \|\rho(D) - \rho(D')\|_1 \tag{5}$$

where $D' \sim_{x,i} D$ means D' and D differ on a single record x's impressions on site i. That is, there exists $D_0 \in \mathbb{D}$, a record $x \in \mathcal{X}$ such that $\{D, D'\} = \{D_0 + x, D_0 + x^{i \to \emptyset}\}$, where $x^{i \to \emptyset} := (d, e, F^{i \to \emptyset})$ is the record obtained by removing all the impressions on i from x.

We also define the per-epoch-site global L_1 sensitivity of ρ

as follows:

$$\Delta_i(\rho) := \max_{x \in \mathcal{X}} \Delta_{i,x}(\rho) \tag{6}$$

To simplify subsequent results, we define some notation:

Definition 7 (Zeroing-out)**.** *Fix a vector of impression sets $\mathbf{F} = (F_1, \ldots, F_k) \in \mathcal{P}(\mathcal{I})^k$ for any $k > 0$. For $i \in \mathcal{S}$ and $j \in [k]$ we define:*

- $\mathbf{F}^{j \to \emptyset} := (F_1, \ldots, F_{j-1}, \emptyset, F_{j+1}, \ldots, F_k)$, i.e., *we zero-out the jth epoch.*
- $\mathbf{F}^{j,i \to \emptyset} := (F_1, \ldots, F_{j-1}, F_j \setminus \mathcal{I}_i, F_{j+1}, \ldots, F_k)$, i.e., *we zero-out all the impressions $\mathcal{I}_i := \{(i, imp), imp \in \mathcal{I}\}$ belonging to site i from the jth epoch.*

Plugging Def. 2 into Def. 5 and Def. 6 immediately gives:

Lemma 1 (Global sensitivity of reports)**.** *Fix a device d, a set of k epochs E, an attribution function A and the corresponding report $\rho : D \mapsto A(D_d^E)$. We have:*

$$\Delta(\rho) = \max_{\mathbf{F} \in \mathcal{P}(\mathcal{I})^k, j \in [k]} \|A(\mathbf{F}) - A(\mathbf{F}^{j \to \emptyset})\|_1. \tag{7}$$

Moreover, for any impression site $i \in \mathcal{S}$ we have:

$$\Delta_i(\rho) = \max_{\mathbf{F} \in \mathcal{P}(\mathcal{I})^k, j \in [k], i \in \mathcal{S}} \|A(\mathbf{F}) - A(\mathbf{F}^{j,i \to \emptyset})\|_1. \tag{8}$$

Theorem 3 (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 $I_r = \{i_1, \ldots, i_{m_e}\}$ for each epoch $e \in E_r$, an attribution function A with relevant events F_A, and the corresponding report $\rho : D \mapsto A(D_{d_r}^{E_r, \{I_r\}_{e \in E_r}})$. Fix a device-epoch record $x = (d, e, F_j) \in \mathcal{X}$, where $F \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{C} \cup \mathcal{S} \times \mathcal{I}$, so that $x_i = (d, e, F_{j,i})$ is the projection where F_i contains only events related to site i.*

We can upper bound the individual sensitivity of reports per epoch-site $\Delta_{x,i}(\rho)$ by:

$$\Delta_{x,i}(\rho) \leq \begin{cases} 0 & \text{if } d \neq d_r, e \notin E_r, i \notin I_r, \\ & \text{or } F_{j,i} \cap F_A = \emptyset \\ \|A(F_i) - A(\emptyset)\|_1 & \text{if } d = d_r, E_r = \{e\}, \text{ and } I_r = \{i\} \\ \Delta_i(\rho) & \text{if } d = d_r, e \in E_r, i \in I_r, F_{j,i} \cap F_A \neq \emptyset, \\ & \text{and } (|E_r| \geq 2 \text{ or } |I_r| \geq 2) \end{cases} \tag{9}$$

Proof. Fix a report ρ, an impression site i and $x = (d, e, F) \in \mathcal{X}$ with impressions F_i on site i. Consider any $D, D' \in \mathbb{D}$ such that $D' = D + x_i$. We have $\rho(D) = A(D_{d_r}^{E_r, \{I_r\}_{e \in E_r}})$ and $\rho(D') = A((D')_{d_r}^{E_r, \{I_r\}_{e \in E_r}})$.

- First, if $d \neq d_r$, $e \notin E_r$, or $i \notin I_r$, then $(D')_{d_r}^{E_r, \{I_r\}_{e \in E_r}} = D_{d_r}^{E_r, \{I_r\}_{e \in E_r}}$. 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:

    $$\|\rho(D) - \rho(D')\|_1 = \|A(\mathbf{F}^{j,i \to \emptyset}) - A(\mathbf{F})\|_1, \tag{10}$$

    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.

□

Next, we show that $\Delta_i(\rho) \leq 2\Delta(\rho)$ for any report. For reports using certain attribution functions, we can have a tighter $\Delta_i(\rho) = \Delta(\rho)$ bound, but it does not hold in general.

Lemma 2 (Relationship between Device-Epoch-Site and Device-Epoch Sensitivities)**.** *For any report ρ with attribution function A, let F denote the full dataset, F_j denote all data in F pertaining to epoch j, and $F_{j,i}$ denote all data in F_j pertaining to site i. The following inequality holds:*

$$\Delta_i(\rho) \leq 2\Delta(\rho) \tag{11}$$

Proof. Recall the definition of device-epoch-site global sensitivity:

$$\Delta_i(\rho) = \max_{\mathbf{F} \in \mathcal{P}(\mathcal{I})^k, j \in [k], i \in \mathcal{S}} \|A(\mathbf{F}^{j,i \to \emptyset}) - A(\mathbf{F})\|_1. \tag{12}$$

We can decompose this expression using the triangle in-

equality:

$$\|A(\mathbf{F}^{j,i\to\emptyset}) - A(\mathbf{F})\|_1 \tag{13}$$

$$=\|A(\mathbf{F}^{j,i\to\emptyset}) - A(\mathbf{F}^{j\to\emptyset}) + A(\mathbf{F}^{j\to\emptyset}) - A(\mathbf{F})\|_1 \tag{14}$$

$$\leq\|A(\mathbf{F}^{j,i\to\emptyset}) - A(\mathbf{F}^{j\to\emptyset})\|_1 + \|A(\mathbf{F}^{j\to\emptyset}) - A(\mathbf{F})\|_1 \tag{15}$$

For the first term, note that $A(\mathbf{F}^{j,i\to\emptyset})$ uses events $F_{j,\neg i}$ from epoch j where $F_{j,\neg i} = F_j \setminus F_{j,i}$, while $A(\mathbf{F}^{j\to\emptyset})$ uses no events from epoch j. Since $F_{j,\neg i} \subseteq F_j$ and all events from epoch j for a given device come from a single device-epoch record, we can view this as the change from adding a single record containing events $F_{j,\neg i}$. This is bounded by the definition of $\Delta(\rho)$:

$$\|A(\mathbf{F}^{j,i\to\emptyset}) - A(\mathbf{F}^{j\to\emptyset})\|_1 \leq \Delta(\rho). \tag{16}$$

The second term represents the sensitivity to removing events from the entire epoch j. In the individual DP setting with device-epoch records, all events from epoch j for a given device come from a single device-epoch record. Removing this entire record corresponds exactly to one of the cases considered in the definition of $\Delta(\rho)$. Therefore:

$$\|A(\mathbf{F}^{j\to\emptyset}) - A(\mathbf{F})\|_1 \leq \Delta(\rho). \tag{17}$$

Therefore, substituting the two upper bounds into Eq. (15):

$$\Delta_i(\rho) \leq \Delta(\rho) + \Delta(\rho) = 2\Delta(\rho). \tag{18}$$

□

Alg. 2 presents an abstract model of Big Bird's operation, capturing how it answers beneficiary queries sequentially. In Alg. 2, t indexes a *batch query step*. In each such step, the `AnswerQuery` function is called with a batch of report requests Q^t. `AnswerQuery` then processes each individual report p_r (for $r \in R$, where R is the set of report identifiers in Q^t), which includes `GenerateReport` creating an individual clipped attribution report, followed by aggregation for the batch, and finally receiving a noisy result for that batch. This abstract model provides well-defined mechanisms for which we can prove privacy properties in §B.2

Algorithm 2 Formalism for DP analysis

Input
  Database D
  Stream of adaptively chosen queries, up to $t_{\max}$ steps
function $\mathcal{M}(D)$
  $(S_b)_{b\in\mathcal{S}} = (\emptyset)_{b\in\mathcal{S}}$
  for $(d, e, F) \in D$ **do**
    for $f \in F : f = (c, \mathbf{b}, \text{conv})$ **do**
      Generate report identifier $r \xleftarrow{\$} U(\mathbb{Z})$
      // Save mapping from r to the device that generated it
      $d_r \leftarrow d$
      for $b \in \mathbf{b}$ **do**
        $S_b \leftarrow S_b \cup \{(r, f)\}$
  // Each beneficiary receives its public events and corresponding report identifiers
  for $b \in \mathcal{S}$ **do**
    output S_b to b
  // Beneficiaries ask queries interactively. If b has nothing to ask, it can send an empty query with zero sensitivity.
  for $t \in [t_{\max}]$ **do**
    for $b \in \mathcal{S}$ **do**
      receive Q_t^b, λ_t from beneficiary site b.
      output AnswerQuery(D, Q_t^b, λ_t, b) to b
// *Collect, aggregate and noise reports to answer Q*
function AnswerQuery(D, Q, λ)
  $(\rho_r)_{r\in R} \leftarrow Q$ // *Get report identifiers from Q*
  for $r \in R$ **do**
    $\hat{\rho}_r \leftarrow$ GenerateReport(D, ρ_r, λ)
  Sample $X \sim \mathcal{L}(\lambda)$
  return $\sum_{r\in R} \hat{\rho}_r + X$

B Global Filter Management (Gap 2)

B.1 Algorithm

Overview. Big Bird manages per-site and global privacy filters using the quota mechanisms described in §4.3. Alg. 3 depicts the functionality triggered on receiving a report request (i.e., `measureConversion()` and `getReport()`). Big Bird checks and consumes budget from the relevant filters and prunes the resulting report based on filter status. First, Big Bird ensures all filters and quotas are initialized. Second, it computes the privacy losses incurred—both at the epoch level ([28, §C]) and at the site level (Def. 8). Third, it checks whether all filters have sufficient budget and attempts to consume it. To ensure avoid wasting budget from some filters (or quotas) when other filters are out of budget, Big Bird uses a two-phase commit protocol to deduct privacy losses from multiple filters atomically. Alg. 4 formalizes this: if any filter cannot afford its share of the privacy loss, the report is zeroed out, and no budget is consumed from any filter for that report.

Subroutines. Def. 9, 8, 10 and Alg. 4 define subroutines used in Alg. 3. The first two definitions rely on the sensitivity bounds from Thm. 3 (for the impression-site quota) and [28] (for the filters and the other quotas).

Definition 8 (EpochImpSiteBudget)**.** *Let $x = (d, e, F) \in \mathcal{X}$ be a device-epoch record, $i \in S$ an impression site, ρ an attribution report, and $\lambda > 0$ the Laplace noise scale applied to ρ. Given the upper bound $\tilde{\Delta}_{x_i}(\rho) \geq \Delta_{x_i}(\rho)$ on the per-epoch-site individual sensitivity given by Thm. 3, we can upper bound the epoch–site privacy loss consumed by ρ at (x, i) by*

$$\mathit{EpochImpSiteBudget}(x, i, \rho, \lambda) := \frac{\tilde{\Delta}_{x_i}(\rho)}{\lambda}. \tag{19}$$

Definition 9 (EpochBudget, from [28])**.** *Fix a device-epoch record $x \in \mathcal{X}$, ρ an attribution report, and $\lambda > 0$ the Laplace noise scale applied to ρ. Given the upper bound $\tilde{\Delta}_x(\rho) \geq$*

$\Delta_x(\rho)$ *on the per-epoch individual sensitivity given by [28], the individual privacy loss for device-epoch record* x *is:*

$$\text{EpochBudget}(x, \rho, \lambda) := \frac{\tilde{\Delta}_x(\rho)}{\lambda} \tag{20}$$

Definition 10 (Filters $\mathcal{F}_x$)**.** *For each device-epoch record* $x = (d, e, F)$*, we maintain several pure DP filters [26]:*

- $\mathcal{F}_x^{\text{per-site filter}[b]}$ *for each beneficiary site* b*, with capacity* $\epsilon_{\text{per-site}}$*,*
- $\mathcal{F}_x^{\text{global filter}}$ *with capacity* ϵ_{global}*,*
- $\mathcal{F}_x^{\text{conversion-site quota}[c]}$ *for each conversion site* c*, with capacity* $\epsilon_{\text{conv-quota}}$*,*
- $\mathcal{F}_x^{\text{impression-site quota}[i]}$ *for each impression site* i*, with capacity* $\epsilon_{\text{imp-quota}}$*.*

For each filter $\mathcal{F}$*, we adapt the notation from [26] to recursively define* $\mathcal{F}$.canConsume*,* $\mathcal{F}$.tryConsume *and* $pass_{\mathcal{F}}[t]$*, for a sequence of adaptively chosen privacy budgets* $\epsilon_x^1, \ldots, \epsilon_x^t$*, as follows:*

- canConsume(ϵ_x^t)*: Returns* TRUE *if the filter can accommodate additional privacy loss* ϵ_x^t*,* i.e.*,* $\epsilon_x^t \leq \epsilon_{initial} - \sum_{k\in[t-1]} \epsilon_x^k \cdot pass_{\mathcal{F}}[k]$ *where* $\epsilon_{initial}$ *is the filter capacity.*
- tryConsume(ϵ_x^t)*: Calls* canConsume(ϵ_x^t)*. If successful, sets* $pass_{\mathcal{F}}[t] = 1$ *to deduct* ϵ *from the filter's remaining capacity; otherwise, sets* $pass_{\mathcal{F}}[t] = 0$*.*

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 [28, Alg. 2] generalizes to adaptively generated data. Another simplification compared to [28] is that public events are never relevant events for our attribution functions (*i.e.,* , the output of a conversion report can only depend on impressions, not on other conversions). This is enforcing the constraint on queries mentioned in Case 1 of [28, Thm. 1].

Theorem 4 (Global DP Guarantee)**.** *Consider* $x \in \mathcal{X}$ *with global filter capacity* ϵ_{global}*. Then,* $\mathcal{M}$ *satisfies individual device-epoch* ϵ_{global}*-DP for* x *under public information* C*.*

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

$$\left|\ln\left(\frac{\Pr[\mathcal{M}(D + x_C) = v]}{\Pr[\mathcal{M}(D + x) = v]}\right)\right| \leq \epsilon_{\text{global}}. \tag{21}$$

Consider any database D'. v is the vector of values returned at different points in Alg. 2. We split it into $v = (v_{\text{pub}}, v_1, \ldots, v_{t_{\max}})$

Algorithm 3 Big Bird algorithm (on-device)

Input
Filter and quota capacities ϵ_{global}, $\epsilon_{\text{per-site}}$, $\epsilon_{\text{imp-quota}}$, $\epsilon_{\text{conv-quota}}$
InitializeFilters$(\epsilon_{\text{global}}, \epsilon_{\text{per-site}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$ sets up per-device filter and quota capacities
AtomicFilterCheckAndConsume$(\mathcal{F}_x, b, c, \mathbf{i}, \epsilon_x^t, \epsilon_{\mathbf{x}}^{\mathbf{i},t})$ atomic quota check and updates the filters (Alg. 4)
// Generate report and update on-device budget
function GenerateReport(D, ρ, λ)
Read ρ to get device d, conversion site c, beneficiary site b, impression sites $\mathbf{i}$, target epochs E, attribution function A.
for $e \in E$ **do**
$x \leftarrow (d, e, D_d^e)$
if $\mathcal{F}_x$ is not defined **then**
$\mathcal{F}_x \leftarrow$ InitializeFilters$(\epsilon_{\text{global}}, \epsilon_{\text{per-site}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$
$F_e \leftarrow D_d^e$
$\epsilon_x^t \leftarrow$ EpochBudget(x, ρ, λ)
$\epsilon_{\mathbf{x}}^{\mathbf{i},t} \leftarrow \{\}$
for $i \in \mathbf{i}$ **do**
$\epsilon_x^i \leftarrow$ EpochImpSiteBudget(x, i, ρ, λ)
$\epsilon_{\mathbf{x}}^{\mathbf{i},t}[i] \leftarrow \epsilon_x^i$
if AtomicFilterCheckAndConsume$(\mathcal{F}_x, b, c, \mathbf{i}, \epsilon_x^t, \epsilon_{\mathbf{x}}^{\mathbf{i},t})$ = FALSE **then**
$F_e \leftarrow \emptyset$ *// Empty the epoch if any filter check fails*
$\rho \leftarrow A((F_e)_{e\in E})$ *// Clipped attribution report*
return ρ

where v_{pub} is a value for the output from Line 15 of Alg. 2 (initial public events), and v_t is the output for the query Q^t at step t (Line 20). We denote by $\mathcal{M}_{\text{pub}}(D')$ the random variable of the output at Line 15, and $\mathcal{M}_t(D')$ the random variable of the output at Line 20. By conditioning over past outputs $(v_{\text{pub}}, v_1, \ldots, v_{t-1})$ at each time step $t \in [t_{\max}]$ we get:

$$\Pr[\mathcal{M}(D') = v] \tag{22}$$

$$= \Pr[\mathcal{M}_{\text{pub}}(D') = v_{\text{pub}}] \cdot \prod_{t=1}^{t_{\max}} \Pr[\mathcal{M}_t(D') = v_t | v_{<t}]. \tag{23}$$

Take $t \in [t_{\max}]$. By Algorithm 2, M_t corresponds to the processing of a query Q^t. We have:

$$\Pr[M_t(D') = v_t | v_{<t}] = \Pr[\text{AnswerQuery}(Q^t; D', \mathcal{F}_t) = v_t], \tag{24}$$

where the query Q^t and the state of the privacy filters $\mathcal{F}_t$ 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 report
$\epsilon_{\mathbf{x}}^{\mathbf{i},t}$: epoch-site-level privacy loss for a particular report
canConsume: function as is defined in Def. 10
tryConsume: function as is defined in Def. 10

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

$$\Pr[\mathcal{M}_t(D') = v_t | v_{<t}] = \Pr\left[\sum_{r \in R_t} \rho_r(D'; \mathcal{F}_{t,r}) + X_t = v_t\right], \tag{25}$$

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

Now, we instantiate D' to be either $D' = D + x_c$ or $D' = D + x$. In particular, when $D' = D + x_C$ we have $\rho_r(D + x_C) = \rho_r(D + x_C \cap F_A) = \rho_r(D)$ by Def. 2 since $F_A \subset \mathcal{I}$ and $x_C \cap \mathcal{I} = \emptyset$. We also abuse notation by letting $\rho_r(\cdot) := \rho_r(\cdot, \mathcal{F}_{t,r})$, since the state of the filter $\mathcal{F}_{t,r}$ is fully determined by $v_{<t}$. We get:

$$\left|\ln\left(\frac{\Pr[\mathcal{M}(D + x_C) = v]}{\Pr[\mathcal{M}(D + x) = v]}\right)\right| \tag{26}$$

$$= \left|\ln\left(\frac{\prod_{t=1}^{t_{\max}} \Pr[\sum_{r \in R_t} \rho_r(D) + X_t = v_t]}{\prod_{t=1}^{t_{\max}} \Pr[\sum_{r \in R_t} \rho_r(D + x) + X_t = v_t]}\right)\right| \tag{27}$$

$$\leq \sum_{t=1}^{t_{\max}} \left|\ln\left(\frac{\Pr[\sum_{r \in R_t} \rho_r(D) + X_t = v_t]}{\Pr[\sum_{r \in R_t} \rho_r(D + x) + X_t = v_t]}\right)\right| \tag{28}$$

where the first equality comes from Eq. 22 the fact that the outputs v_{pub} at Line 15 are identical across both worlds by definition of x_C.

Now, take $t \in t_{\max}$ and $r \in R_t$. We will show that:

$$\|\rho_r(D) - \rho_r(D + x)\| \leq \Delta_x(\rho_r)\text{pass}_r \tag{29}$$

Recall that pass_r denotes whether r passed the atomic filter check in Alg. 4. There are two cases:

- If $\text{pass}_r = 0$, we have $\rho_r(D + x) = \rho_r(D)$ because of Alg. 3, Line 19. Note that $\text{pass}_r = 0$ can happen even if $\mathcal{F}^{\text{global filter}}$ has enough budget, for instance if the per-site filter or a quota is out of budget. Hence Eq. 29 holds in this case.
- If $\text{pass}_r = 1$, we have $\|\rho_r(D + x) - \rho_r(D)\|_1 \leq \Delta_x(\rho_r)$, so Eq. 29 holds in this case too.

Thus by triangle inequality followed by Def. 9 we have:

$$\|\sum_{r \in R_t} \rho_r(D) - \sum_{r \in R_t} \rho_r(D + x)\| \leq \sum_{r \in R_t} \Delta_x(\rho_r)\text{pass}_r \tag{30}$$

$$\leq \sum_{r \in R_t} \lambda_t \cdot \epsilon_r \text{pass}_r \tag{31}$$

And since $X_t \sim \text{Lap}(\lambda_t)$, by property of the Laplace distribution we get:

$$\left|\ln\left(\frac{\Pr[\sum_{r \in R_t} \rho_r(D) + X_t = v_t]}{\Pr[\sum_{r \in R_t} \rho_r(D + x) + X_t = v_t]}\right)\right| \leq \sum_{r \in R_t} \epsilon_r \text{pass}_r \tag{32}$$

Injecting Eq. 32 into Eq. 28 gives:

$$\left|\ln\left(\frac{\Pr[\mathcal{M}(D + x_C) = v]}{\Pr[\mathcal{M}(D + x) = v]}\right)\right| \leq \sum_{t=1}^{t_{\max}} \sum_{r \in R_t} \epsilon_r \text{pass}_r \tag{33}$$

Finally, since $\text{pass}_r = 1$ implies that r passes $\mathcal{F}^{\text{global filter}}$, by definition of $\mathcal{F}^{\text{global filter}}$, the accumulated loss over all reports related to query x is below the filter capacity:

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

Hence we have shown Eq. 21. □

B.3 Adaptively generated data

While Alg. 2 takes a fixed database D as input, the privacy guarantees from Thm. 4 hold even when data is generated adaptively. We sketch the argument here and refer to [28] for a detailed treatment.

To define adaptively generated data, the algorithm (or privacy game) takes an adversary $\mathcal{A}$ as input, along with a challenge bit b and a left-out record $x = (d_0, e_0, F_0)$. For each epoch $e = 1, 2, \ldots$ the adversary uses past results $v_{<e}$ to generate data $D^e = \mathcal{A}(v_{<e})$ for the new epoch. If $e = e_0$, the privacy game inserts x into the database iff $b = 1$. The privacy guarantees are stated by comparing the view of $\mathcal{A}$ across both worlds, *i.e.,* when $b = 0$ and $b = 1$.

To prove the privacy guarantees, we use the fact that the databases $D^{\leq e}_{b=0}$ and $D^{\leq e}_{b=1}$ differ by at most one element, since the adversary generates the same base data D^e at each step in both worlds, with at most one additional element x, once we condition on past results $v_{<e}$.

B.4 DoS resilience proofs

This section proves our main resilience result for Big Bird's quota-based online algorithm: Thm. 1. First, Lem. 3 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, Lem. 4 and 6 bound the total privacy budget the adversary can consume from the global filter at any qualified time, using the atomic consumption guarantees of Lem. 3. This final bound directly implies Thm. 1.

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

Lemma 3 (2-phase commit filter guarantees)**.** *For any individual report generation request (*e.g., *, for a report r processed by* GenerateReport *when invoked by* AnswerQuery *for a batch query Q^t in Alg. 2, or more generally, any call to* AtomicFilterCheckAndConsume *in Alg. 3 and Alg. 4), let:*

$$pass(r) = \begin{cases} 1 & \text{if AtomicFilterCheckAndConsume returns} \\ & \quad \textsf{TRUE} \text{ for report } r \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

If $pass(r) = 1$, then AtomicFilterCheckAndConsume *guarantees the following properties for that specific report r: The* AtomicFilterCheckAndConsume *function in Alg. 4 guarantees the following properties:*

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

1. ***Epoch-level Consistency Property:*** *exactly the same amount of budget ϵ_x^t is consumed by the per-site filter, global filter, and conversion-site quota-filter for that query.*
2. ***Epoch-site-level Consistency Property:*** *exactly $\epsilon_x^i[i]$ is consumed by the impression-site quota filter, which represents the device-epoch-impressionsite-level individual privacy loss.*

Proof. We can prove both properties at the same time. Fix an arbitrary individual report request, let's denote it by k for consistency within this proof, for which $\text{pass}(k) = 1$ (as defined in the lemma statement, meaning AtomicFilterCheckAndConsume returns TRUE for this report k). From Alg. 4, we observe that AtomicFilterCheckAndConsume returns TRUE for this individual report k if and only if: (1) all canConsume checks in Phase 1 pass, and (2) All tryConsume operations in Phase 2 are executed. For this individual report k, originating from conversion site c_k with beneficiary site b_k and intended impression sites $\mathbf{i}_k$, the function calls:

- $\mathcal{F}_{\text{per-site filter}}[b_k].\,\text{tryConsume}(\epsilon_x^{\text{report } k})$
- $\mathcal{F}_{\text{global filter}}.\,\text{tryConsume}(\epsilon_x^{\text{report } k})$
- $\mathcal{F}_{\text{conv-quota}}[c_k].\,\text{tryConsume}(\epsilon_x^{\text{report } k})$
- For each $i \in \mathbf{i}_k$: $\mathcal{F}_{\text{imp-quota}}[i].\,\text{tryConsume}(\epsilon_x^{\text{report } k}[i])$

Note that $\epsilon_x^{\text{report } k}$ is the device-epoch-level individual privacy loss computed for this specific report k (*e.g.,* , via EpochBudget in Line 9 of Alg. 3). Similarly, each $\epsilon_x^{\text{report } k}[i]$ is the device-epoch-site-level individual privacy loss for impression site i relevant to this report k (via EpochImpSiteBudget in Alg. 3). Therefore, when $\text{pass}(k) = 1$, the conversion-site quota filter (for c_k), the per-site filter (for b_k), and the global filter all consume exactly the same amount $\epsilon_x^{\text{report } k}$. Concurrently, each relevant impression-site quota filter (for $i \in \mathbf{i}_k$) consumes its specific amount $\epsilon_x^{\text{report } k}[i]$, which is proportional to its sensitivity at the impression site level for this report k. □

With such atomic guarantees for every individual report processed up to some batch query step t (as per Alg. 2), we can show upper bounds on how much an adversary can deplete the global filter by the end of batch query step t, on any device-epoch $x \in \mathcal{X}$. The main isolation result is in Thm. 1, which uses Lem. 4, 5 and 6.

Notation. First, we introduce some notation describing adversarial behavior, which we use in subsequent proofs:

- At step t in Line 17, suppose that beneficiary site b requests a report $\rho_{r,E,A}$ with noise λ through conversion site c for impression sites $\mathbf{i}$. Consider a device-epoch x, with individual budget ϵ_x^t computed at Line 9 in Alg. 3.
- A report at time step k concerns with *one* conversion site c_k, and a set of impression sites $\mathbf{i}_k \subseteq S$.
- The union of attackers can control an arbitrary subset of conversion sites $\mathbf{bad_c} \subseteq S$. 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 $\mathbf{bad_i} \subseteq S$. We denote by M^{adv} the size of $|\mathbf{bad_i}|$ over the entire lifetime. We let $\mathbf{bad} = \mathbf{bad_c} \cup \mathbf{bad_i}$ and $\mathbf{good} = S \backslash \mathbf{bad}$.
- 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 4. *Consider a sequence of T batch query steps. If an adversary, across all individual report generation attempts within these T steps, successfully leads to the instantiation (via* saveImpression *calls) and subsequent use (successful budget consumption from) of at most M^{adv} distinct imp-quota filters and N^{adv} distinct* `conv-quota` *filters in an attempt to deplete the global filter, then the adversary consumes at most*

$M^{\text{adv}}\epsilon_{\textit{imp-quota}}$ *budget from the global filter.*

Proof. The lemma considers a sequence of T batch query steps. Let s be the index for these batch query steps, from 1 to T. The total privacy loss in the global filter incurred by adversarial report generations up to (not including) batch query step T (i.e., after $T-1$ steps have completed) is:

$$\epsilon_{\text{global}}^{\leq T-1,\textbf{bad}} = \sum_{s=1}^{T-1} \sum_{r\in R_s: c_r \in \text{bad}_c \wedge \text{pass}(r,s)=1} \epsilon_{r,s}^{\text{global}}. \quad (36)$$

Here, s indexes batch query steps, R_s is the set of individual reports in batch s, c_r is the conversion site for report r, bad_c are adversarial conversion sites, $\text{pass}(r,s)=1$ indicates report r in batch s was successfully processed, and $\epsilon_{r,s}^{\text{global}}$ is the global budget portion consumed by that individual report r in batch s. By the consistency property of Lem. 3, for each successfully processed adversarial report r associated with a conversion site $c_r \in \text{bad}_c$, the filter $\mathcal{F}_{\text{conv-quota}}[c_r]$ precisely tracks the privacy loss $\epsilon_{r,s}^{\text{global}}$. Thus, we can write:

$$\mathcal{F}_{\text{conv-quota}}[c_r]^{\leq T-1} = \sum_{s=1}^{T-1} \sum_{r'\in R_s: c_{r'}=c_r \wedge \text{pass}(r',s)=1} \epsilon_{r',s}^{\text{global}}. \quad (37)$$

The quantity $\epsilon_{\text{conv-quota}}[c_r]^{\leq T-1}$ (defined as $\mathcal{F}_{\text{conv-quota}}[c_r]^{\leq T-1}$ in the preceding equation) represents the sum of all $\epsilon_{r',s}^{\text{global}}$ terms for reports r' associated with a specific conversion site c_r up to step $T-1$. Therefore, by summing $\epsilon_{\text{conv-quota}}[c]^{\leq T-1}$ over all adversarial conversion sites $c \in \textbf{bad}_\textbf{c}$, we are effectively re-summing all the individual $\epsilon_{r,s}^{\text{global}}$ contributions that constitute $\epsilon_{\text{global}}^{\leq T-1,\textbf{bad}}$. Thus, it follows directly from the definitions and equation (36) that:

$$\epsilon_{\text{global}}^{\leq T-1,\textbf{bad}} = \sum_{c\in\textbf{bad}_\textbf{c}} \epsilon_{\text{conv-quota}}[c]^{\leq T-1}. \quad (38)$$

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

$$= \sum_{c\in\textbf{bad}_\textbf{c}: \epsilon_{\text{conv-quota}}[c]^{\leq T-1}>0} \epsilon_{\text{conv-quota}}[c]^{\leq T-1} \quad (39)$$

$$\leq \sum_{c\in\textbf{bad}_\textbf{c}: \epsilon_{\text{conv-quota}}[c]^{\leq T-1}>0} \epsilon_{\text{conv-quota}}, \quad (40)$$

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

$$\leq \left|\left\{c \in \textbf{bad}_\textbf{c} : \epsilon_{\text{conv-quota}}[c]^{\leq T-1} > 0\right\}\right| \cdot \epsilon_{\text{conv-quota}} \quad (41)$$

$$= N^{\leq T-1,\text{adv}} \cdot \epsilon_{\text{conv-quota}}. \quad (42)$$

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,

$$\epsilon_{\text{global}}^{\leq T,\textbf{bad}} = \epsilon_{\text{used}}^{\textbf{bad}} + \epsilon_x^T \leq N^{\leq T,\text{adv}} \cdot \epsilon_{\text{conv-quota}}. \quad (43)$$

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

$$\epsilon_{\text{conv-quota}}[c_t]^{\leq T-1} + \epsilon_x^T \geq \epsilon_{\text{conv-quota}}, \quad (44)$$

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,

$$\epsilon_{\text{global}}^{\leq T,\textbf{bad}} = \epsilon_{\text{used}}^{\textbf{bad}} + 0 = \epsilon_{\text{used}}^{\textbf{bad}} \leq N^{\leq T-1,\text{adv}} \cdot \epsilon_{\text{conv-quota}}, \quad (45)$$

by equation (42).

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 □

Lemma 5 (Impression-Site Quota Allocation Consistency)**.** *Fix a record $x = (d, e, F)$ and a report ρ at step k. Denote by ϵ_x^k the epoch-level privacy loss given by EpochBudget, denote by $\epsilon_x^{\mathbf{i},k}[i]$ the epoch-impression-site-level privacy loss given by EpochImpSiteBudget (Def. 8) using $2\Delta(\rho)$ as an upper bound for $\Delta_i(\rho)$ (Lem. 2). We have:*

$$\epsilon_x^k \leq \sum_{i\in\mathcal{S}} \epsilon_x^{\mathbf{i},k}[i] \quad (46)$$

Proof. We go through the different cases for the upper bound on $\Delta_x(\rho)$:

- If $d \neq d_r, e \neq E_r$ or $F \cap F_A = \emptyset$, then $\epsilon_x^k = 0$. In that case, by Thm. 3 we also have $\epsilon_x^{\mathbf{i},k}[i] = 0$ for all i.
- If $d = d_r$ and $E_r = \{e\}$:
  - If $I_r = \{i\}$, we have $\epsilon_x^k = \epsilon_x^{\mathbf{i},k}[i] = \|A(F) - A(\emptyset)\|_1/\lambda$.
  - Else, we have $\epsilon_x^{\mathbf{i},k}[i] = 2\Delta(\rho)/\lambda \geq \|A(F) - A(\emptyset)\|_1/\lambda = \epsilon_x^k$.
- Else, we have $\epsilon_x^{\mathbf{i},k}[i] = 2\Delta(\rho)/\lambda \geq \Delta(\rho)/\lambda = \epsilon_x^k$.

□

Remark that the upper bound in Lem. 5 can be quite loose. Since the impression-site quota has no privacy meaning and is only used through Lem. 5 to obtain isolation guarantees, instead of using per-epoch-site privacy loss we could use any heuristic that also satisfies Lem. 5. For instance, we could define $\epsilon_x^{\mathbf{i},k}[i]$ by dividing ϵ_x^k uniformly across impression sites i with non-zero contributions.

Lemma 6. *Consider a sequence of T batch query steps. If an adversary, across all individual report generation attempts within these T steps, successfully leads to the instantiation (via* saveImpression *calls) and subsequent use (successful budget consumption from) of at most M^{adv} distinct imp-quota filters and N^{adv} distinct* `conv-quota` *filters in an attempt to deplete the global filter, then the adversary consumes at most $N^{\text{adv}}\epsilon_{\textit{conv-quota}}$ budget from the global filter.*

Proof. By basic composition under a pure DP filter, $\epsilon_{\text{global filter}}^{\leq t-1,\text{bad}}$ is the sum of global filter consumption by reports associated with adversarial conversion sites, bad_c across all query steps

s from 1 up to $t-1$. Let R_s be the set of reports in query step s, $c_{r,s}$ be the conversion site for report r from query step s, $\epsilon_{r,s}^{\text{global filter}}$ be the global budget consumed by that report, and $\text{pass}(r,s)$ indicate if it was successfully processed. Then,

$$\epsilon_{\text{global}}^{\leq t-1,\textbf{bad}} = \sum_{s\in[t-1]} \sum_{r\in R_s: c_{r,s}\in \textbf{bad}_\text{c}} \epsilon_{r,s}^{\text{global filter}} \cdot \text{pass}(r,s) \quad (47)$$

$$\leq \sum_{s\in[t-1]} \sum_{i\in I_x^s \cap \textbf{bad}_\text{i}} \epsilon_{\textbf{x}}^{\textbf{i},s}[i] \cdot \text{pass}(s), \quad (48)$$

where the last inequality follows from Lem. 5. First, by the restriction in the sum, we know s satisfies $c_s \in \textbf{bad}_\text{c}$. 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 $c_s \in \textbf{bad}_\text{c}$, then $\textbf{i}_s \subseteq \textbf{bad}_\text{i}$. Now, continuing where we ended in Eq. 48, we get:

$$\epsilon_{\text{global}}^{\leq t-1,\textbf{bad}} \leq \sum_{s\in[t-1]: c_s\in\textbf{bad}_\text{c}} \sum_{i\in\textbf{bad}_\text{i}} \epsilon_{\textbf{x}}^{\textbf{i},s}[i] \cdot \text{pass}(s) \quad (49)$$

$$= \sum_{i\in\textbf{bad}_\text{i}} \sum_{s\in[t-1]: c_s\in\textbf{bad}_\text{c}, i\in\textbf{i}_s} \epsilon_{\textbf{x}}^{\textbf{i},s}[i] \cdot \text{pass}(s), \quad (50)$$

$$\leq \sum_{i\in\textbf{bad}_\text{i}} \sum_{s\in[t-1]: i\in\textbf{i}_s} \epsilon_{\textbf{x}}^{\textbf{i},s}[i] \cdot \text{pass}(s), \quad (51)$$

by changing order of summation, and the last inequality by relaxing the "$c_s \in \textbf{bad}_\text{c}$" condition. But note that

$$\epsilon_{\text{imp-quota}}^{\leq t-1}[i] = \sum_{s\in[t-1]: i\in\textbf{i}_s} \epsilon_{\textbf{x}}^{\textbf{i},s}[i] \cdot \text{pass}(s), \quad (52)$$

because, by epoch-site-level consistency property in lemma 3, we know that only relevant site i at time s, 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 (51), we get:

$$\epsilon_{\text{global}}^{\leq t-1,\textbf{bad}} \leq \sum_{i\in\textbf{bad}_\text{i}} \epsilon_{\text{imp-quota}}^{\leq t-1}[i] \quad (53)$$

$$= \sum_{i\in\textbf{bad}_\text{i}: \epsilon_{\text{imp-quota}}^{\leq t-1}[i]>0} \epsilon_{\text{imp-quota}}^{\leq t-1}[i] \quad (54)$$

$$\leq \sum_{i\in\textbf{bad}_\text{i}: \epsilon_{\text{imp-quota}}^{\leq t-1}[i]>0} \epsilon_{\text{imp-quota}} \quad (55)$$

$$= \left|\left\{i \in \textbf{bad}_\text{i} : \epsilon_{\text{imp-quota}}^{\leq t-1}[i] > 0\right\}\right| \cdot \epsilon_{\text{imp-quota}}, \quad (56)$$

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

$$\epsilon_{\text{global}}^{\leq t-1,\textbf{bad}} \leq M^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{imp-quota}}. \quad (57)$$

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

- Suppose $\epsilon_x^t \leq \epsilon_{\text{imp-quota}}$,

$$\epsilon_{\text{global}}^{\leq t,\textbf{bad}} \leq M^{\leq t,\text{adv}} \cdot \epsilon_{\text{imp-quota}}. \quad (58)$$

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

$$\epsilon_{\text{global}}^{\leq t,\textbf{bad}} = \epsilon_{\text{global}}^{\leq t-1,\textbf{bad}} + 0 = \epsilon_{\text{global}}^{\leq t-1,\textbf{bad}} \leq M^{\leq t-1,\text{adv}} \cdot \epsilon_{\text{imp-quota}}, \quad (59)$$

by equation (57).

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:

$$\epsilon_{\text{global}}^{\leq t,\textbf{bad}} \leq M^{\leq t,\text{adv}} \epsilon_{\text{imp-quota}} \leq M^{\text{adv}} \epsilon_{\text{imp-quota}}, \quad (60)$$

as desired. □

We can now combine Lem. 4 and 6 to obtain our main isolation theorem:

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_{global}^{adv} that the adversary can consume from the global filter on a device d is such that:*

$$\epsilon_{global}^{adv} \leq \min(M^{adv}\epsilon_{imp\text{-}quota},\ N^{adv}\epsilon_{conv\text{-}quota}). \quad (61)$$

Proof. At any time t, if the adversary controls at most $M^{\leq t,\text{adv}} \leq M^{\text{adv}}$ `imp-quota` and $N^{\leq t,\text{adv}} \leq N^{\text{adv}}$ `conv-quota` filters, then by Lem. 4 and 6:

$$\epsilon_{\text{global}}^{\text{adv}} \leq M^{\leq t,\text{adv}} \epsilon_{\text{imp-quota}} \leq M^{\text{adv}} \epsilon_{\text{imp-quota}} \quad (62)$$

$$\epsilon_{\text{global}}^{\text{adv}} \leq N^{\leq t,\text{adv}} \epsilon_{\text{conv-quota}} \leq N^{\text{adv}} \epsilon_{\text{conv-quota}}. \quad (63)$$

Therefore:

$$\epsilon_{\text{global}}^{\text{adv}} \leq \min(M^{\text{adv}}\epsilon_{\text{imp-quota}},\ N^{\text{adv}}\epsilon_{\text{conv-quota}}). \quad (64)$$

□

C Batched Algorithm to Improve Utilization

C.1 Algorithm

Alg. 5 describes the batched algorithm on a single device. Instead of executing GenerateReport as soon as a request comes, as in Alg. 3, requests are accumulated in a batch. Each epoch is divided into k scheduling intervals, and since a request can request older epochs (up to a maximum attribution window length, *a.k.a.* data lifetime) we release budget progressively over T intervals. For instance, if requests have attribution window of at most 2 epochs, we can divide this data lifetime into $T = 4$ releases, with $k = 2$ releases happening inside each epoch. We can also do $T = 2$ releases with $k = 1$ interval per epoch.

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

We define $\mathcal{A}, \mathcal{U} \leftarrow$ TryAllocate($\mathcal{R}$) as follows. TryAllocate takes a set of report requests $\mathcal{R}$. For each request, it executes a heuristic that estimates whether Alg. 3's GenerateReport will successfully allocate budget for the request (*i.e.,* the whether the filters will return TRUE at Line 18). It

Algorithm 5 Batched Algorithm (On-Device)

Input:

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

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

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

then calls GenerateReports on the requests that were predicted to be allocatable, and returns two sets: $\mathcal{A}$ the reports for requests that were executed, and $\mathcal{U}$ the remaining unallocated requests. TryAllocateOne behaves like TryAllocate, except that it stops after the first executed request. Our heuristic is a variant of GenerateReport that relies purely on public information — using proxy filters $\tilde{\mathcal{F}}$ with IDP optimizations turned off — allowing the scheduler to make decisions without leaking privacy across epochs.

SortBatch attaches a weight (b_r, ϵ_r) to each request r in a batch, and then sorts by smallest weight first (in lexicographic order). The weights are defined as follows, using the proxy filters $\tilde{\mathcal{F}}$ define for TryAllocate. 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, where the budget consumed by i over the set of epochs E considered in the queue is defined by the maximum budget consumed by i over any epoch:

$$b_r := \min_{i \in \mathbf{i}_r} \max_{e \in E} \tilde{\mathcal{F}}^{\text{imp-quota}}[i].\text{consumed} \tag{65}$$

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

C.2 DoS resilience under batching

Under our mixed online and batch algorithm, the effort in terms of on device user interactions U^{adv} required from an adversary to consume global filter budget depends on the overall workload of the system. As we saw in §4.4, in the worst case (*e.g.,* when there are no legitimate queries in the system) this can lead to a weaker upper-bound on the budget consumption by an adversary compared to Thm. 2, with the following result:

$$\epsilon_{\text{global}}^{\text{adv}} \leq (1+r)\epsilon_{\text{per-site}} \times \texttt{quota-count} \times (U^{\text{adv}} - 1). \tag{66}$$

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

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

Theorem 5 (Graceful degradation for online queries under the batch algorithm)**.** *Consider a set of legitimate target queries,*

with total requested budget summing to ϵ^{good}_{global}. To deny service to those target queries, an attacker requires the following lower-bound on user interactions U^{adv}:

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

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

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

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

Reorganizing the terms concludes the proof. □

In addition, during the batch allocation phase (lines 9 to 14 in Algorithm 1), the adversary would still need overcome the scheduler's sorting mechanism, to be scheduled before waiting legitimate requests. Since the sorting mechanism favors low-budget and underrepresented impression sites, the adversary would likely require more than one user interaction ($U^{\text{adv}} \gg 1$) to mount an attack, making Equation 66 pessimistic. The time dynamics and workload dependency of the batching phases make the analysis of such guarantees challenging though, and we leave a proper formal treatment of any guarantees related to sorting for future work.

D Adaptive Cross-report Privacy Loss Optimization

In §B, we presented a general algorithm where beneficiaries pay for each report separately. This section formalizes the cross-report optimization from §4.2, by defining variations of Alg. 2, Alg. 3 in §D.2. We focus on histogram reports, defined in D.1, and show that paying only once for a sequence of properly correlated histogram reports still satisfies global DP guarantees in §D.3. We leave the generalization to other queries for future work.

D.1 Histogram reports definition and properties

The following definitions (Def. 11 and 12) are adapted from [28, Thm. 18]. Histogram reports distribute a positive value across impressions, map each impression to a bucket, and then sum up the attributed value in each bucket. Lem. 7 gives the global sensitivity of such histogram reports, which is bounded by the maximum attributable value.

Definition 11 (Scalar attribution function)**.** *Fix $k > 0$ a number of epochs. A* scalar attribution function *is a function $a : \mathcal{P}(\mathcal{I})^k \times \mathcal{I} \to \mathbb{R}_+$ that attributes a positive value $a_{\mathbf{F}}(f)$ to each impression $f \in \mathcal{I}$, depending on all the impressions in k epochs $\mathbf{F} \in \mathcal{P}(\mathcal{I})^k$.*

For a scalar attribution function a, we define its maximum attributable value $a^{\max}$ *as follows:*

$$a^{\max} := \max_{\mathbf{F}\in\mathcal{P}(\mathcal{I})^k} \sum_{j=1}^{k} \sum_{f\in\mathbf{F}_j} a_{\mathbf{F}}(f) \tag{67}$$

Definition 12 (Histogram report)**.** *Consider a scalar attribution function $a : \mathcal{P}(\mathcal{I})^k \times \mathcal{I} \to \mathbb{R}_+$, a support set of impressions $S \subset \mathcal{I}$, an output dimension $m > 0$, and 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 $\forall f \in \mathcal{I}, ||H(f)||_1 = 1$.*

First, we define $A_{a,S,H} : \mathcal{P}(\mathcal{I})^k \to \mathbb{R}^m$ as follows:

$$A(\mathbf{F}) = \sum_{j=1}^{k} \sum_{f\in\mathbf{F}_j} \mathbb{1}[f \in S] a_{\mathbf{F}}(f) \cdot H(f) \tag{68}$$

$A_{a,S,H}$ is a well-defined attribution function (in the sense of Def. 2).

Second, for a device d and a set of epochs E we define the histogram report *associated with $A_{a,S,H}$, as in Def. 3:*

$$\rho : D \mapsto A_{a,S,H}(D^E_d) \tag{69}$$

Next, Lem. 7 and 8 give two preliminary properties of histogram reports, that will be used in Thm. 6.

Lemma 7 (Histogram sensitivity)**.** *Consider a histogram report ρ with associated attribution function $A_{a,S,H}$. We have:*

$$\Delta(\rho) \leq 2a^{\max} \tag{70}$$

Proof. Take a report ρ with scalar attribution function a, device d and epochs E. Consider two neighborhing databases D, D' and denote $\mathbf{F} := D^E_d$ and $\mathbf{F}' := {D'}^E_d$. We have:

$$\|\rho(D) - \rho(D')\|_1 = \|A_{a,S,H}(\mathbf{F}) - A_{a,S,H}(\mathbf{F}')\|_1 \tag{71}$$

$$= \left\| \sum_{j=1}^{k} \sum_{f\in\mathbf{F}_j} a_{\mathbf{F}}(f) \cdot H(f) - \sum_{j=1}^{k} \sum_{f\in\mathbf{F}'_j} a_{\mathbf{F}'}(f) \cdot H(f) \right\|_1 \tag{72}$$

$$\leq \sum_{j=1}^{k} \sum_{f\in\mathbf{F}_j} a_{\mathbf{F}}(f)\|H(f)\| + \sum_{j=1}^{k} \sum_{f\in\mathbf{F}'_j} a_{\mathbf{F}'}(f)\|H(f)\| \tag{73}$$

$$\leq 2a^{\max} \tag{74}$$

Even though this bound holds even for non-neighborhing databases, [28, Thm. 18] provides mild conditions under which the bound is tight. □

D.2 Algorithm

Overview. Alg. 6 updates the formalism from Alg. 2. Instead of generating a report immediately upon conversion, the first beneficiary to request a report attached to a report identifier r calls MeasureConversion to create a stateful attribution object

Algorithm 6 Updated formalism for cross-report optimization (diff with Alg. 2)

// *Collect, aggregate and noise reports to answer Q*
function AnswerQuery(D, Q, λ, b)
  $(\rho_r^b)_{r\in R} \leftarrow Q$ // *Get report identifiers from Q*
  for $r \in R$ **do**
    if α_r is not defined **then**
      // *Initialize attribution object*
      $\alpha_r \leftarrow$ MeasureConversion(D, ρ_r^b, λ)
    $\rho_r^b(D), \alpha_r \leftarrow$ GetReport$(\alpha_r, \rho_r^b, \lambda, b)$
  Sample $X \sim \mathcal{L}(\lambda)$
  return $\sum_{r\in R} \rho_r^b(D) + X$

α_r, which pays upfront for any valid sequence of histogram reports over non-overlapping impressions.

MeasureConversion performs attribution using the pre-specified attribution function a and the noise scale λ, which give an upper bound on the total leakage of any sequence of future reports applying a on disjoint sets of impressions. We run through a modified two-phase commit protocol, to deduct budget from the global filter and the quotas, but not from the per-site filters. [3] Finally, α_r stores the attribution function $a_{\mathbf{F}}$ with corresponding impressions $\mathbf{F}$, the privacy parameters λ, and the support S of impressions requested so far.

GetReport allows a beneficary to receive a report from r, once the attribution object α_r has been created. α_r is only created once, and is reused every time a beneficiary requests a report from r. GetReport checks that privacy parameters match what was committed in the attribution object, and that impressions are not queried twice. If these checks pass, GetReport spends per-site budget and computes the report using the predefined attribution function a stored in α_r. We update the support of impressions $S \leftarrow S \sqcup S_\rho$ in α_r each time a new report ρ requests impressions in $S_\rho \subset \mathcal{I}$

Subroutine. We define AtomicFilterCheckAndConsume2 as in Alg. 4, except that per-site filters are not part of the atomic commit. That is, we do not take b as an input, and we delete Line 8 and Line 18 from Alg. 4.

D.3 Privacy proof

Lemma 8 (Correlated histogram sensitivity)**.** *Fix a device-epoch $x = (d, e, F)$ and a database D. Fix a report identifier $r \in \mathbb{Z}$ corresponding to a histogram attribution object α_r. Fix a sequence of reports $\rho_1, \ldots, \rho_n$ that request a report from r, ordered by lexicographically by time and beneficiary (t, b). In particular, MeasureConversion is called for ρ_1 and then reused for subsequent reports. $\sum_{i=1}^n \|\rho_i(D) - \rho_i(D + x)\|/\lambda_i$ represents the total contribution over reports computed from α_r, each with its own requested noise scale λ_i. Denote by $pass_1$ the output of the 2PC for x in Alg. 7.*

[3] The per-site budget is left out of the 2PC because we don't know ahead of time which beneficiaries will request a report, and we don't want to block some beneficiaries if other beneficiaries are out of budget.

Algorithm 7 Big Bird algorithm (on-device) with cross-report optimization for histogram reports

Input
  Filter and quota capacities ϵ_{global}, $\epsilon_{\text{per-site}}$, $\epsilon_{\text{imp-quota}}$, $\epsilon_{\text{conv-quota}}$
function MeasureConversion($D, \rho, \lambda, \mathbf{b}$)
  Read ρ to get device d, epoch E, conversion site c, impression sites $\mathbf{i}$, histogram attribution function $A_{a_\rho, S_\rho, H_\rho}$ with scalar attribution function a, support impressions S and histogram bin mapping H.
  for $e \in E$ **do**
    $x \leftarrow (d, e, D_d^e)$
    if $\mathcal{F}_x$ is not defined **then**
      $\mathcal{F}_x \leftarrow$ InitializeFilters$(\epsilon_{\text{global}}, \epsilon_{\text{per-site}}, \epsilon_{\text{imp-quota}}, \epsilon_{\text{conv-quota}})$
    $\epsilon_x \leftarrow 2a^{\max}$
    $\epsilon_{\mathbf{x}}^{\mathbf{i}} \leftarrow \{i : 2a^{\max}, i \in \mathbf{i}\}$
    if AtomicFilterCheckAndConsume2$(\mathcal{F}_x, c, \mathbf{i}, \epsilon_x, \epsilon_{\mathbf{x}}^{\mathbf{i}}) =$ FALSE **then**
      $F_e \leftarrow \emptyset$ // *Empty the epoch if any filter check fails*
  return $\alpha = (a_{\mathbf{F}}, \mathbf{F}, \lambda, \emptyset)$ // *Start with $S_\alpha = \emptyset$*

function GetReport(α, ρ, λ)
  Read α to get attribution function a_α, impressions $\mathbf{F}_\alpha$, noise scale λ_α.
  Read ρ to get device d, beneficiary site b, target epochs E, histogram attribution function $A_{a,S,H}$ with scalar attribution function a, support impressions S and histogram bin mapping H.
  if $S \cap S_\alpha \neq \emptyset \vee \lambda \neq \lambda_\alpha \vee a_\rho \neq a_\alpha$ **then**
    $\rho \leftarrow A(\emptyset, \ldots, \emptyset)$ // *Null report if inconsistent with α*
    return ρ
  for $e \in E$ **do**
    $x \leftarrow (d, e, \mathbf{F}_{\alpha e})$
    $\epsilon_x \leftarrow$ EpochBudget(x, ρ, λ) // *Per-site budget only*
    if $\mathcal{F}_x^{\text{per-site filter}[b]}$.tryConsume$(\epsilon_x^t) =$ FALSE **then**
      $F_e \leftarrow \emptyset$
    else
      $F_e \leftarrow \mathbf{F}_{\alpha e}$
  $\alpha \leftarrow (a_\alpha, \mathbf{F}, \lambda, S_\alpha \sqcup S_\rho)$ // *Update impression support*
  $\rho \leftarrow A_{a,S,H}((F_e)_{e\in E})$ // *Clipped attribution report*
  return ρ, α

We have:

$$\sum_{i=1}^{n} \|\rho_i(D) - \rho_i(D + x)\|/\lambda_i \leq pass_1 \cdot 2a^{\max}/\lambda_1 \tag{75}$$

Proof. If $\text{pass}_1 = 0$, then $\rho_i(D) = \rho_i(D + x)$ because in both cases the data for x is zeroed-out ($F_e = \emptyset$ at Line 12), and we're done.

Now, suppose that $\text{pass}_1 = 1$. Take a report ρ_i with scalar attribution function a_i and support impressions S_i. These only depend on past results $v_{<t_i,b_i}$. If $S_i \cap (S_1 \cup \cdots \cup S_{i-1}) \neq \emptyset$, $\lambda_i \neq \lambda_1$ or $a_i \neq a_1$, then $\rho_i(D) = A(\emptyset, \ldots, \emptyset) = \rho_i(D + x)$. Also, if $x = (d, e, F)$ is not queried by ρ_i ($d \neq d_i$ or $e \notin E_i$), then $\rho_i(D) = \rho_i(D + x)$.

Denote by I the set of remaining reports verifying $S_i \cap (S_1 \sqcup \ldots S_{i-1}) = \emptyset$, $\lambda_i = \lambda_1$, $a_i \neq a_1$, $d_i = d$ and $e \in E_i$. We have:

$$\sum_{i=1}^{n} \|\rho_i(D) - \rho_i(D + x)\|/\lambda_i \tag{76}$$

$$= \sum_{i \in I} \|\rho_i(D) - \rho_i(D + x)\|/\lambda_1 \tag{77}$$

$$= \sum_{i \in I} \| \sum_{j=1}^{k} \sum_{f \in \mathbf{F}_j} \mathbb{1}[f \in S_i] a_{\mathbf{F}}(f) H_i(f) - \tag{78}$$

$$\sum_{j=1}^{k} \sum_{f \in \mathbf{F}'_j} \mathbb{1}[f \in S_i] a_{\mathbf{F}'}(f) H_i(f) \|/\lambda_1 \tag{79}$$

$$\leq \sum_{i \in I} \sum_{j=1}^{k} \sum_{f \in \mathbf{F}_j} \mathbb{1}[f \in S_i](a_{\mathbf{F}}(f) + a_{\mathbf{F}'}(f))/\lambda_1 \tag{80}$$

$$\leq \sum_{j=1}^{k} \sum_{f \in \mathbf{F}_j} \mathbb{1}[f \in \sqcup_{i \in I} S_i](a_{\mathbf{F}}(f) + a_{\mathbf{F}'}(f))/\lambda_1 \tag{81}$$

$$\leq 2a^{\max}/\lambda_1 \tag{82}$$

by Def. 11 and using the fact that the S_i are disjoint so each impression is counted at most once.

□

Theorem 6. *Consider $x \in \mathcal{X}$ with global filter capacity ϵ_{global}. Then, $\mathcal{M}$ as defined in Alg. 6 satisfies individual device-epoch ϵ_{global}-DP for x under public information C.*

Proof. Take a device-epoch $x = (d, e, F) \in \mathcal{X}$ and a database D that doesn't contain (d, e). Denote by $x_C = (d, e, F \cap C)$ the device-epoch obtained by keeping only public events C from x, where public events are the set of all conversions. Take $v \in \text{Range}(\mathcal{M})$. As in Thm. 4, want to show that:

$$\left| \ln \left(\frac{\Pr[\mathcal{M}(D + x_C) = v]}{\Pr[\mathcal{M}(D + x) = v]} \right) \right| \leq \epsilon_{\text{global}}. \tag{83}$$

Using Bayes' rule and $\rho(D + x_C) = \rho(D)$, as in Thm. 4, we have:

$$\left| \ln \left(\frac{\Pr[\mathcal{M}(D + x_C) = v]}{\Pr[\mathcal{M}(D + x) = v]} \right) \right| \tag{84}$$

$$= \left| \ln \left(\frac{\prod_{t=1}^{t_{\max}} \prod_{b \in S} \Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D) + X_t^b = v_t^b]}{\prod_{t=1}^{t_{\max}} \prod_{b \in S} \Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D + x) + X_t^b = v_t^b]} \right) \right| \tag{85}$$

$$\leq \sum_{t=1}^{t_{\max}} \sum_{b \in S} \left| \ln \left(\frac{\Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D) + X_t^b = v_t^b]}{\Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D + x) + X_t^b = v_t^b]} \right) \right| \tag{86}$$

where each query $Q_t^b = \{\rho_{r,t}^b, r \in R_t^b\}$ is chosen adaptively, potentially based on previous results $v_1^b, \ldots, v_{t-1}^b$. Since the filters and attribution functions are identical for D and $D + x$ when we condition on past results, we write $\rho_{r,t}^b(D)$ for simplicity instead of $\rho_{r,t}^b(D; \mathcal{F}_{v<t}, a_{v<t})$.

Fix $t \in [t_{\max}]$ and $b \in \mathcal{S}$. Without the optimization, as in Eq. 32, the device would pay for each report sent to any beneficiary, which would give:

$$\left| \ln \left(\frac{\Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D) + X_t^b = v_t^b]}{\Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D + x) + X_t^b = v_t^b]} \right) \right| \tag{87}$$

$$\leq \sum_{r \in R_t^b} \Delta_x(\rho_{r,t}^b) \text{pass}_r^b / \lambda_t^b \tag{88}$$

Instead of upper-bounding the difference $\|\rho_{r,t}^b(D) - \rho_{r,t}^b(D + x)\|$ for each report by $\Delta_x(\rho_r)$ separately, which takes a maximum over all D right away, we keep information about D a bit longer. This will allow us to leverage the fact that reports $\rho_{r,t}^b$ across different timesteps and beneficiaries tied to a same identifier r are correlated:

$$\left| \ln \left(\frac{\Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D) + X_t^b = v_t^b]}{\Pr[\sum_{r \in R_t^b} \rho_{r,t}^b(D + x) + X_t^b = v_t^b]} \right) \right| \tag{89}$$

$$\leq \sum_{r \in R_t^b} \|\rho_r^b(D) - \rho_r^b(D + x)\|/\lambda_t^b \tag{90}$$

For a report identifier $r \in \mathbb{Z}$, we now define $\mathcal{T}_r$, which keeps track of all beneficiaries that requested a report from r and at which timesteps they requested it:

$$\mathcal{T}_r := \{(t, b) \in [t_{\max}] \times \mathcal{S} : r \in R_t^b\} \tag{91}$$

This notation allows us to swap the sums, after putting Eq. 90 into Eq. 86:

$$\left| \ln \left(\frac{\Pr[\mathcal{M}(D + x_C) = v]}{\Pr[\mathcal{M}(D + x) = v]} \right) \right| \tag{92}$$

$$\leq \sum_{t=1}^{t_{\max}} \sum_{b \in S} \sum_{r \in R_t^b} \|\rho_r^b(D) - \rho_r^b(D + x)\|/\lambda_t^b \tag{93}$$

$$= \sum_{r \in \mathbb{Z}} \sum_{(t,b) \in \mathcal{T}_r} \|\rho_r^b(D) - \rho_r^b(D + x)\|/\lambda_t^b \tag{94}$$

Fix a report identifier $r \in \mathbb{Z}$ corresponding to a histogram

attribution object α_r. Denote by t_0, b_0 the first time step and first beneficiary that requests a report from r, thereby calling MeasureConversion and obtaining ρ^* at Line **??** of Alg. 7. By Lem. 8, we have:

$$\sum_{(t,b)\in\mathcal{T}_r} \|\rho_r^b(D) - \rho_r^b(D+x)\|/\lambda_t^b \leq \text{pass}_{r,t_0}^{b_0} \cdot \Delta_x(\alpha_r)/\lambda_{t_0}^{b_0} \tag{95}$$

Finally, since $\text{pass}_{r,t_0}^{b_0}$ implies that $\epsilon_{r,t_0}^{b_0} = \Delta_x(\alpha_r)/\lambda_{t_0}^{b_0}$ passes the global filter, by definition of the global filter Eq. 94 becomes:

$$\left|\ln\left(\frac{\Pr[\mathcal{M}(D+x_C)=v]}{\Pr[\mathcal{M}(D+x)=v]}\right)\right| \tag{96}$$

$$\leq \sum_{r\in\mathbb{Z}} \text{pass}_{r,t_0}^{b_0}\epsilon_{r,t_0}^{b_0} \tag{97}$$

$$\leq \epsilon_{\text{global}} \tag{98}$$

which concludes the proof. □

Remark. Lem. 7 and 8 show that for histogram reports with $k > 1$ epochs, Alg. 7 spends up to $|\mathbf{b}|$ times less budget than Alg. 3 when $|\mathbf{b}|$ beneficiaries request reports from the same conversion with a single report identifier r. This is because the privacy loss in these cases is proportional to $\Delta(\rho) = 2a^{\max}$. For histogram reports with a single epoch, we can use the individual sensitivity, which renders this optimization unnecessary.

E Prototype Screeenshot

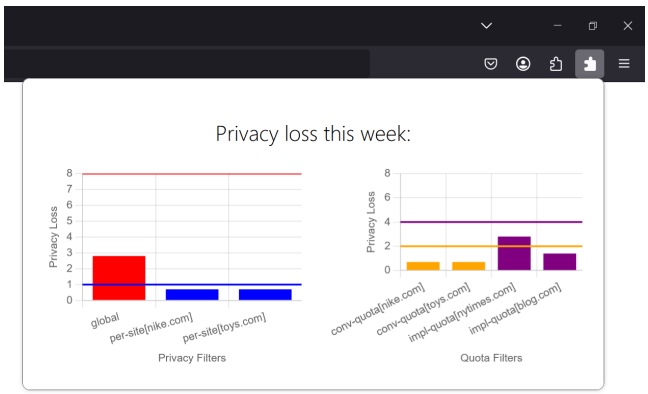


Fig. 5. Firefox privacy loss dashboard.

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

F Discussion regarding per-site DP guarantees

In §F.1 we provide a high-level intuition about how data and budget adaptivity impact per-site semantics. Next, in §F.2, we propose a strong assumption that is sufficient to prove per-site guarantees. The proof shows more precisely where adaptivity can cause leakage under adaptivity with no assumptions. Finally, we sketch some potential directions to maintain per-site guarantees under more realistic assumptions in §F.3. We leave a more formal and general treatment of the limitations of per-site guarantees in adaptive settings for future work.

F.1 Fundamental challenges

To show the global DP guarantees in Thm. 4, we considered the mechanism $\mathcal{M}$ all the outputs of Alg. 2. Thanks to the formalism from §A, Alg. 2 also defines one mechanism $\mathcal{M}^b$ per beneficary. It is thus possible to study the DP guarantees of $\mathcal{M}^b$, and ideally to show that $\mathcal{M}^b$ is $\epsilon_{\text{per-site}}$-DP.

Under adaptive data generation, §B.3 can be refined as follows. We consider a data generation process $\mathcal{G}$ and one adversary per beneficary $(\mathcal{A}_b)_{b\in\mathcal{S}}$. At each epoch e, new data D^e is generated by $\mathcal{G}$ based on the past results from all beneficiaries, *i.e.,* $D^e = \mathcal{G}(v_{<e})$. A challenge bit governs whether the game should introduce an additional record x. Then, at each time step t, each beneficiary $\mathcal{A}_b$ asks queries interactively depending on its own past results $v_{<t}^b$. Taking $D^e = \mathcal{G}(v_{<e})$ instead of something like $D^{e,b} = \mathcal{G}(v_{<e}^b)$ models the fact that in the most general case, whether an impression occurs depends on real-world actions that various beneficiaries take depending on their past results. For instance, news.ex might decide to display an impression for either shoes.ex or hats.ex, depending on the bid or creative for each site, where the bid from shoes.ex depends on shoes.ex's own past results.

As we will see more formally in the proof for Thm. 7, there are two shared data structures that depend on results from all beneficiaries: the dataset itself, and the global filters and quotas. Each of these data structures can act as a side-channel: if a beneficiary b_1 writes down some information gained through its own queries $v_{<t}^{b_1}$ (after paying up to ϵ_{global}), different beneficary b_2 can later read this information, which can affect results beyond the leakage permitted through b_2's own budget.

F.2 Guarantees under additional assumption

Assumption 1. Assume the two following properties:

- First, data is not generated adaptively, *i.e., D* is fixed upfront as in Alg. 2 instead of being generated by a process that depends on past results $v_{v<t}$ as described

in §B.3.

- Second, given any beneficiary b and past views $v_{<t}$ that asks query Q on database D with filters $\mathcal{F}$, the results from other beneficiaries do not impact which reports get filtered:

$$Q_t^b(v_{<t}^b)(D;\mathcal{F}(D,v_{<t}) = Q_t^b(v_{<t}^b)(D;\mathcal{F}(D,v_{<t}^b) \tag{99}$$

This condition is achieved if queries are chosen non-adaptively for queriers other than b, or if the global filter and quotas are never triggered on any device in the query.

Theorem 7 (Per-Beneficiary DP Guarantee)**.** *Consider the view of a single beneficiary $b \in \mathcal{S}$ in Alg. 2, which defines a mechanism $\mathcal{M}_b$. Consider $x \in \mathcal{X}$ with impression-site quota capacity $\epsilon_{\text{imp-quota}}$. Given Assumption 1, the mechanism $\mathcal{M}_b$ satisfies individual device-epoch $\epsilon_{\text{per-site}}$-DP for x with respect to public information C_b.*

Proof. As in the proof for Thm. 4, take a device-epoch $x = (d, e, F) \in \mathcal{X}$ and a database D that doesn't contain (d, e). Denote by $x_{C_b} = (d, e, F \cap C_b)$ the device-epoch obtained by keeping only public events C from x, where public events are the set of all conversions for b. Take $v \in \text{Range}(\mathcal{M})$ (the global mechanism) and denote by v^b the outputs in v that are sent to beneficiary b. Our goal is equivalent to showing that:

$$\left|\ln\left(\frac{\Pr[\mathcal{M}^b(D + x_C) = v^b]}{\Pr[\mathcal{M}^b(D + x) = v^b]}\right)\right| \leq \epsilon_{\text{per-site}}. \tag{100}$$

For any database D', with Bayes's rule we have:

$$\Pr[\mathcal{M}^b(D') = v^b] \tag{101}$$

$$= \Pr[\mathcal{M}^b(D') = v_{\text{pub}}^b] \cdot \prod_{t=1}^{t_{\max}} \Pr[\mathcal{M}_t^b(D') = v_t^b | v_{<t}^b]. \tag{102}$$

However, we can't directly decompose $\mathcal{M}_t^b$ into the query at time t conditioned purely on past results from v_t^b. Indeed, Big Bird maintains a global filter, which is mutable state that gets updated after each query, and is shared across beneficiaries. Instead, we have:

$$\Pr[\mathcal{M}_b(D)|v_{<t}^b] = \int_{u_{<t}:u_{<t}^b = v_{<t}^b} Q_t^b(v_{<t}^b)(D;\mathcal{F}(D,u_{<t}) + X_t^b(v_{<t}^b) du_{<t} \tag{103}$$

Note that the query Q_t^b and the privacy parameters for X_t^b only depend on $v_{<t}^b$, because that's the only information a non-colluding beneficiary can use to formulate its request. However, the state of the filters $\mathcal{F}(D, v_{<t})$ depends on queries from all the other beneficiaries $v_1^{<t}, \ldots, v_{b-1}^{<t}$. Moreover, if the data was generated adaptively as in §B.3, by a process $D^e \leftarrow \mathcal{A}(u_{<t})$ depending on the view of all beneficiaries, then we would still need to integrate over $u_{<t}$.

By Assumption 1, the results from other beneficiaries do not impact which reports get filtered:

$$Q_t^b(v_{<t}^b)(D;\mathcal{F}(D,v_{<t}) = Q_t^b(v_{<t}^b)(D;\mathcal{F}(D,v_{<t}^b) \tag{104}$$

Then, we can remove the integral and condition only on $v_{<t}^b$: $\Pr[\mathcal{M}_b(D)|v_{<t}^b] = Q_t^b(v_{<t}^b)(D;\mathcal{F}(D,v_{<t}) = Q_t^b(v_{<t}^b)(D;\mathcal{F}(D,v_{<t}^b)$. We omit the $v_{<t}^b$ input next, since it is identical whether the input is $D + x$ or $D + x_{C_b}$. Eq. 101 thus becomes:

$$\Pr[\mathcal{M}_b(D') = v] = \Pr[\mathcal{M}^b(D') = v_{\text{pub}}^b] \tag{105}$$

$$\cdot \prod_{t\in[t_{\max}]} \Pr\left[\sum_{r\in R_t^b} \rho_r(D';\mathcal{F}_{t,r}^b) + X_t^b = v_t^b\right]. \tag{106}$$

As in Thm. 4, we can bound the privacy loss ϵ_t at any given time $t \in [t_{\max}]$ by the property of Laplace distribution:

$$\left|\ln\left(\frac{\Pr\left[\sum_{r\in R_t^b} \rho_r(D) + X_t^b = v_t^b\right]}{\Pr\left[\sum_{r\in R_t^b} \rho_r(D+x) + X_t^b = v_t^b\right]}\right)\right| \leq \sum_{r\in R_t^b} \text{pass}_r \Delta_x \rho_{r,t}^b/\lambda \tag{107}$$

$$\leq \sum_{r\in R_t^b} \text{pass}_r \epsilon_r \tag{108}$$

Since pass_r implies that r passes $\mathcal{F}^{\text{per-site filter}}$ succesfully, we get:

$$\sum_{t\in[t_{\max}]:b_t=b} \sum_{r\in R_t} \epsilon_r \text{pass}_r \leq \epsilon_{\text{per-site}}. \tag{109}$$

Finally, using the fact that the public information is identical across both worlds, we have:

$$\left|\ln\left(\frac{\Pr\left[\mathcal{M}_b(D + x_C) = v\right]}{\Pr\left[\mathcal{M}_b(D + x) = v\right]}\right)\right| \tag{110}$$

$$\leq \sum_{t\in[t_{\max}]} \left|\ln\left(\frac{\Pr\left[\sum_{r\in R_t} \rho_r(D) + X_t = v_t\right]}{\Pr\left[\sum_{r\in R_t} \rho_r(D+x) + X_t = v_t\right]}\right)\right| \tag{111}$$

$$\leq \sum_{t\in[t_{\max}]} \epsilon_t \leq \sum_{t\in[t_{\max}]} \sum_{r\in R_t} \epsilon_r \text{pass}_r \tag{112}$$

$$\leq \epsilon_{\text{per-site}}. \tag{113}$$

□

It is important to note that the $\epsilon_{\text{per-site}}$-DP guarantee of Thm. 7 relies critically on the assumption for per-beneficiary analysis. If this assumption does not hold (e.g., if other beneficiaries can adaptively influence the global filter in a way that helps a beneficiary b learn more than its share), the formal $\epsilon_{\text{per-site}}$-DP guarantee may be compromised. In such scenarios, while the per-site filter still limits what b can learn, it becomes a valuable heuristic instead of a formal differential privacy guarantee.

F.3 Future work

While Assumption 1 from §F.2 is sufficient to prove per-beneficiary DP guarantees, it might not be necessary. We can imagine more realistic assumptions, especially if we constrain the class of queries. For instance, we could allow a form of siloed adaptive data generation, where each beneficary generates data $D_b^e \leftarrow \mathcal{G}_b(v_{<e}^b)$ based on its own past results $v_{<e}^b$ only, and each beneficary can only read its own data $Q^b(D) = Q^b(D_b)$. Additionally, it might be possible to analyze Eq. 103 more tightly, for instance if the shared filters and quotas are sufficiently noisy or if they are guaranteed to only impact a small number of reports.