Technical Comments on FLoC Privacy

Eric Rescorla
Mozilla
ekr@mozilla.com

Martin Thomson
Mozilla
mt@mozilla.com

June 10, 2021

Abstract

FLoC is a proposed replacement for "cross-site" cookies that is intended to allow for interest-based ad targeting while bounding the amount of information available to observers. We analyze the current proposal against this objective and find a number of privacy issues: When considered as replacing existing state-based cross-site tracking mechanisms such as cookies, FLoC still allows for significant linkage of user behavior, both via measuring the longitudinal evolution of cohort IDs as proposed by Wilander and in combination with existing partial identifiers such as IP address, geolocation and browser fingerprinting. In particular, it may be possible to identify individual users by using both FLoC IDs and relatively weak fingerprinting vectors. When considered as coexisting with existing state-based tracking mechanisms, FLoC has the potential to significantly increase the power of cross-site tracking. In particular, in cases where cross-site tracking is prevented by partitioned storage, the longitudinal pattern of FLoC IDs might allow an observer to re-synchronize visits by the same user across multiple sites, thus partially obviating the value of these defenses. It is unclear if it is possible to address these issues within the general structure of FLoC; we include some ideas for improvements and explore their shortcomings.

1 Introduction

Behavioral advertising on the Web is primarily enabled by "cross-site cookies", i.e., cookies [Bar11] which are associated not with the top-level site which the user is browsing but with another origin embedded on the page. This practice allows trackers to build up a profile of each user's activity, thus having severely detrimental affects on user privacy.

Google Chrome has proposed Federated Learning of Cohorts (FLoC) [Yao21] as a partial replacement for third party cookies, enabling "interest-based targeting" of users, i.e., showing ads based on general user interests.[1] Note: "federated learning" is a misnomer as the learning is not in fact federated. FLoC provides a browser API which allows advertisers to ask for a user's "interest cohort" which can then be used both to train classifiers and to target specific ads.

Chrome is currently running an "origin trial"[2] for FLoC. This means that they have enabled it for a subset of Chrome users. Sites can register to take part of the origin trial and will gain access to the API.

FLoC is presently Google-proprietary technology. For FLoC to be deployed more broadly, it will need to be built on a standard that has community and industry support and is published by a recognized standards body, like the W3C. This paper is intended as a contribution to that standardization process.

[1] As opposed to retargeting/remarketing applications in which the user is shown ads based on specific pages they have seen.

[2] https://www.chromium.org/Home/chromium-privacy/privacy-sandbox/floc

A number of concerns [Cyp21] [SE21] have been raised about FLoC privacy. In this paper, we attempt to provide an examination of its privacy properties with a modest degree of rigor. We also explore some options that might improve FLoC privacy. The remainder of this paper is structured as follows: In Section 2 we provide an overview of FLoC. In Sections 3 and 4 we lay out the threat model and use it to evaluate FLoC's privacy. In Section 5 we examine Google's proposed privacy mitigations and in Section 6 we provide some ideas for potential improvement.

2 Overview of FLoC

Probably the easiest way to think of FLoC is as partial replacement for cross site cookies. Effectively a cross site cookie is a lookup key for the browsing history of a specific browser, collected by a given tracker from its vantage point on various sites. When the tracker wants to target an ad, it can then use the cookie to look up that browsing history and target based on that history.

FLoC is implemented as a browser API. Browsers provide sites that use the proposed `interestCohort()` function with a FLoC cohort identifier, or FLoC ID, which is just an integer. A FLoC ID is effectively a unique identifier for a set of users with somewhat common interests (computed as described below). A tracker can group together all of the visits it sees associated with a given FLoC ID. Then, when it wants to target an ad, it can use the FLoC ID to look up the history of that group and target based on that history. The key insight is that the FLoC ID can be treated as an identifier for a synthetic user with the browsing history that's the union of the histories of everyone in the cohort. Obviously, if the cohorts are randomly assigned, this is not useful to advertisers, but to the extent to which cohort members are similar, then it will be useful for targeting.

FLoC assumes that cohort identifiers will be observed on multiple sites. As cohort identifiers do not carry any inherent information about cohort interests, it is up to observers to infer interests for each cohort based on what content users in that cohort visit. It is expected that observers will gather information about the content that different cohorts view and then make inferences about the interests of those cohorts, using the tools used today for tracking individual interests. The idea is that tracking systems need minimal changes to adapt to FLoC targeting.

The description in the following sections is based heavily on the Google FLoC origin trial[3] description.

2.1 Cohort Construction

The cohorts are constructed using a technique called PrefixLSH, which is similar to SimHash[4] The details can be found on the Chromium Projects site [5]. At a high-level, PrefixLSH seeks to generate a b-bit hash for each user which digests that user's entire browsing history for the past 7 days (this is a tunable parameter).

For a given site in the user's browsing history, the hash of the site's domain is used to seed a pseudo-random number generator (PRNG) that draws from a Gaussian distribution. A b-dimensional floating point vector is created from the PRNG by drawing b numbers. This process is repeated for each domain in the user's browsing history, resulting in N b-dimensional floating point vectors. Then, these N vectors are summed together, and the b-bit hash is generated by taking the *sign* of the result; i.e., the ith value of the b-bit hash is the sign of the ith item of the sum of the individual domain vectors.

[3] `https://www.chromium.org/Home/chromium-privacy/privacy-sandbox/floc`
[4] `https://en.wikipedia.org/wiki/Locality-sensitive_hashing#Random_projection`
[5] `https://www.chromium.org/Home/chromium-privacy/privacy-sandbox/floc`

2.2 K-Anonymity

FLoC aims to provide individual users with k-anonymity. That is, an individual user cannot be identified as a result of the information FLoC releases, except by identifying a group of users of a size at least k users. However, the basic algorithm described in the previous section does not guarantee a minimum cohort size, as they will be distributed around the average number of users per cohort. In order to address this, Google starts with a 1 bit length vector and treats it as a binary tree, continuing to extend the tree as long as each subtree has at least 2000 users, up to a current maximum length of 20 bits. This computation is performed server side based on users who are syncing their history data and logging cohort calculations. Because there are users who are not qualified for the trial and so this is the minimum size of the set. The list of prefixes is then distributed to all Chrome users to match their computed cohort ID against.

Note that FLoC IDs are not the only identifying information available to observers: when combined with other forms of information such as browser fingerprinting, the k-anonymity guarantees become much weaker, as observed by Don Marti and others. This topic is discussed further in Section 4.4.

2.3 Sensitive Categories

Even if cohorts are guaranteed to be k-anonymous, that does not mean that they are not leaking sensitive information. Consider, for instance, the impact of a cohort which has high interest in some illegal or embarrassing activity. Users might not wish sites to know that they had such an interest, and additionally there may be risks to the user in allowing sites to target messages at those categories. Google proposes using two mechanisms in tandem to manage these risks:

- Cohorts which are too closely correlated[6] with "sensitive" categories are removed.
- Sites which fall into sensitive categories aren't included in the FLoC calculation.

The first method sets a maximum threshold of t-closeness[7] between the overall distribution of user visits to websites in a sensitive category and the distribution of visits to that category within any given cohort. The intuition is that if 20% of Chrome users generally visit adult sites and cohort X has 30% of users visiting adult sites, then the cohort is correlated with an interest in adult sites and should be considered sensitive. For each cohort, the t-closeness is calculated for every sensitive category, and if a cohort exceeds a threshold difference in closeness for any category it is marked as sensitive. The FLoC API returns an empty string when a cohort is marked as sensitive.

Neither of these mechanisms seems entirely adequate; see Section 5.2 for more details.

2.4 Cohort Changes

Because user interests change over time, the history that FLoC considers is limited to a fixed period of 7 days. Browsing history over that period is input to cohort construction (Section 2.1). Once a cohort identifier is selected, browsers will not change the value that they advertise for a similar period.

[6] https://docs.google.com/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDo1Mzg4MjYzOWI2MzU2NDgw

[7] https://en.wikipedia.org/wiki/T-closeness

3 Threat Model

Establishing the threat model for FLoC is more difficult than usual because FLoC is explicitly designed to leak information about the user's browsing history; therefore any security objectives based on indistinguishability or leaking no information are immediately impossible. Instead, we need to frame objectives around bounding the degree of leakage. As an approximation, we should consider two main types of observer:

- *Direct.* What do observers learn about user behavior on sites on which they are embedded, both their own sites and other sites on which their content is embedded.
- *Indirect.* What do observers learn about user behavior on sites on which they are not embedded?

Roughly speaking, the first question compares FLoC to existing cookie-based trackers. Ideally, FLoC should be no more powerful than those trackers because otherwise it would actually make the privacy situation worse than the status quo.

The second question is more complicated, as it compares FLoC to two reference points: (1) existing websites, which learn basically nothing about your browsing behavior outside their own site, and (2) trackers, which use some non-cookie identifier such as a supercookie or a login ID. Ideally, we would like FLoC to be no more powerful than (1) and not to enable (2). However, because FLoC gathers information from across your entire browsing history, the current design does not achieve the first and may not achieve the second.

In addition, we need to consider observers independently and those who cooperate. Two observers A and B who cooperate can be modeled as a single observer AB, but this is an upper bound on their capabilities, as it may not always be possible to synchronize between A and B and, if they are competitors, they may choose not to cooperate. Cooperation is already a common feature of the ads ecosystem and occurs both via the browser using techniques like cookie syncing and directly between observers using persistent identifiers such as e-mail or IP addresses.

Clearly, FLoC increases the amount of information available to observers over not having FLoC, so just adding FLoC to existing browsers without changing their cookie policies is trivially worse. For the purposes of this document, we assume that the browsers will adopt a more restrictive cookie policy than their current policies, though it is an open question whether they will really do so. Thus, we need to also consider the effect of FLoC in combination with each storage policy.

4 Data Leakage Analysis

This section provides an analysis of the leakage properties of FLoC. We note at the outset that we are not the first to look at FLoC and so a number of these issues are already known. We have attempted to give credit where we know that unobvious points were previously made by others while avoiding crediting obvious observations that have been made by many. We welcome reports of instances where we have inadvertantly failed to give credit.

4.1 Preliminaries

Denote the set of all users as U and the ith user as u_i. The label i not visible to sites and must be indirectly inferred, but there are cases where some other value which is a function of i will be available, such as $email(i)$.

We use the nomenclature of O'Connor [Tes20] to define "site" and "origin" and related terms.

Denote the set of all observers (i.e., entities who serve web resources) as O and the jth observer as o_j. The set of sites S and the kth site is s_k. Iff $j = k$ then observer o_j is same site to site s_k. If content provided by observer o_j is embedded on site s_k then $Embedded(j,k)$ is **True**, otherwise it is **False**. We don't consider different types of embedding, such as image, script, or frames, differently, under the assumption that the observer is able to rely on the same capabilities across all embeds.

Denote a visit of u_i to site s_k as $v_{i,k}$. In the current world, observer j can see visit $v_{i,k}$ iff either it is same site ($j = k$) or, if a cross site observer ($j \neq k$) is embedded ($Embedded(j,k)$). Given two visits that it can observe $v_{i,k}$ and $v_{x,y}$, observer j can determine if they are the same user ($i = x$) iff it is able to store a cookie (or similar state) that is visible across both visits.

In the current Web, there are three basic models for state access:

- *Unrestricted cross site storage access.* In the original permissive cookie policy (e.g., the default policy in Chrome), the operators of both sites and embedded content can determine if $i = x$.
- *Cross site storage access blocked.* If cross site storage access is blocked, then only top-level sites ($j = k$) will be able to determine if $i = x$. Firefox and Edge currently implement tracking protection mechanisms in which some sites believed to be trackers are not able to access their storage in a cross site context. Cross site trackers will only learn that they have been identified as such.
- *Partitioned cross site storage.* Users agents may *partition* storage so that cross origin content accesses a store that is specific to the tuple of top-level site (s_k) and observer (o_j), that is (s_k, o_j). If storage partitioning is in use, then $v_{i,k}$ and $v_{x,k}$ can be identified as being for the same user, but all observers O will have two different stores for $v_{i,k}$ and $v_{x,y}$ if $k \neq y$. This is the case in Safari, where all cross site cookies are blocked and other storage APIs are partitioned ephemerally for the session. Other browser vendors are experimenting with combinations of these policies.

In addition, some browsers remove cookies and other site-visible state under different conditions. This can be a policy decision or because the users deliberately delete history.

We denote the set of all FLoC cohorts at a given time as C. u_i is assigned a FLoC cohort identifier $c_i \leftarrow Id(i) \in C$ with $Id()$ being the FLoC algorithm defined in Section 2.1. Note that this notation does not capture the fact that cohort allocation changes over time.

4.2 Direct Observers

As described above, with direct observers we are interested only in what they can learn from the visits they themselves see.

4.2.1 Unrestricted Storage Access

By comparison to a simple unrestricted storage access policy, FLoC clearly leaks less information: the observer goes from being able to certainly distinguish visits between users u_i and u_x at all times to only when $c_i \neq c_x$. This is a strict improvement as long as FLoC cohorts are reasonably large.

If storage is cleared or expires, however, then the observer learns more: given two visits to the same site on opposite sides of the storage clearing event $v_{i,k}$ and $v_{x,k}$, an observer that does not receive c_i after the event does not learn that these visits are from the same entity. However, with FLoC the situation is different unless FLoC IDs are reset when storage is reset, for instance because the user deletes their cookies.

- If $c_i \neq c_x$ then the users are different. This is a modest leak of information because most users are different. At best it eliminates users from $|c_i|$ from consideration — a reduction in the anonymity set by a factor of $\frac{|U|}{|U|-|c_i|}$.
- If $c_i = c_x$ then the probability that the users are the same is naively the inverse of the size of this FLoC cohort i.e., $\frac{1}{|c_i|}$. This is a sizable informational improvement, reducing the anonymity set by a factor of $\frac{|U|}{|c_i|}$.

Not exposing a FLoC ID after storage is cleared, at least until enough history is accumulated, might be necessary. However, as browsing habits are expected to be relatively stable, it might be that users end up being assigned the same identifier at the end of that period.

It is not entirely clear how to manage this information leakage: simply recomputing the FLoC cohorts does not help that much because they are intended to be somewhat stable. One could simply clear the history over which the cohorts are calculated, but then how do we handle cases in which the storage is only deleted for a single site? Another alternative would be merely to remove the site for which cookies were erased, but it's not clear that that has the desired impact, as the cohort may not change from just removing one site.

When added to a client with an unrestricted storage access policy, FLoC makes the situation modestly worse in two ways. First, FLoC identifiers — especially in combination with fingerprinting information (see Section 4.4) — might be used to restore cookie state after it has been cleared on the client. Second, it may make cookie syncing[8] easier because it can match cookies in a way that is invisible to the browser (by contrast to existing syncing techniques which rely on Web mechanisms).

4.2.2 Cross Site Storage Access Blocked

When cross site storage access is blocked, either in whole or in part, FLoC clearly leaks more information to any observer j. For observers who are able to access storage, the situation is the same as described in the previous section.

In the simplest setting, assume that time is divided up into a set of periods: at the beginning of period, each browser computes its FLoC ID based on the current browsing history and uses that ID for the rest of the period. If we assume that the observer knows the duration of each period, then the situation is the same as with storage clears. I.e., within a given period:

- If $c_i \neq c_x$ then the users are different.
- If $c_i = c_x$ then there is a $\frac{1}{|c_i|}$ chance that the users are the same.

Outside the period, the situation is more complicated, but some information still leaks. Although cohorts are likely to shift *some* between periods, this is not an entirely random process and each browser instance is more likely to move to a nearby cohort than to a far away one.

That small changes in history result in small changes in cohort allocation is implicit in the assumption that the FLoC ID carries information. The similarity of cohorts is also inherent in the assignment algorithm. The Hamming distance between identifiers is a reasonable proxy for the similarity of cohorts. Recall that assignments are the result of summing contributions from multiple items of browsing history. A small change in browsing history — that is, a change that results in a small number of additions or removals from the browsing history — is unlikely to result in large changes to the total.

[8] `https://headerbidding.co/cookie-syncing/`

Thus, if two visits have very different cohorts, it is likely they are from the different users (see Section 4.4 for more applications of this). Conversely, if they have nearby cohorts, then it is more likely they are from the same user.

The situation is obviously more complicated in situations where there are no fixed periods in that two successive visits by the same browser may or may not cross a period boundary.

4.2.3 Partitioned Storage

In partitioned storage scenario, cross origin content accesses a store that is specific to the tuple of top-level site (s_k) and observer (o_j), that is (s_k, o_j). As compared to this case, as with the previous two scenarios. FLoC leaks more information to that observer, because the FLoC ID contains information about other sites than the one presently being visited. This is unsurprising, as it is part of the purpose of FLoC to allow targeting on this kind of information.

More seriously, FLoC *in combination with* a partitioning policy has significantly worse privacy properties than partitioning alone, with a result that is not that different from the client implementing an unrestricted storgage policy as described in Section 4.2.1.

Because partitioned storage separates user visits spatially rather than temporally, visits by the same user to different sites cannot be correlated using site storage. However, visits from u_i will have the same FLoC ID, which can be used to partially reconnect them. As with the case "in-period" case in the previous setting, inequality of IDs indicates different users and equality of IDs indicating an increased probability of the same user.

Moreover, because partitioning still allows observers to link visits across time, it now becomes possible to link visits across multiple sites where the observer is embedded. Consider the case where we have users 1, 2, 3, and 4 with FLoC IDs as shown in Figure 1:

Period	User 1	User 2	User 3	User 4
1	A	A	B	B
2	A	B	A	B

Figure 1: FLoC IDs example

During each time period, there are two users with each FLOC ID, so it is not possible to identify any individual user by FLoC ID alone. However, the pattern of IDs is unique for each user. Thus, an observer who is able to link up visits during each time period obtains a unique identifier (e.g., $u_1 = [A, B]$) that it can then use to link up visits on any site on which it is embedded, effectively breaking the isolation boundary provided by storage partitioning. Note: this is a variant of an attack proposed by John Wilander on visits to multiple sites[9]. This precise attack is not possible with tracker blocking because the trackers are not able to link up multiple visits even on the same first party origins. However, in many cases trackers are actually JavaScript embedded in the top-level context of the site, which allows them to store an identifier against the storage for the site (e.g., cookies, IndexedDB, etc) and mount a similar attack.

4.3 Indirect Observers

By an *indirect* observer, we mean one who is embedded on some set of sites and wishes to learn about a user's behavior on *other* sites. Because the IDs themselves are meaningless, just having a FLoC ID is not

[9] `https://github.com/WICG/floc/issues/100`

useful. However, because FLoC IDs are stable across sites, it is straightforward to map an ID onto other information of interest.

The most obvious avenue here is that an observer embedded on many sites can publish the information they learned from those observations. For instance, they might publish the URLs associated with each FLoC or merely a query interface such as "What is the probability that FLoC ID x likes cars?". Note that in order to make use of such an interface, sites need not embed any trackers at all: they merely need to call the FLoC API.

Another avenue is for multiple indirect observers to collaborate to build a joint browsing history index using the FLoC ID as the lookup key. I.e., each site calls the FLoC API to get the ID and then shares all their visits with the other members of the group. These can also be linked with per-site user identifiers as proposed by John Wilander to help produce more dense profiles. Note that even if the time series of FLoC IDs is not unique, it still leaks extra information: for instance, if FLoC IDs A and C are both associated with interest in cars, a user with the series of IDs $[A,C]$ is probably more interested in cars than one with $[B,C]$.

The general pattern here is that FLoC allows sites to learn about — and share — user interests without embedding any code from trackers or visibly participating in any tracking network. All that the client's browser sees is that the observer called the `interestCohort()` API and it has no way of knowing what the site does with that data. As with other APIs which are usable for fingerprinting (e.g., WebRTC), the browsers then have the problem of distinguishing "legitimate" from "illegitimate" uses of the API (even assuming that we could draw a conceptual bright line between those uses). Naively, one might imagine that any site that asked for the FLoC ID is using it for tracking, and that therefore browsers could opt to block it, but that's not clearly true: for instance, a news site might use your interest cohort to select which news articles to promote to you.

4.4 Fingerprinting

Fingerprinting of clients — the practice of using secondary, but stable, traits of browsers and their operating system to better isolate individuals or groups — can be combined with FLoC to produce smaller cohorts.

The combination of a cohort ID and fingerprinting information might be able to isolate some individuals into groups that are much smaller than the target cohort size, $|C|$, as observed by Don Marti. In practice, user requests supply information to servers that can be used for fingerprinting, including: machine type, browser type and version, language, approximate location.

There is active debate about the actual amount of fingerprintable information that browsers supply [Eck10] [LRB16] [GBLB18]. Even the most conservative estimates indicate in excess of 10 bits from the user's device properties. Ten bits might not be individually identifying, but we need to consider the segregation this provides when combined with FLoC.

It is possible that some fingerprinting inputs are strongly correlated with FLoC cohorts. For instance, people with the same interests might also tend to buy the same computers, use the same browsers, and apply the same customizations. A study would be needed to understand what sort of correlation exists and how stable that correlation is as browsing activity changes or new seeds are used in the algorithm.

It might never be possible to remove certain attributes that are used for fingerprinting. Users will choose the computers they buy and the browsers (and browser engines) they run. Some characteristics, like CPU performance, might always result in differences that can be observed by sites. Some information, like IP addresses [MLV+20], is fairly unique, stable, and difficult to remove[10]. Where eliminating differences might

[10]The best available techniques for removing IP addresses involve proxies or VPNs, which incur additional cost. Lassey [Las20b] has proposed a proxying technique called "near-path NAT" that is intended to reduce the cost, but to our knowledge there are no extant deployments and so operational feasibility is unknown.

be possible in some cases, the effect on functionality could be difficult to justify in others.

Take the example of screen pixel density: it would be possible for every browser to report the same pixel density, but the cost might be that either that users with high pixel density screens receive low quality images or users with low pixel density screens download larger high pixel density images that they do not benefit from but use more time, bandwidth, and processing power. Given the low number of pixel density options, the high correlation with other hardware choices, and the high value of exposing the information, pixel density is a fingerprinting input that seems likely to remain.

As it is likely that FLoC will be deployed in a world where fingerprinting continues to be possible, its privacy guarantees need to be considered in that context. Given that the amount of information provided by fingerprinting appears to be roughly similar to the size of FLoC cohorts (i.e., $entropy(FLoC) + entropy(fingerprinting) \approx log_2(total population)$) this is concerning. Google has proposed "privacy budget" [Las20a]. As part of this, they aim to provide browsers more direct control over how fingerprinting information is exposed to sites. Once controls are in place, the amount of fingerprinting information that sites can access can be controlled. Features can be blocked or put behind user permission prompts if too much fingerprinting information has already been exposed to a site. We do not presently have enough information on the details of the budget calculation to evaluate its effectiveness. Even if we assume that this bound can be enforced, the large volume of passive fingerprinting information and the frequency of fingerprinting in the existing Web makes it unclear that it will be possible to deploy FLoC as a drop-in mechanism.

4.5 Navigation

The final consideration for matching user identity across sites involves the information that is exchanged when a user navigates from one site to another. In this case, the originating site is able convey information to the destination site, both through the URL that is the target of the navigation as well as the HTTP `Referer` (sic) header field.

It is generally considered infeasible to prevent sites from using navigation to share information. Sites that cooperate can use navigation to pass information they have gathered, including FLoC IDs.

The use of automatic redirects is a technique that is used widely. Redirects are used in OAuth [Har12] to transfer authorization information between sites. Automated redirects are visible and can degrade performance, but they are an effective means of distributing information between sites.

With this capability, sites are able to pass information. As with fingerprinting, we cannot ignore the effect of this means of transfer of correlating information.

4.6 Strong Identifiers

There are some identifiers that provide much stronger correlation between visits than fingerprints. In particular, email addresses are now highly sought after by publishers and advertisers specifically for this reason. Strong identifiers like email addresses or phone numbers allow trackers to build profiles across those sites where the identifiers are used.

With FLoC, individual profiles are a potential source of additional information about the properties of the FLoC as a whole. For instance, information from individual profiles can be generalized to inform decisions about the FLoC cohort as a whole. The effect on the privacy of FLoC in the presence of strong identifiers for some users needs to be considered.

5 Impact of Proposed Privacy Protections

Google has proposed two sets of mechanisms to limit the leakage of sensitive information. First, they require a minimum cohort size, as described in Section 2.2. Second, they attempt to limit the exposure of sensitive information via FLoC as described in Section 2.3. This section examines those protections.

5.1 K-Anonymity

Setting a minimum size of cohorts is a necessary privacy protection, but has two important limitations that undercut the claim that it provides actual k-anonymity.

First, a cohort of size *k* only provides *k-anonymity* when there are no other indicia of user identity and no fingerprinting (see Section 4.4). Google's initial origin trial[11] uses cohorts of minimum size of 2000 (~ 11 bits); in the worst case where the existing fingerprints and FLoC are independently distributed, then 11 bits of fingerprinting entropy would be sufficient to identify individual users. Moreover, as noted by Wilander and in Section 4.2.3, multiple FLoC cohort samples can also be used to reduce the anonymity threshold further.

Another side effect of this is that the privacy/targeting quality tradeoff is worse for less popular browsers and platforms because each client represents its own cohort: thus a 2000-person cohort on Chrome represents a more cohesive set of users than one on Firefox.

Second, it may be possible for attackers to further reduce the anonymity threshold by mounting a Sybil attack. As noted above, Google's method for ensuring minimum cohort size is to count the cohorts assigned by users and then to continue subdividing until they reach the cohort threshold. An attacker can artificially inflate the size of a cohort by submitting extra reports for that cohort, thus forcing the centralized cohort assignment algorithm to extend the prefix for that cohort. In principal, it might be possible to extend a given prefix all the way to the maximum of 20 bits. As a prefix is extended, the cohort captures more specific information about a user. Sybil attacks are notoriously hard to prevent, and it is not clear how to do so here while also protecting user privacy.[12]

Finally, the mechanism which Google is using to compute size of the cohorts relies on having server-side measurements of the size of cohorts computed using Chrome Sync data. We note that server-side cohort sizing is not possible with browsers that encrypt history data before syncing it, as Firefox does. This in principle could be replaced with a private data collection such as the heavy hitters mechanism of Boneh et al [BBCG$^+$21].

5.2 Sensitive Categories

Although it seems helpful to factor out cohorts which are highly correlated to sensitive attributes, it's not at all clear it will be sufficient.

First, the definition of "sensitive" is problematic. We know that sensitive characteristics manifest in ways that are not obvious even to the person who is leaking that information[13]. In their current origin trial Chrome is using the sensitive interest categories that Google uses for targeted ads[14], but this clearly is not

[11] `https://www.chromium.org/Home/chromium-privacy/privacy-sandbox/floc`

[12] One possibility would be to use Google's user sync data to compute the cohort sizes (thus pulling in whatever anti-Sybil mechanisms Google already has for their accounts) but of course this relies on users sending their browsing history to Google in the clear and is not applicable to other browsers, like Firefox, which encrypt history data.

[13] `https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/`

[14] `https://support.google.com/adspolicy/answer/143465?hl=en`

a sufficient list of information people might not want revealed. As a concrete example, Google prohibits targeting based on "divorce", but a married person who is looking for extended stay hotels might well be considering a relationship change and would not want their spouse to be aware of that. In general, the idea that there can be a global list of all the categories that are sensitive seems extremely problematic because each individual has their own view of what parts of their information are sensitive.

Second, since t-closeness is calculated per-category, cohorts may still have affiliations with subcategories of sensitive categories. For instance, suppose that 20% of the population visits health-related websites. Cohort A may have 20% of users visiting a cancer-specific website and very few users visiting other health-related topics. This passes the t-closeness test but still correlates to a sensitive user attribute if $<<$ 20% of users visit cancer-specific sites. Breaking down a "health-related" topic into more granular categories might help, but more categories makes the assessment more challenging and risks marking more cohorts as sensitive.

Finally, FLoC's current response to determining that the user is in a sensitive cohort is to return the empty string. However, this signals that the user has some sensitive characteristic. For instance, it might be advantageous for a health insurance company to know that a user is in a sensitive category. Of course they won't know which sensitive category, but there's a much higher than normal chance that the user has a pre-existing medical condition. The risk here depends on the other conditions in which FLoC returns an empty string (i.e., if it also does this in the browser's private / incognito mode).

Google also proposes to exclude sites which are sensitive from the FLoC calculation. However, there are also sites which correlate to sensitive properties but are not themselves sensitive, which would not be excluded.

More generally, all of these countermeasures rely on the ability of the browser manufacturer to determine which FLoC inputs and outputs are sensitive, which itself depends on their ability to analyze user browsing history as revealed by FLoC. This creates inherent gaps in the protection provided by these countermeasures and an arms race between the advertisers and the browser makers. It is also unclear the browser makers are right entities to determine what is "sensitive", but there is also no obvious alternative answer, either in terms of ability or governance.

6 Potential Improvements

As discussed in the previous section, FLoC in its current form has a number of undesirable privacy properties, both when used alone and in combination with existing anti-tracking technologies. This section suggests some potential avenues for improvement. These are incompletely worked out, so may either not work, lead to negative impacts on targeting, or both.

We attempt to identify the risks and shortcomings of these ideas. We have included both ideas that are likely infeasible as a result of these shortcomings and ideas that might have potential value with additional work.

We have two goals here: First, to achieve the stated goal of FLoC: to provide an identifier for users that provides advertisers with some information about a user's browsing history, but to retain k-anonymity. That is, even if FLoC IDs are shared by users, advertisers are unable to isolate groups of users that are smaller than some target size. The second goal is that FLoC does not degrade privacy when deployed to the Web, under the three broad storage regimes described in Section 4.1. That is, this would not make privacy worse if it were deployed to the Web today. Furthermore, our goal is that k-anonymity be possible in a future where site storage is restricted in some way (Sections 4.2.3 and 4.2.2), but where some browser fingerprinting remains

possible.[15]

6.1 Per-Domain FLoC IDs

As noted in Section 4.3, one of the major respects in which FLoC is a regression is that it allows observers to infer user behavior even from sites on which they are not embedded. For instance, `example.com` can capture a user's FLoC ID and then use a service which maps IDs to preferences. This is much easier with FLoCs because they are consistent rather than per-domain.

One potential countermeasure is to create new FLoC IDs for every observer domain (i.e., run the partitioning algorithm with a seed based on either the observer origin or registerable domain). An observer that uses `observer1.example` would see a different FLoC ID from `observer2.example` even when they are both embedded on `example.com`. However, it would see the same FLoC ID when embedded on `example.org` as on `example.com`. One potential approach to implementing this would be to have a per-observer seed for PrefixLSH rather than the same seed for all observers. This would prevent the creation of a single ID→interest mapping. Potential issues here include:

- Cooperation between parties embedded on the same site. Similar to cookie synchronization [Goo21], two third parties embedded on the same site could share their FLoC IDs directly with each other. Each party could retrieve their respective FLoC ID within an iframe and postMessage the IDs between frames. Additionally the site itself could instantiate a large number of apparent (but actually coordinated) third parties and use that to either (1) create a supercookie for the user or (2) extract far more information about the user by collecting multiple independent FLoCs that each leak some of the user's interests.

- How closely do the cohorts for origin A match those for origin B? If they have the same members but different IDs, then it's easy to create a mapping table between cohorts.

- Embedded JavaScript has the origin of the site on which it was loaded, not the origin of the JavaScript, so the API calls would ordinarily return the identity of the site. Potential approaches here would be (1) tracking where the JS was loaded from (2) putting the FLoC ID in a header (potentially using Client Hints) or (3) only allowing FLoC in Fenced Frames or the like (see Section 6.4).

- A site and any embedded observers can cooperate to look up and use the FLoC ID for the observer's site. This isn't necessarily fatal, as some of this behavior is visible to the browser, but it isn't ideal. Here too, some information flow control might help.

6.2 Per-site FLoC IDs

A second potential countermeasure is to partition the FLoC ID by the first-party site. Unlike the countermeasure introduced in Section 6.1, this approach would provide the same ID to all callers irrespective of their origin when they are loaded on the same site, and would provide a different ID to the same caller across sites. Conceptually, this approach is similar to storage partitioning (Section 4.2.3) — noting that the temporal attacks discussed in that section will also apply here. There are a few ways in per-site FLoC IDs could be implemented, all of which have vulnerabilities.

First, one could do a per-site shuffle of identifiers (i.e., apply a pseudo-random permutation that is keyed by top-level registerable domain), then the shuffle needs to be consistent between users or they get different

[15] While we allow for some fingerprinting, this requires that we assume that fingerprinting using IP addresses is somehow not possible.

identifiers. This results in a known transform from one low entropy domain to another domain of the same size, which is trivially reversible.

Alternatively, one could do per-site repartitioning (i.e., run the partitioning algorithm with a seed based on the top-level registerable domain). This produces cohort identifiers $c_{i,k}$ that are dependent both on the history of a user i and the visited site k. Per-site cohorts prevent the trivial mapping discussed above, but introduces another problem. If the tracker is able to join partitioned FLoC IDs across websites, the information revealed by each of these IDs may be non-overlapping and thus add to the information the tracker has about the user. While browsers seek to restrict the flow of information between sites, eliminating all avenues of information transfer is a monumental task, and FLoC should be robust against this possibility.

The information leaked potentially scales very badly: if a tracker is able to correlate activity across sites, then they learn about which subset of users in each cohort have different cohort identifiers for different sites. This sort of correlation might be performed by annotating navigations between sites (see Section 4.5).

Whether this is bad depends on how tightly users in each cohort are clustered:

- A tightly clustered cohort might not result in changes as a result of using a different seed. If this is common, the effect is similar to a simple reshuffle of the identifiers of affected cohorts.
- If user activity is uniformly distributed, each new seed might result in the users of any cohort under one seed being split in half when a new seed is used. In that case, given a cohort size $|C|$, correlating activity across $\upsilon \leftarrow \lceil log(|C|) \rceil + 1$ sites could be sufficient for individual targeting of nearly all users in the cohort. That is, $|\bigcap_{k=0}^{\upsilon} c_{i,k}| \rightarrow 1$.
- As the distribution of each division is unlikely to be perfectly uniform, there is likely to be a mix, with some users who are not separated from each other and a small subset of users — those with slightly different browsing history — who are split into a different cohort a new seed. The privacy is much worse for the few who do move relative to their neighbors. In the extreme, being able to correlate user activity across two sites could reveal that just one user moves from cohort $c_{i,k}$ to cohort $c_{i,l}$. If the site can confirm that these cohort identifiers belong to the same user, that would result in that user being uniquely identifiable. That is $|c_{i,k} \cap c_{i,l}| = 1$.

Even ignoring these issues, it's unclear how useful cohorts are when partitioned by site. Tracking origins will see a different partitioning for every site, meaning the cohorts can only be used to optimize ad selection per-site. This likely will be sufficient on websites that have a large number of users, but it's unclear how far down the long tail of websites that will hold.

6.3 ID Truncation

Because FLoC IDs are sent by every client in the FLoC, observers get a far larger training set than they would with cookies. For instance, if there are 2000 members in a given cohort, then the observer gets 2000 users worth of history, not just one. If we were just interested in training, this suggests that we could downsample, with users only sending their FLoC IDs with low probability. For instance, if users only sent their FLoC IDs 1% of the time, the observer would still have far more training data than with cookies, but far less data would leak. However, because FLoC IDs are also used for targeting and ads are shown far more than 1% of the time, on its own this technique won't work.

This suggests a hybrid approach. Whenever the FLoC API is called, the browser usually returns a truncated FLoC ID but occasionally returns the full ID. To implement this, the browser value between 0 and 1 based on $PRF(K, origin)$ where K is a secret per-browser key. The browser then returns the full ID with probability $thresh$ and an ID truncated to b bits with probability $1 - thresh$, as shown below.

$$r = PRF(K, origin)$$
$$Id = \begin{cases} Id & r \leq thresh \\ [Trunc(Id, b), |Id|] & r > thresh \end{cases}$$

The result is that any origin consistently receives a truncated or full ID for each client. If the ID is truncated, it learns the length of the full ID as well. For training purposes, observers can simply ignore truncated FLoC IDs and train solely on the full IDs. As noted before, they will still have more training data than with cookies.

For targeting, the situation is somewhat more complicated. If the observer gets the full ID, they can just target as usual. If they get a truncated ID, however, then they have limited information about the client. This can be addressed by client-side targeting techniques: the advertiser can send more than one candidate ad, each of which is labeled with a full ID (or a range of IDs) that has the truncated ID as a prefix. The browser can then render the ad locally in an isolated frame based on the true ID. Because the ad is rendered locally, the advertiser never learns the true ID.

As with per-domain IDs (Section 6.1), there are a number of open issues that would have to be worked out for this to work:

- The number of candidate ads may be very large: if the ID is truncated by t bits, then there may be 2^t candidate ads. The question of whether there is a good performance/privacy tradeoff would need to be studied.
- As in the previous section, it might be possible for multiple observers to cooperate to discover the true FLoC ID. Here too, some sort of taint tracking/information flow control might help, though it might be sufficient to have all the observers on a site restricted to the shortest ID for any observer.

6.4 Information Flow Control

As noted several times above, the privacy properties of FLoC are significantly weakened by the interaction of FLoC IDs with existing Web storage and information passing mechanisms. In particular, any case in which it is possible to simultaneously access (1) the FLoC ID and (2) other browser state mechanisms such as cookies allows bootstrapping of other information from the FLoC ID. It is unclear whether there is a scalable way to address this, but one potential approach might be to restrict the use of FLoC to isolated contexts which have limited information flow both in and out. This would allow the use of FLoC without also leveraging the troublesome existing Web APIs. There has been prior work in this general area, including COWL [SYM+14] and Fenced Frames [Sha21] but further analysis would be required in order to determine feasibility.

6.5 Randomized Misreporting

Users that randomly fail to report a cohort ID might protect others who cannot provide a cohort ID. That is, users without sensitive histories might falsely suppress their cohort ID at some agreed probability. This would reduce the overall proportion of true "sensitive" users in the set of those who report no cohort ID.

This would require estimating the proportion of users with sensitive histories (a rate that is unlikely to be well-represented in Chrome sync data). The randomized suppression rate could then be adjusted to reduce the proportion of those who are genuinely sensitive from those who send no cohort ID.

One potential problems with this approach is that it suppresses the cohort ID for those who are randomly selected, which might in turn reduce advertising efficacy.

6.6 Falsely Reporting Sensitivity

As noted in Section 5.1, it is possible for an attacker to mount a sybil attack on the k-anonymity algorithm by forging reports for cohorts of special interest. As this is a narrowly targeted attack, it could be quite cost-effective.

One potential countermeasure is to allow users to report cohorts/topics which they believe sensitive. This could estimate the total proportion of reports that claimed sensitivity in this phase, using the total number of reports — rather than the number of reports for a given cohort — as a baseline. Cohorts which were above threshold could then be omitted from FLoC reporting, even if it was otherwise large enough to be included. An attacker who wanted to track a given cohort would then need to falsify a similar number of reports against every other cohort. This gives an advantage to those who want to mark cohorts as sensitive in the order of the number of cohorts, divided by the number of cohorts they want to protect. This advantage increases if the targeted sensitive topic can be kept secret from the tracker.

A lot here rides on the strength and composition of defenses against bots and falsified reports. It is also potentially quite wasteful if it results in escalation.

7 Summary

FLoC is intended to allow for interest-based ad targeting while bounding the amount of information available to observers. Our analysis suggests that this first iteration does not adequately achieve this privacy objective. When considered as replacing existing state-based cross-site tracking mechanisms such as cookies, FLoC still allows for significant linkability of user behavior, both via measuring the longitudinal evolution of cohort IDs as proposed by Wilander and in combination with existing partial identifiers such as IP address, geolocation and browser fingerprinting. In particular, it may be possible to identify individual users by using both FLoC IDs and relatively weak fingerprinting vectors. When considered as coexisting with existing state-based tracking mechanisms, FLoC has the potential to significantly increase the power of cross-site tracking. In particular, in cases where cross-site tracking is prevented by partitioned storage, the longitudinal pattern of FLoC IDs might allow an observer to re-synchronize visits by the same user across multiple sites, thus partially obviating the value of these defenses.

It is unclear if it is possible to address these issues within the general structure of FLoC. Google and others have proposed some mitigations but it is too early to evaluate them in detail. We have presented a number of potential approaches, but none is clearly satisfactory and several of them appear to be subject to attack in ways that are not straightforward to repair.

Acknowledgements

Steven Englehardt contributed extensively to this document while working at Mozilla. Thanks to Karthik Bhargavan, Robin Berjon, Daniel Kahn Gillmor, Joseph Lorenzo Hall, Mallory Knodel, and Brad Lassey for their helpful comments.

References

[Bar11] Adam Barth. HTTP State Management Mechanism. RFC 6265, April 2011.

[BBCG+21] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. Cryptology ePrint Archive, Report 2021/017, 2021. https://eprint.iacr.org/2021/017.

[Cyp21] Bennett Cyphers. Googles FLoC Is a Terrible Idea. https://www.eff.org/deeplinks/2021/03/googles-floc-terrible-idea, Mar 2021.

[Eck10] Peter Eckersley. How unique is your web browser? In Mikhail J. Atallah and Nicholas J. Hopper, editors, *Privacy Enhancing Technologies*, pages 1–18, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[GBLB18] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *WWW2018 - TheWebConf 2018 : 27th International World Wide Web Conference*, pages 1–10, Lyon, France, April 2018.

[Goo21] Google. Doubleclick ad exchange real-time bidding protocol: Cookie matching. https://developers.google.com/ad-exchange/rtb/cookie-guide, 2021. Retrieved, May 2021.

[Har12] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012.

[Las20a] Brad Lassey. Combating Fingerprinting with a Privacy Budget. https://github.com/bslassey/privacy-budget, Dec 2020.

[Las20b] Brad Lassey. Gnatcatcher: Global network address translation combined with audited and trusted cdn or http-proxy eliminating reidentification. https://github.com/bslassey/ip-blindness/blob/master/near_path_nat.md, 2020.

[LRB16] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 878–894. IEEE, 2016.

[MLV+20] Vikas Mishra, Pierre Laperdrix, Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Martin Lopatka. Dont count me out: On the relevance of ip address in the tracking ecosystem. In *Proceedings of The Web Conference 2020*, pages 808–815, 2020.

[SE21] Pete Snyder and Brendan Eich. A Step in the Wrong Direction. https://brave.com/why-brave-disables-floc/, Apr 2021.

[Sha21] Shivani Sharma. Fenced frames. https://github.com/shivanigithub/fenced-frame, May 2021.

[SYM+14] Deian Stefan, Edward Z. Yang, Petr Marchenko, Alejandro Russo, Dave Herman, Brad Karp, and David Mazières. Protecting users by confining javascript with COWL. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 131–146, Broomfield, CO, October 2014. USENIX Association.

[Tes20] Tess O'Connor. Parties and browsers. https://tess.oconnor.cx/2020/10/parties, Oct 2020.

[Yao21] Yao Xiao and Josh Karlin. Federated Learning of Cohorts. https://wicg.github.io/floc/, Apr 2021. Draft Web Incubation Community Group Report.