Interoperable Private Attribution (IPA)

Date Published: Jan 5th, 2022
Authors: Erik Taubeneck *(Meta),* Ben Savage *(Meta),* Martin Thomson *(Mozilla)*

- Purpose of this document:

1. Introduction
   - 1.1 Major design choices
   - 1.2 Acknowledgements

2. Components of the IPA proposal
   - 2.1 Setting Match Keys
   - 2.2 Event Generation
     - 2.2.1 Source Events
     - 2.2.2 Trigger Events
   - 2.3 Event Collation Model
   - 2.4 Event Attribution
     - 2.4.1 Joining Events with encrypted Match Keys
       - 2.4.1.1 Joining Events Cryptographic Protocol
     - 2.4.2 Attribution
   - 2.5 Measurement Results
     - 2.5.1 Source Queries and Trigger Queries
       - 2.5.1.1 Source Queries
       - 2.5.1.2 Trigger Queries
       - 2.5.1.3 Networks with Source and Trigger Queries
     - 2.5.2 Advanced Aggregation
     - 2.5.3 Result Contents

3. Privacy Considerations
   - 3.1 Differential Privacy
   - 3.2 Privacy Budgeting
   - 3.3 Privacy Grain

4. Security Considerations
   - 4.1 Threat Model
   - 4.2 Attacks and Mitigations
     - 4.2.1 Known Secret Share Value
     - 4.2.2 Known Event Cardinality
     - 4.2.3 Known Linear Relationships

5. Robustness Considerations
   - 5.1 Anti-Poisoning

5.2 Fraud Protection

6. Extensions
6.1 Multiple Match Keys
6.1.1 Attribution with Multiple Match Keys
6.1.2 Privacy Considerations with Multiple Match Keys
6.2 Extension: Model Training and Conversion Optimization
6.3 Extension: Compatibility with Fenced Frames
6.4 Extension: Business Privacy Grain
6.4.1 Defining a business
6.4.2 Delegation to business entities
6.4.3 Moving the privacy grain to business
6.4.4 Comparison with First Party Sets

Purpose of this document:

A draft overview of a web platform proposal for private, interoperable attribution (IPA.) It is intended as a proposal for consideration in the Private Advertising Technology Community Group (PATCG). We are publishing it for feedback, and hope to continue to improve it among the Community Group, as well as anyone else interested in such a standard.

We also have a less technical presentation which provides an introduction to the idea, which may be of interest to readers.

1. Introduction

In designing IPA, we set out to find a win-win-win solution for cross platform attribution measurement that met our goals across **privacy**, **utility**, and **competition**.

Our **privacy** goal is to *limit the total amount of information* IPA releases about an individual over a given period of time. We want to be able to make strong claims about the amount of information, even in the presence of an adversary that is willing to engage in fingerprinting, navigational-tracking, registering a large number of domains, or other attacks.

Our **utility** goal is to support all the major *aggregate conversion measurement* use-cases (view-through, click-through, return-on-ad-spend, conversion-lift, cross-publisher attribution), including in cases where ad impressions and ad conversions happen in different browsers or devices. While beyond the initial scope of this proposal, IPA could also be extended to support other forms of post-attribution aggregation, such as model training and other forms of sophisticated inference, which we explore in the *Extensions* section 6.2.

Our **competition** goal is to ensure all the utility use-cases listed above would function for *all digital advertising players*. Furthermore, we wanted to avoid designs that would create barriers to entry for new players.

1.1 Major design choices

These goals led us to make the following major design decisions:

- Matching of ad interactions (*source events*) and conversions (*trigger events*) happens within a server-side secure multiparty computation (MPC), rather than on-device.
- Random noise must be added to the aggregate result to provide ε-differential privacy.
- ε-differential privacy needs to extend to the entire system over a fixed window of time, not just for individual queries, and thus the system needs to manage a "privacy budget."
- To ensure events can be joined across browsers, devices, and even app and web, there must be some kind of *match key*.
    - We propose a write-only field that can be set by apps and websites that people commonly log-in to across devices.
    - To achieve our competition goals, we propose allowing any ad-tech player to use (though never see) a *match key* set by any other ad-tech player.

1.2 Acknowledgements

We'd like to thank a number of people who also contributed and provided feedback to this document.

Benjamin Case (Meta) provided considerable input into the cryptographic protocol described in section 2.4.1.1. Eric Rescorla (Mozilla), Vikash Rungta (Meta), Sanjay Saravanan (Meta), Dennis Buchheim (Meta), Danny Frenkel (Meta), Casey Beal (Meta), Katie Glass (Meta), and Dominic Cooney (Meta) all contributed valuable feedback and suggestions throughout this document.

We would also like to call out the work happening in the WICG on the Attribution Reporting API and Privacy Preserving Ads, which was highly influential to this work. In particular, we'd like to acknowledge Charlie Harrison (Google), John Delaney (Google), and Mariana Raykova (Google) for their work on the Aggregate Reports proposal, and Erik Anderson (Microsoft) and Mehul Parsana (Google - formerly Microsoft, at the time the proposal was published) for their work on Multi-party Computation of Ads on the Web (MaCAW).

2. Components of the IPA proposal

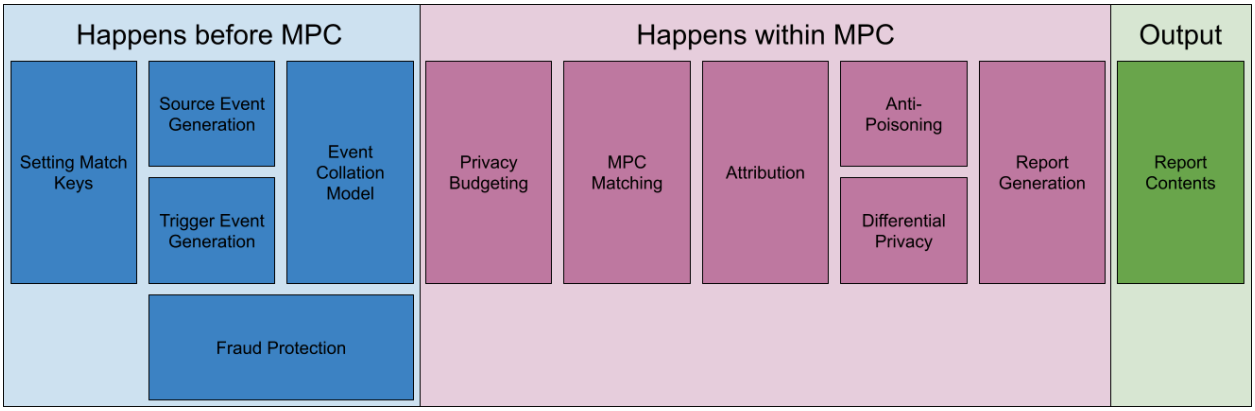


Figure 1: Components of IPA

2.1 Setting Match Keys

The intention is to have an **interoperable API**, such that every browser / mobile device used by a particular person is capable of generating standardized *ad impression* and *ad conversion* reports which are **only joinable within a specific secure multi-party computation** designed to use the proposed Privacy Preserving Measurement protocol, with helper servers that the user - or user agent - trusts.

Aggregate attribution measurement is the process of producing summary statistics about events that occur on different websites. We want to extend that to allow these events to occur in different browsers, apps and browsers, or in different apps. In order to match these events, we need something in common between different contexts. To this end, we have introduced the concept of a *match key*.

This *match key* could be set by any apps/websites, most usefully by those that people commonly log-in to on multiple devices. We propose that any entity be able to reference a *match key* set by any other entity.

As a strawman, imagine that this API might look like:
`navigator.setMatchKey( match_key )`

If the eTLD+1 of that window were `site.example`, then the browser would add to local private storage the mapping of `{ site.example: match_key }`.

To achieve our privacy goals, it's critical that this *match key* is not readable. It should only be visible to the browser or OS. The value is only used for the purpose of generating events which can be joined within the MPC. It need not, however, be high entropy or unguessable. For instance, it could be the user's site-specific user_id.

The *match key* would be:

- Consistent across devices, scoped to a match key provider
- Writeable
- Not readable
- Limited to purpose constrained aggregations via MPC.

This is different from current global identifiers like IDFA which are:

- Device scoped
- Not writeable
- Readable
- Usable for purposes beyond aggregate attribution measurement

While 3rd party cookies can be set consistently across multiple browsers / devices to enable cross-device measurement, they differ in important ways. 3rd party cookies are:

- Readable
- Scoped to just the entity which set them
- Usable for purposes beyond aggregate attribution measurement

Match keys are also a critical component of the privacy properties of the proposed aggregate measurement solution, as outlined in Privacy Considerations. Additionally, we also explore an extension that would allow the reference of Multiple Match Keys.

2.2 Event Generation

We propose a new API that enables developers to generate standardized *source events* (ad impressions) and *trigger events* (ad conversions) from apps and websites.

As a strawman, imagine that these APIs might look like:

```
navigator.generateSourceEvent( match_key_provider )
navigator.generateTriggerEvent( conversion_value,
match_key_provider )

AttributionManager.sharedInstance().generateSourceEvent(keyProvider
: match_key_provider)
AttributionManager.sharedInstance.generateTriggerEvent(conversionVa
lue: conversion_value ,keyProvider: match_key_provider)
```

A few points in common between *source* and *trigger events*:

- **No new information.** These events are intended to reveal no new information. The information they contain is encrypted for the MPC helpers.

- **No delays.** As these events reveal no new information, they do not need to be randomly delayed, or IP-blinded. They can be returned immediately in the same context where the API was invoked.
- **Only usable within MPC.** These events are not usable outside of the specific MPC servers that the user - or user agent - trusts.
- **Not reliant on rate limiting.** The goal of the API is to provide ε-differential privacy to individual users on the web platform. This is achieved through managing a privacy-budget for each user and ensuring that aggregation functions are ε-differentially private. Our aim is to achieve this in a way which would be robust against any number of reports being generated (though some amount of rate limiting may be useful against other forms of abuse.)

2.2.1 Source Events

A *source event* is an event which precedes a *trigger event* that the participating websites and apps may find useful within an aggregate attribution measurement. *Source events* need not be limited to just clicks, or *viewable impressions*, but could also include *ad opportunities* where no ad was shown at all.

This proposal is agnostic on the specific measurement methodology, and ideally could support many of the key types of aggregated conversion measurement including *view-through*, *click-through*, and *conversion lift*. As such, the generation APIs would be callable anytime by apps / websites, and not rely on the browser / mobile-OS vetting certain actions, such as link clicks or pixels on screen.

It should be possible for websites and apps to delegate the generation of *source events* to 3rd party service providers, such as ad-networks. As it is common for publishers to work with multiple ad-networks, who compete for the opportunity to show ads through an auction. Publisher websites should be able to delegate the generation of source events to multiple entities.

Like *trigger events*, source events could have an associated value. Our proposal does not currently include any source value, but we do consider this in section 6.3.

2.2.2 Trigger Events

A *trigger event* is an event which the participating websites and apps wish to measure in relation to *source events*. Once again, the intention is to support a wide variety of use-cases by not overly specifying what these *trigger events* indicate. They might represent *purchases* or *app installs* or *sign-ups* or possibly even just *page views* - anything a business might be trying to drive through advertising.

Trigger events are also able to contain a *trigger value*, a numerical value that represents something of meaning to the business generating the conversion. Our strawman proposal is to

limit this to 8-bits, and to utilize the ideas in prio3 to prevent poisoning, but we hope this will evolve with feedback.

As with *source events*, it should be possible for websites and apps to delegate the generation of *trigger events* to 3rd party service providers that help manage event collection and query execution.

2.3 Event Collation Model

IPA proposes logging that is no different from what is possible today, except that global identifiers are removed, and encrypted blobs are added.

Source events can be collected in real-time and logged together with other metadata about the ad impression (such as creative ID, time stamp, context, etc.). Similarly, *trigger events* can be collected in real-time and logged together with other metadata about the conversion, such as the product-ID purchased, the time of the purchase, the total amount spent, etc.

Here is an overall system architecture, which we aim to make compatible with the proposed Privacy Preserving Measurement architecture:

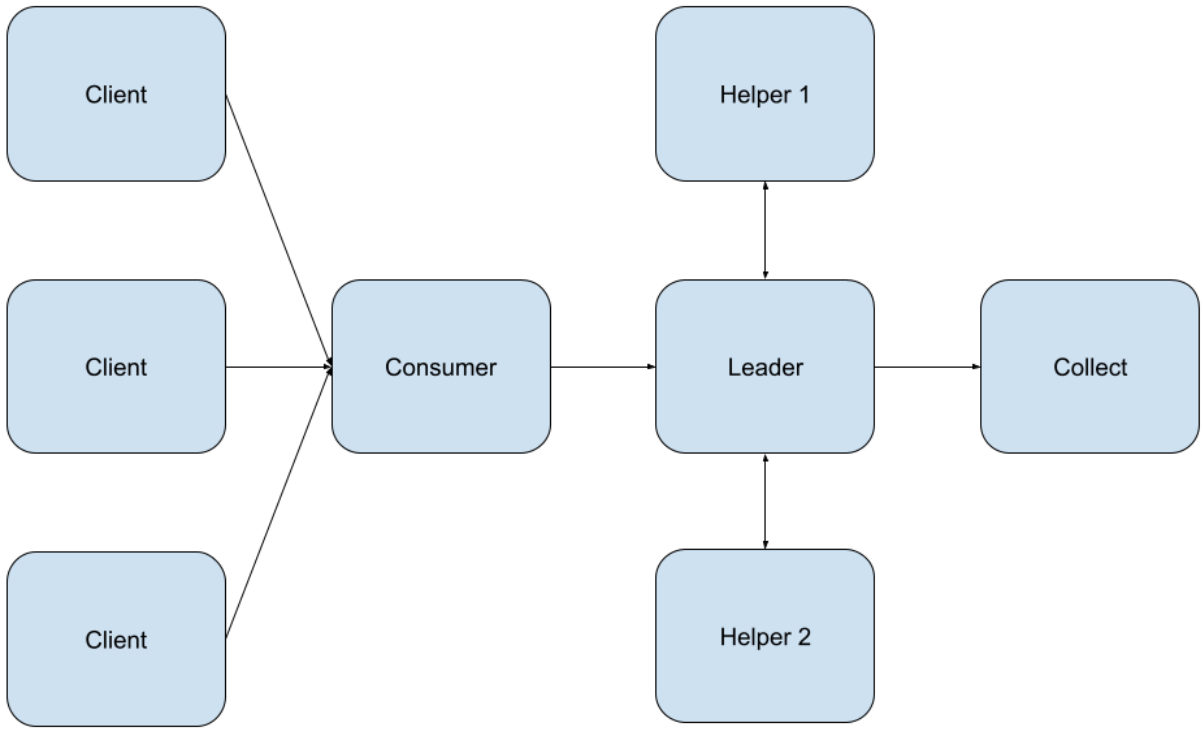


Figure 2: System Architecture

The work of joining source and trigger events together to perform attribution is delegated to the MPC. IPA would only support batch-processing of events in queries which would return aggregate results (e.g. counts and sums).

Businesses (both ad-tech vendors who facilitate selling ads, and merchants who purchase ads) will collect large batches of these encrypted *source* and *trigger events* on their servers. Periodically, they will make queries, by selecting a set of *source* and *trigger events* and initiating the IPA Privacy Preserving Measurement protocol to aggregate across those events.

One divergence from the status quo is the assumption that **both** ad-sellers and ad-buyers will issue queries. In the status quo world, it's common for ad-buyers to forward *conversion events* to ad-sellers, but much less common for ad-sellers to forward *ad impressions* to ad-buyers. However, given that events are all cryptographically protected, and can only be used in a specific MPC with known output, there is no longer a privacy rationale to prevent forwarding of events to ad-buyers as well.

In our proposed design, ad-sellers and ad-buyers can each make queries over the same events without any need for coordination.

2.4 Event Attribution

2.4.1 Joining Events with encrypted *Match Keys*

While other aggregate attribution measurement proposals have proposed joining *source* and *trigger events* on-device, we propose joining these events within a secure multi-party computation. This is the key innovation of the IPA proposal. There are a number of motivations for this:

1. **Improved Privacy**. When joined-up data leaves a device, fingerprinting attacks can be used to reconstruct user-level cross-site data. Mitigations like adding random time-delays, limiting entropy, and hiding IP addresses are only somewhat helpful at mitigating such attacks, and come at significant costs to ad-buyers who rely on timely data.
2. **Simplified Fraud Prevention**. On-device matching introduces new challenges in fraud detection and prevention. While token-based schemes might improve the situation, there is no clear way to integrate this approach with differential privacy.
3. **Cross-Device**. On-device attribution joins a *source event* and a *trigger event* on a single device, and almost entirely excludes joining events that occur on different devices. By moving the join into the MPC, it's trivially easy to make a cross-device API. We simply need to standardize the format of the *source events* and *trigger events* generated by browsers / mobile devices. Facebook has publicly shared data about

how common it is for user-journeys to span multiple user-agents and devices on the “path to conversion”.

4. **Immediate**. With this approach, there is no need for the user agent to be active at some future point in time, days later, to submit the report. This is a likely source of lost information in other approaches like PCM. The only latency in the system is the time that it takes to accumulate enough reports to aggregate into a useful result.

At a high-level, the design is intended to limit the information to which the helper servers have access. The proposed protocol extends the cryptography ideas of Private Set Intersection so that helper servers eventually have *source* and *trigger events* which are joinable, but the events are unlinkable to anything else, including the original encrypted reports. In this way, helper servers are able to collect groups of events that all relate to the same person, but are unable to discern who that person is.

2.4.1.1 Joining Events Cryptographic Protocol

Here we provide a sketch of the proposed underlying cryptographic protocol which enables the joining of *source* and *trigger* events within the MPC. This is only a sketch and is not intended to provide details on the entire flow of the API. We currently only address the joining of *match keys*, and not the associated data (e.g. timestamps and trigger values.) The associated data would also need to be encrypted and re-randomized in a similar fashion so that they cannot be used to identify specific *source* or *trigger reports.*

We intend to continue to publish more details, along with a technical paper with more details on this protocol.

We use the following three functions:

- `EncryptHE/DecryptHE`: A homomorphic encryption scheme with threshold encryption, e.g., ElGamal
    - Each helper has a public and secret key pair: `helper_i_pk, helper_i_sk`
    - The public keys can be combined to a threshold public key: `helpers_pk`
- `Blind`: A pseudo-random function (PRF) which commutes with the HE function, i.e. Elliptic Curve exponentiation. Helpers should use a different, random blinding function per batch to ensure data is not linkable across batches.
- `Shuffle`: A random reordering of an array.

With each report that is generated by a client, the *match key* is encrypted with EncryptHE with the threshold key. As shown in *Figure 2*, the client sends the encrypted match key (as part of the report) to a *Consumer*, which is determined by the site/app generating the report.

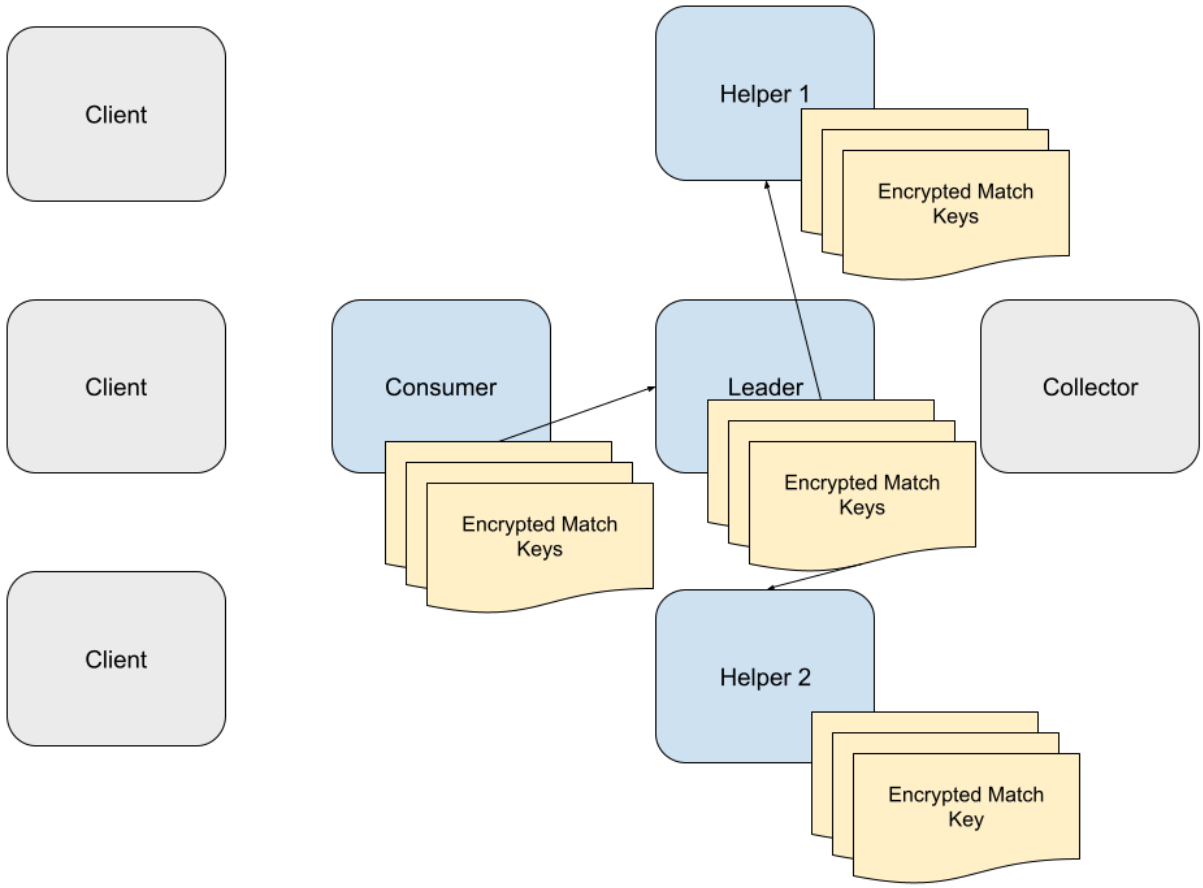


Figure 3: Client side encryption.

Once the *Consumer* has collected enough reports to run a query, the reports are forwarded into the MPC aggregation. The *Consumer* sends the reports to the *Leader*, who then distributes the reports to *Helper 1* and *Helper 2.*

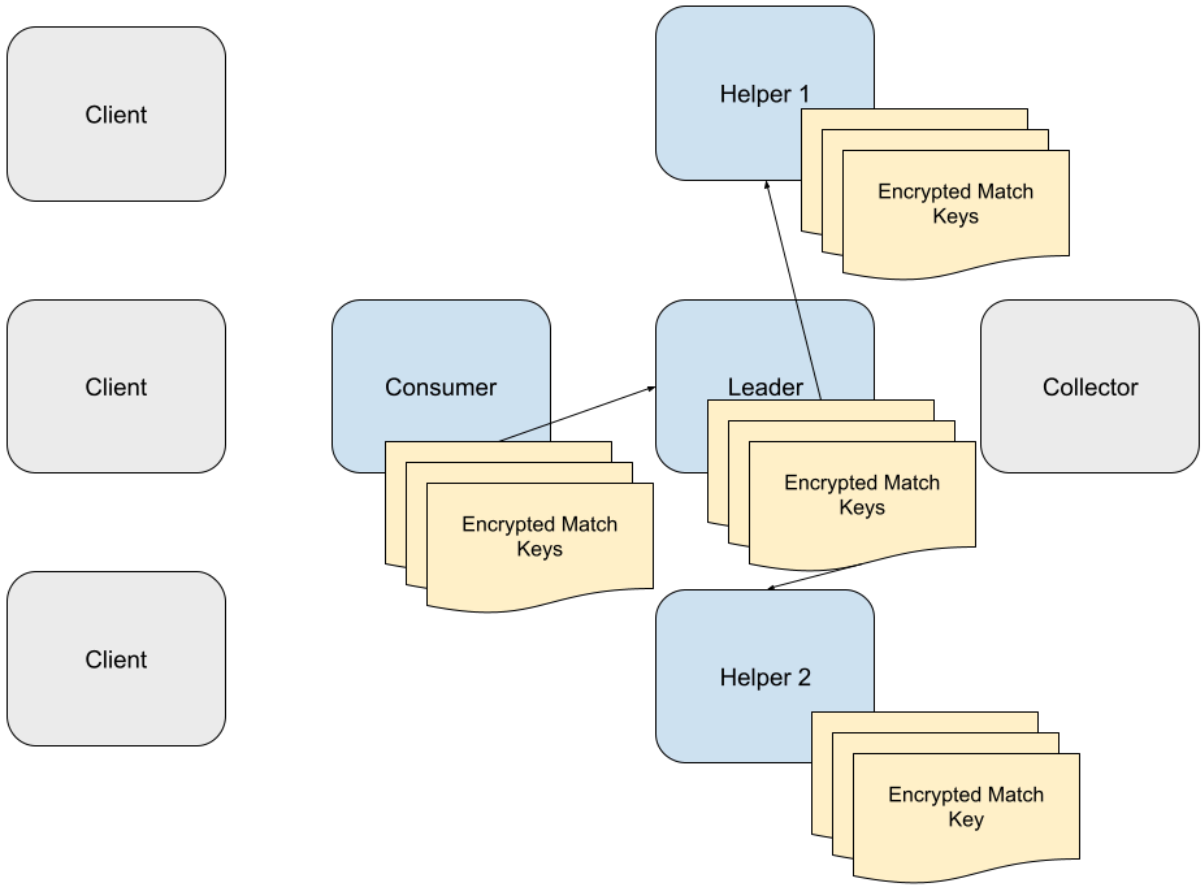


Figure 4: Beginning the Aggregation MPC

The following is a simplified overview of the protocol for arriving at an unlinkable *match key* for attribution. It leaves out more detail which would be required for other security and privacy considerations, such as handling associated data.

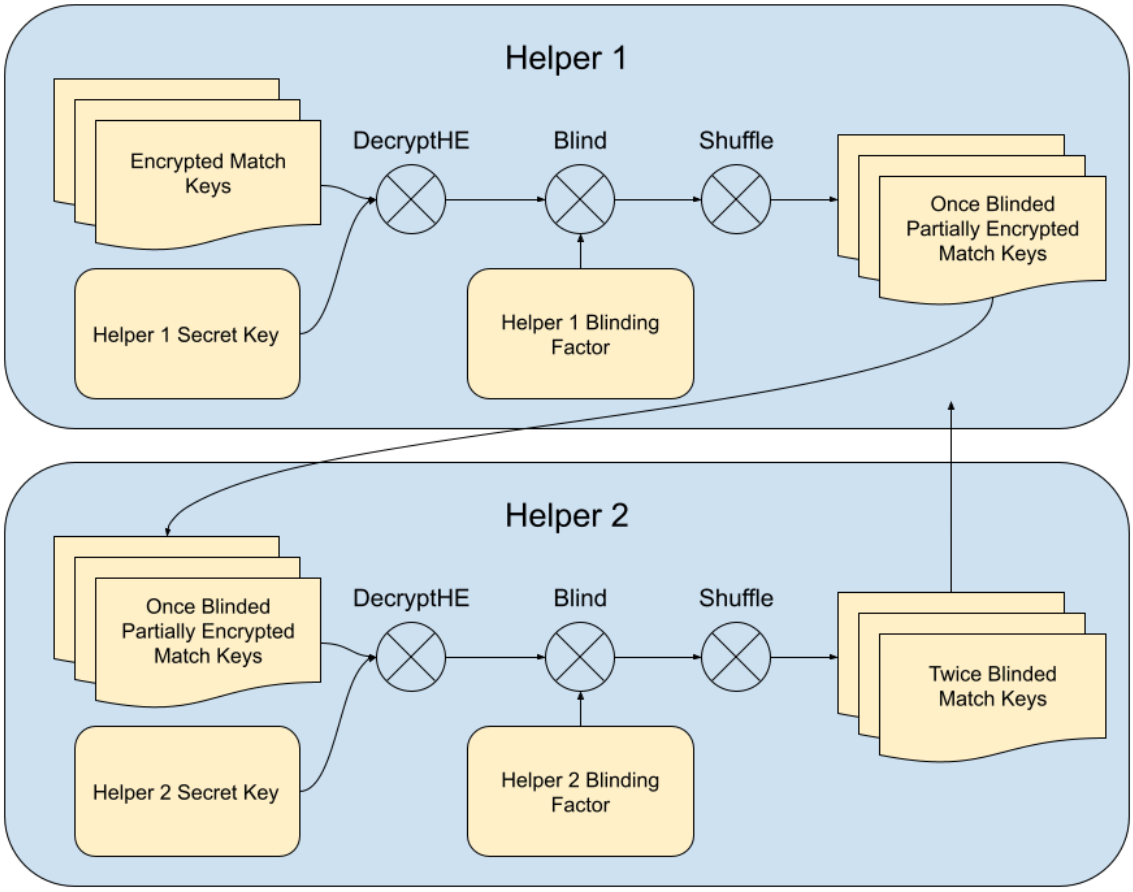


Figure 5: Unlinkable match key generation process.

Because the blinding process is deterministic, *twice blinded match keys* will be the same if and only if the original *match keys* had the same value. However, it should not be possible to determine the original *match key* value for any *twice blinded match key* without knowing both of the blinding factors.

2.4.2 Attribution

Attribution is typically more involved than simply joining *source* and *trigger events*. The process typically involves assuring that the *source event* occurs before the *trigger event* and resolving non-one-to-one joins cases. It may also involve assigning partial credit across more than one *source event*, i.e. *multi-touch attribution*.

We propose developing a small number of predefined queries, which can be executed in an MPC between the helper servers, which cover the following use cases:

1. **Time order:** There is no way a purchase was caused by an *ad impression* that came after the purchase was made. As such, *source* and *trigger events* need, in some fashion, to carry information about the timestamp at which the event occurred.

    a. We propose that the timestamps are secret shared to ensure helper servers cannot see the original values, and that the comparison happens inside on an MPC. Using a XOR secret share and a garbled circuit is likely the best option for this type of computation.
2. **Cross-publisher attribution:** The currently proposed design should not prevent the inclusion of *source events* from multiple websites and apps. We explore this further in Trigger Queries.
    a. This is a very exciting prospect because it is much better aligned with what ad-buyers actually want from ads measurement. When multiple ad-networks all take credit for the same conversion, their reporting falls out of sync with reality from the buyer's perspective.
    b. The biggest challenge to enabling this functionality would be coordinating the *match key provider* across all publishers. We explore the idea of Multiple Match Keys below, which help solve this challenge. Further work in this area will be needed.
3. **Multi-touch attribution** is a natural extension of cross-publisher attribution, which we aim to explore. More work is necessary, both to perform optimally within an MPC and fit within the proposed privacy budget framework.

2.5 Measurement Results

2.5.1 Source Queries and Trigger Queries

We've designed the system with two types of queries, which operate in relation to section *3.3 Privacy Grain*. In that section, we propose that each site/app has a global privacy budget (per match key). First, consider the case where a site/app is only generating either *source reports* or *trigger reports*.

2.5.1.1 Source Queries

Suppose that *source.example* is a site which only generates *source reports*, and works with a number of sites, *trigger1.example, ... , trigger9.example*, which each generate *trigger reports.* Each of these trigger sites forward their (encrypted) *trigger reports* to *source.example*, along with any relevant context.

When *source.example* has enough reports to run a query, it can initiate the aggregate MPC, which will produce differentially private aggregates (e.g. sums, averages) of the *trigger values* which were attributed. If a multi-touch attribution model is adopted, this could result in an aggregate of partial values.

Note that *source.example* can use its own context and the context provided by trigger sites to group these queries into relevant sets. For example, if the *source reports* were a set of ad impressions, *source.example* could choose to run a query for a specific campaign, and only include trigger reports for items relevant to that campaign.

These queries consume privacy budget from the *source.example* budget. Note that *source.example* and *trigger1.example* could run the same query, essentially doubling their privacy budget. However, they would use their budget exclusively with one other site/app at the cost of not having any remaining budget for any other sites/apps.

2.5.1.2 Trigger Queries

Conversely, suppose that *trigger.example* is a site which only generates *trigger reports*, and works with a number of sites, *source1.example, …, source9.example,* which each generate *source reports.* Just as *trigger.example* forwards their (encrypted) *trigger reports,* each of these *source sites* forwards their (encrypted) *source reports* to *trigger.example,* along with any relevant context.

Queries operate just as the same as *source queries*, except that they consume privacy budget from *trigger.example* budget.

2.5.1.3 Networks with Source and Trigger Queries

Many *source sites* and *trigger sites* utilize 3rd parties, who will perform actions that include generating *source* and *trigger events* on behalf of the 1st party site/app. When these parties run a query, it will either have to take the form of a *source query* or a *trigger query*, and consume privacy from the 1st party *source* or *trigger site*.

For the advertising use case, *source sites* often work with many networks, which each compete to provide the greatest value in placing an advertisement on that page. This query structure would require those sites to split their budget across those networks, which could lead to adverse market effects. This is an open issue about which we hope to get feedback.

2.5.2 Advanced Aggregation

After the MPC helper servers have joined together all of the matching *source* and *trigger events* and performed attribution, there are a variety of different kinds of computation they can do. The simplest aggregation that we've explored in section 2.5.1 is summation and averages, which is easily computed in an MPC by summing additive secret shared values from the *trigger events* which were attributed to source events.

We can envision supporting more complex types of reports in the future, including Machine Learning. Linear and logistic regression seem like good candidates for future investigation as they could support many use-cases and lend themselves well to efficient computation in MPC.

2.5.3 Result Contents

The primary output in a measurement result would be an aggregation of trigger values, with differentially private noise added. We might also want to return some metadata indicating the amount of differential privacy which was added.

Certain known facts about the calculation could also be included, to validate the process. For example, the number of reports provided, the distribution from which differentially private noise was drawn (e.g. Gaussian or Laplacian) and the parameters of that distribution (e.g. mean and variance) would be shared. Depending on the threat model adopted by the aggregation, the result may also include a cryptographic proof of correctness.

3. Privacy Considerations

3.1 Differential Privacy

Ideally, this API will only produce aggregation statistics with a provable limit to the information which is revealed by queries. This includes not only limiting information in individual queries, but also across queries and considering the potential of Sybil attacks or differencing attacks.

Differential privacy is an important mechanism to enforce individual user privacy in the usage of the API. Producing differentially private aggregates typically involves adding random noise to the aggregate statistics. Fortunately, the amount of noise that must be added does not increase with the number of events in the query. This means that the larger the batch, the better the “signal-to-noise ratio”. By providing businesses with flexibility in how they structure their queries, and the periodicity in which they run those queries, we can give them more of an ability to run fewer, larger batches, thereby decreasing the relative amount of noise in the result.

There are definite downsides to differential privacy. Ad-buyers will not be happy to learn “noisy” results when they try to measure their ad campaigns. Businesses which generate few conversions experience a worse signal to noise ratio than businesses which generate larger numbers of conversions.

3.2 Privacy Budgeting

Differential privacy provides a measure of the information revealed by a specific query, however each query reveals more information. We seek to limit the information leakage for all queries issued over some fixed period of time (e.g. 7 days). We will refer to this as the *reporting window*. The exact duration of the *reporting window* is an open topic of discussion and we would like input here.

We have chosen to approach this from a “worst case” analysis - assuming entities will intentionally act adversarially to try to maximize the amount of information they can extract about individuals. We assume such adversaries are capable of setting up an unlimited number of domains / apps, and are capable of collusion across those domains / apps.

One important characteristic of IPA is that events can be generated outside of the browser. This is because the code for event generation is open-source, and because any entity can use

the *match key* of its own choosing. As such, we cannot make any assumption that *source events* or *trigger events* actually came from a browser / mobile operating system. They could have been synthetically generated on servers.

Another important characteristic of IPA is that there is no need for user-interaction to generate a *source event*. This is intentional, as conversion lift is one of the target use-cases, and this requires counting the number of conversions coming from a “holdout group” who never saw the ad in question at all.

Given these constraints, the only approach to limiting information seems to be maintaining some kind of state about how much information has been released in the given *reporting window* and filtering out any events relating to individuals for whom the privacy budget has been reached.

We think it best to give websites and apps the flexibility to specify how much of their privacy budget they would like to consume per query. Perhaps a business wants to consume 100% of their weekly privacy budget on a single query per week. They can do so with the minimal amount of noise added - but will be unable to run any more queries that week (for that set of users). Another business might want to run 5 queries per day, each day of the week. They should be able to specify that each query ought to consume 1/35th of their total privacy budget for that week. The output of these queries will have a lot more noise added (35x) than that of the first business, but perhaps the increased frequency is worth it to them, or there are a sufficiently large number of reports in their query that the aggregate will still be informative for their use-case.

3.3 Privacy Grain

We need to decide the scope over which use of events contributes to the same privacy budget, which we refer to as the “grain” of the privacy budget. Here are a few considerations we can bring to this important question:

- For utility, we need to allow independent entities to make queries without affecting each other.
    - A single privacy budget for every user might give strong privacy guarantees, but it is terrible for utility as queries from one site would then consume budget for other sites.
- At the same time, we want to reduce the number of ways that queries might be segmented as more segments could lead to more privacy loss. In particular, we don't want to allow the multiple segments to be used by the same site.

We propose, as a strawman, that the grain is (website/app, user). As outlined in section 2.5.1, when a *source query* is run, the privacy budget would be consumed from that *source site for* every *match key* in the query. Similarly, when a *trigger query* is run, the privacy budget would be consumed from that *trigger site* for every *match key* in the query. An overall budget is

evenly split between source and trigger queries, allowing each type of query to be independent.

When a query runs, both helper servers deduct the specified budget consumption from every *match key* for that website/app, and verify that the remaining budget (post consumption) is greater than or equal to zero. If more than the remaining budget is consumed, the offending reports can be removed, or the whole query can be aborted.

Such a system would require a fixed *match key*, at least over the fixed period of time specified above in section 3.2. In order to accomplish this, we propose a duplication of the protocol in section 2.4.1.1, with a slight adjustment to the blinding process. We expect to publish more details along with greater detail in section 2.4.1.1.

There are some flaws with this strawman proposal. Consider the case of an entity which operates multiple websites across which they have established a linkage of user-identity (such linkage could be established through a shared login, or through navigational tracking). If the "grain" of the privacy budget is (website/app, user), such an entity could exceed our desired privacy budget by running multiple independent queries about the same person.

This item requires further investigation. One potential extension to this approach is outlined below in Business Privacy Grain.

4. Security Considerations

4.1 Threat Model

While at present we are targeting an *honest but curious* threat model for the helper servers, we're hopeful that we may be able to develop an efficient protocol which would protect against a malicious helper server. We should assume that clients are untrusted, and that parties issuing queries may act maliciously.

Our aim is for IPA to be compatible with the Privacy Preserving Measurement (PPM) specification, and with its stated threat model. IPA's design is slightly different, in that it also includes a consumer who stores reports before aggregation. See PPM issue 166 for more details.

4.2 Attacks and Mitigations

We have identified a few attacks which would require a helper server to act maliciously and collude with a match key provider or a source or trigger website/app.

4.2.1 Known Secret Share Value

A site could collude with a helper server to identify a specific match key after the joining phase by forging an event and communicating the secret share values to the helper server. Individual secret shares should have enough entropy to uniquely identify an event.

We may be able to prevent this attack by homomorphically encrypting the secret shares, and having each party add net-zero random noise across the shares. This is inspired by the techniques used in the Private Secret Share Set Intersection protocol introduced in this paper.

4.2.2 Known Event Cardinality

A site could collude with a helper server to identify a specific match key after the joining phase by generating many events for the same match key such that the cardinality of a given match key is uniquely identifying.

Preventing such an attack likely requires the helper servers to forge extra events designed to have no impact on the final result. This is an active area of research where we would appreciate input.

4.2.3 Known Linear Relationships

The cryptographic protocol outlined in section 2.4.1.1 leverages homomorphic properties that allow the *blinding* and *homomorphic encryption/decryption* to be applied in any order (i.e. they commute.) This also preserves linear relationships.

For example, given:

```
match_key_1 + match_key_2 = A
```

will also result in:

```
double_blinded_match_key_1 + double_blinded_match_key_2 =
double_blinded_A
```

An attacker who is aware of match keys (i.e. is a provider or works with a provider), could calculate the value A (as well as many other similar relationships), and generate fake reports with a match key value equal to A. Assuming collusion with one helper server, that helper server would be able to reconstruct these linear relationships, and then learn which unblinded match keys were joined.

We are currently unaware of mitigations to this specific attack, but it is also an area of active research where we'd appreciate input.

5. Robustness Considerations

5.1 Anti-Poisoning

Our intention is to draw as much as possible from the design of PRIO here, to constrain the set *trigger values* to something reasonable and thereby prevent a malicious entity from corrupting measurement with a single malicious input.

As with PRIO, the thought is that the conversion value will be split into a *secret share* in order to hide the value from the helper servers. Valid trigger values will be constrained to some range (e.g. an 8-bit integer). A malicious entity could construct a *trigger event* with secret shares that add to -3,387,156,384 or similarly outrageous value. Including such an event in the sum would destroy all measurement. PRIO has an elegant, computationally efficient approach to determining if a secret shared value lies within a range - which does not reveal what the value itself is. We intend to utilize this same approach.

5.2 Fraud Protection

IPA *source* and *trigger reports* are generated in context, allowing the use of existing authentication anti-fraud protections. Other aggregate attribution proposals use unlinkable tokens for this type of authentication, because those reports are intended to be unlinkable to their original context.

Any new anti-fraud work designed to address the removal of tracking vectors would be complementary to this work.

6. Extensions

6.1 Multiple Match Keys

6.1.1 Attribution with Multiple *Match Keys*

Allowing reports to support multiple *match keys* from different providers could dramatically increase the match rate. Moreover, with the query structure outlined in section 2.5.1, it would be challenging for many *source sites* and *trigger sites* to all agree on the same *match key*.

If reports are able to include multiple *match keys* from different providers, this problem is solved, as each *source site* and *trigger site* can pairwise agree on at least one common *match key provider*.

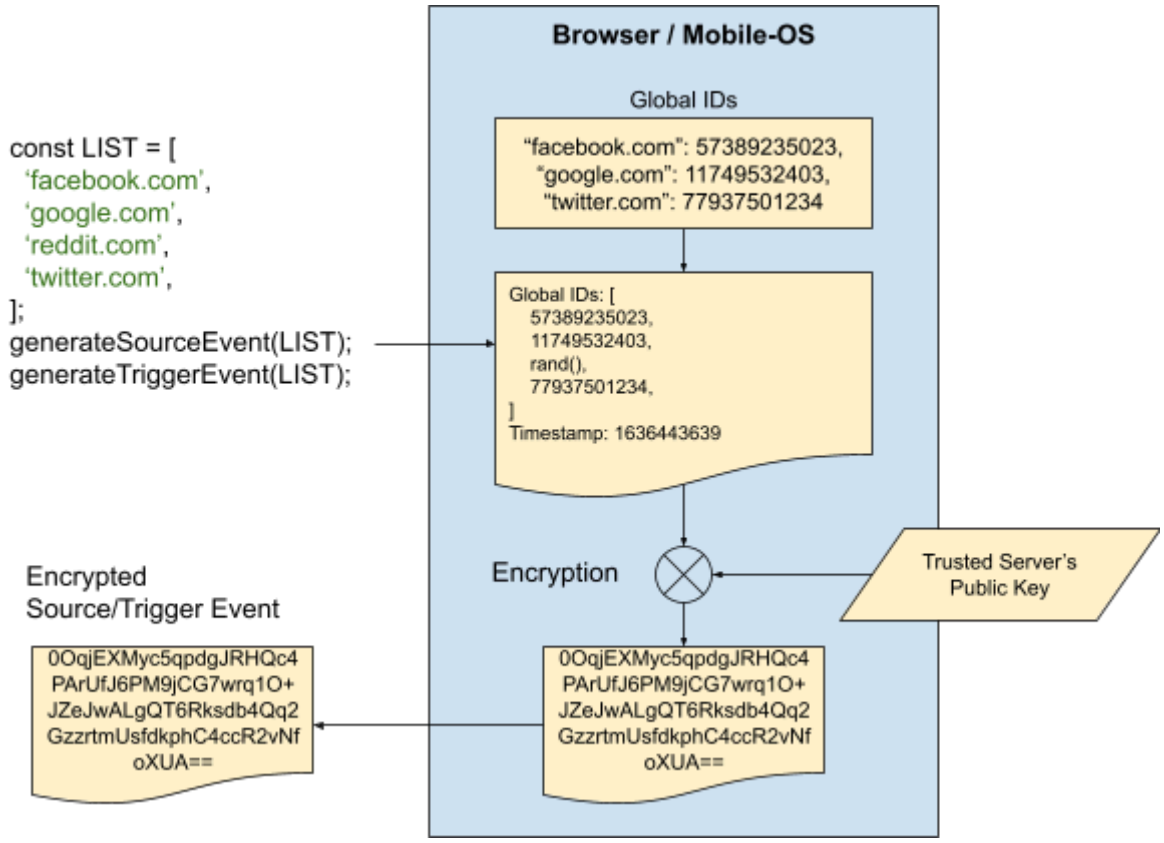
6.1.2 Privacy Considerations with Multiple *Match Keys*

If the privacy budget is scoped to the *match key*, then an entity which knows the user's identity can use multiple match keys to gain additional privacy budget. This issue must be addressed.

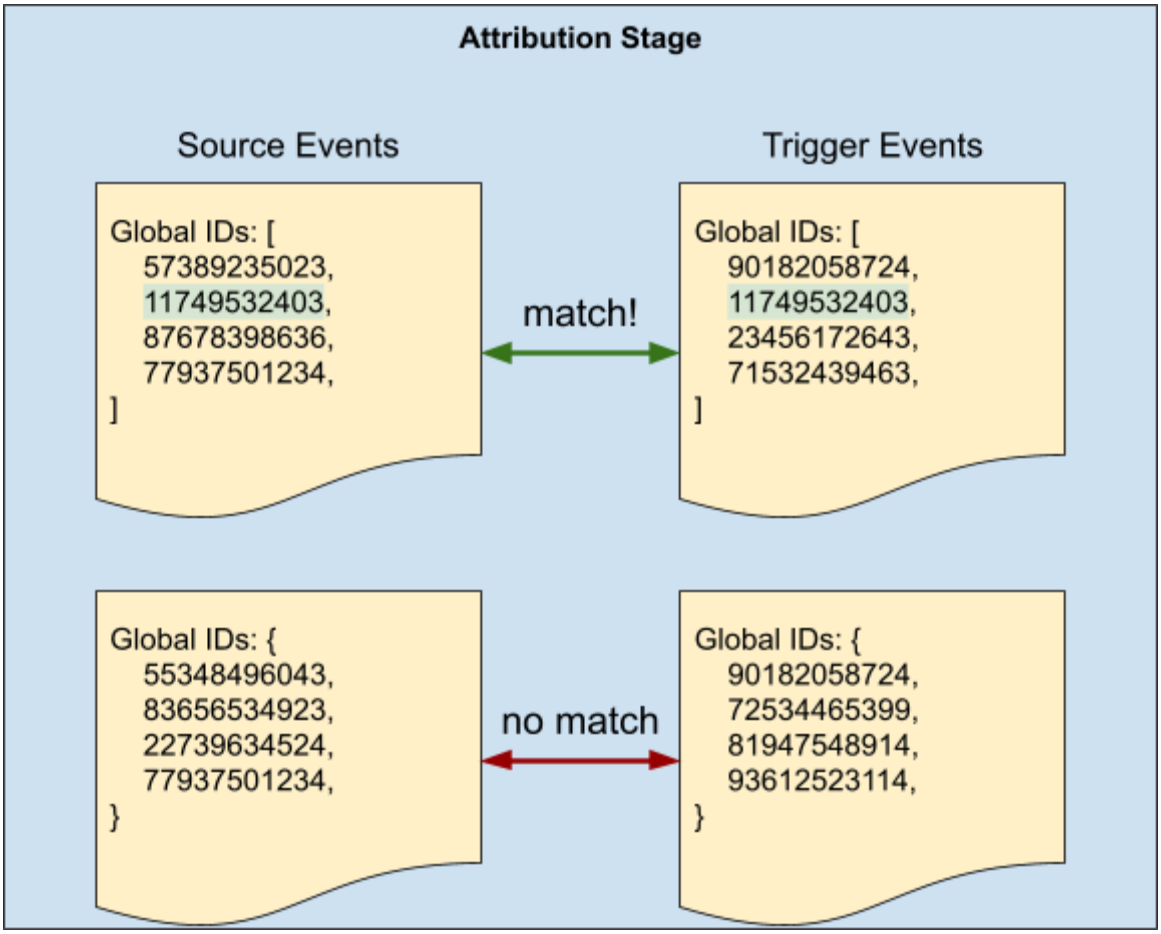
One approach is to have each site/app specify a fixed list of acceptable match keys, e.g:

- facebook.com
- google.com
- reddit.com
- twitter.com
- microsoft.com

When the browser or OS generates a *source* or *trigger event*, this list could be passed as an argument, and all available match keys could be included in the encrypted report:



All of these IDs could be blinded. At the attribution stage, source and trigger events would match if *at least one* of the listed IDs was in common between the lists.



To prevent the usage of different keys in order to exceed the privacy budget, each site/app's list of *match key* providers should be constant, or at least should not change often. For that, we'll need some sort of commitment or consistency scheme.

Even if a *source site* and *trigger site* used slightly different lists, so long as there is at least some overlap, attribution should be possible.

In order to properly account for the privacy budget, we would need to deduct from the budget associated with *every match key* that's included within the report.

6.2 Extension: Model Training and Conversion Optimization

While this proposal focuses on relatively simple aggregation of trigger values for attributed conversions, this is not the only aggregation function that might be useful. Other proposals, such as Masked LARK, explore the idea of training machine learning models within MPC.

IPA is designed to enable both attribution and post-attribution aggregation to take place in an MPC between non-colluding helper servers, and thus should extend to other types of aggregation. This is an interesting, and potentially useful, area for future research. One of the

most challenging barriers to achieving this type of aggregation is implementing machine learning processes within an MPC setting in an efficient manner.

6.3 Extension: Compatibility with Fenced Frames

Throughout this proposal, we make the assumption that the *source* and *trigger events* are returned to the *source* or *trigger site* with full context attached to that event, and that the site generating the report within the user agent has its own context about that event, such as an advertising campaign ID or an item purchased ID. These might even include globally unique IDs like impression IDs or conversion event IDs.

However, there are other proposals under consideration, notably TURTLEDOVE/FLEDGE, which prevent websites from learning which ad is displayed to the user. This design allows cross site information to be used to inform which ad is displayed without revealing that information. This would be incompatible with the existing design of IPA.

There is a potential mitigation for this. In section 2.2.1, we specify that the *source events* do not contain any payload, but there is no technical reason why this needs to be the case. In the event that a *source event* should be triggered within the context of a *fenced frame*, the fenced frame context could provide a *source payload*. This *source payload* would be secret shared to the helper servers, same as the *trigger payload.* Using a secret shared *source payload* in the aggregation function would require more research.

6.4 Extension: Business Privacy Grain

6.4.1 Defining a business

In the IPA proposal, entities will be running queries by sending batches of events to the MPC consortium for processing. This will cost money. As such, we envision a system by which entities must register, supply a payment instrument, and are provided with credentials to facilitate authenticated query execution.

At the time of registration, entities could be required to provide additional documentation, such as a business name, contact details, or even required to provide proof that they are indeed a legitimate business entity.

6.4.2 Delegation to business entities

Since many apps and websites are likely to delegate the generation of *source reports* to ad-tech companies, we would expect to see a much smaller number of “IPA registered businesses” running source queries than *source sites*.

We would also expect most ad-buyers to delegate the generation of *trigger reports*, the collection of *source reports* and issuing of queries to technology vendors. When those technology vendors issue queries, it would be in the service of specific businesses.

6.4.3 Moving the privacy grain to business

Considering *source-queries*, if we managed the privacy budget at the grain of (IPA-registered-business, user) instead of (site, user) we would address the risk of entities using multiple sites to exceed the per-user privacy budget. We would also promote more equal competition by ensuring that all ad-tech vendors have an equal privacy budget regardless of how many websites are a part of their network.

We would need to add various checks to the IPA-registration process to prevent ad-tech vendors from registering a large number of colluding entities. This seems more tractable than limiting the number of sites. Intuitively, the total number of ad-tech companies that serve ads on websites shouldn't be too large. Managing a limited number of independent privacy budgets is attractive from a privacy perspective.

The number of ad-buyers on the other hand is expected to be very large. If each ad-buyer has an independent privacy budget they can use to run **trigger queries** to measure the efficacy of their advertising, we should expect the number of IPA-registered businesses running trigger queries to be quite large. It probably makes sense to treat this type of businesses differently than those which generate source events and issue source queries.

In this design, we should also be mindful that there will likely be a smaller number of technology vendors facilitating query execution on behalf of many businesses. We will need some set of policies and heuristics to ensure that they are not artificially segmenting traffic to exceed the privacy budget.

6.4.4 Comparison with First Party Sets

On the surface, this may sound similar to first party sets, but the two are in fact quite different.

Firstly, while a particular website is limited to only be a member of a single FPS, in this proposal a single website may delegate source-event generation to multiple ad-tech vendors who compete to serve ads. As such, *source events* generated on a single site could correlate to different privacy budgets depending on the entity that generated them.

Secondly, while websites belonging to a single "First Party Set" should be recognizably linked in the minds of users, there is no such requirement here. An ad-tech company which serves ads across 100,000 websites could be managed at a single privacy budget for the *source events* it generates without needing anything in common across those sites.

Thirdly, while the “First Party Sets” proposal involves a relaxation of privacy limitations across sites belonging to a set, managing a privacy budget across many sites corresponds to an increase in privacy, not a relaxation.