Technical Comments on Privacy Budget

Eric Rescorla
ekr@mozilla.com

September 30, 2021

Abstract

Google's Privacy Budget is a proposal to restrict fingerprinting by measuring the amount of information consumed by sites and blocking access to information-containing features when a certain limit has been exceeded. Our analysis identifies a number of potential issues with the proposal that call its practicality into question. In particular, it seems hard to practically estimate the amount of information leaked by particular surfaces and even harder to estimate the amount of information leaked by a particular combination of surfaces which may be correlated, uncorrelated, or anything in between. Deployment of the Privacy Budget proposal seems likely to cause unpredictable breakage of Web sites which find themselves exceeding the information limit, often due to actions outside their direct control, as in the case with third party scripts they have loaded. Finally, the blocking mechanism itself may be usable as a tracking mechanism by exhausting the budget in a distinct pattern for each client. We believe a better approach is to (1) limit the amount of fingerprinting surface exposed by new Web platform features (2) gradually remove fingerprinting surfaces — especially passive fingerprinting services — from existing features and (3) detect individual abusive patterns and block the sites responsible.

1 Introduction

Fingerprinting is a major threat to Web privacy. Unlike "stateful" tracking mechanisms like cookies [Bar11], which depend on explicit cooperation from a browser, fingerprinting depends on measuring stable characteristics of a given browser such as operating system, version, language, screen size, etc. These are properties that are difficult or impossible for a browser to control, allowing for tracking without active browser support. Although any given characteristic may be shared by many browsers, the combination of many such characteristics can be unique or nearly unique.

The most concerning use of fingerprinting is for cross-site tracking: many browsers are adding technical mechanisms which prevent tracking via cross-site cookies. Fingerprinting allows trackers to bypass these protections by creating a stable per-client identifier that does not depend on cross-site storage. For this reason, a number of browsers contain anti-fingerprinting techniques; these are intended to reduce fingerprinting but are imperfect and do not eliminate it.

Chrome has proposed a technique called Privacy Budget [Las20a] as a potential mitigation for fingerprinting. The general idea is to estimate the amount of information exposed by a given fingerprinting surface and then cap the total amount of information available to any given site. When a site exceeded the cap, it would get some sort of uninformative answer such as an imprecise or standardized value. If successful, this would limit — though not eliminate — fingerprinting by those sites. This note attempts to evaluate the feasibility of the Privacy Budget idea.

2 Fingerprinting Overview

The basic idea behind fingerprinting is to observe existing, stable, properties of a given user's browser and use those properties to track the user over time. These properties may be derived from a number of sources, including the user's device, the browser implementation the user is using, and the user's configuration settings. Importantly, these properties need not themselves be unique. If the combination of multiple properties is unique — or even unusual — and the fingerprinting site can link up multiple different measurements as belonging to the same user, such as if they are collected at the same time or if the fingerprinting site uses a tracking cookie to follow the user over time, then the site can build a fingerprint value which is usable for reducing the user's anonymity set.

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 and common estimates are more like 18–20 bits. Fingerprints are not evenly distributed: Gómez-Boix et al. [GBLB18] report that out of a sample of over 2 million users, 33.6% of users had unique fingerprints.

Note that these measurements do not include the IP address, which of course is extremely identifying, often down to the level of individual users [MLV$^+$20].

3 Overview of Privacy Budget

This section provides an overview of the Privacy Budget idea. We note in advance that Chrome's development of Privacy Budget appears to be at the conceptual stage and consists solely of an explainer and a FAQ, neither of which are sufficient to implement the idea, so we are forced to speculate a bit. This section also introduces some new formalism to assist with the analysis.

The first step of the process is to enumerate all the *fingerprinting surfaces* of the browser. A fingerprinting surface is just a specific characteristic which potentially varies from browser instance to browser instance and is potentially stable for the same browser instance. We denote the set of surfaces as $\mathbf{S}$ and the ith surface as S_i. These surfaces may be either *active*, meaning that the site has to take an action visible to the browser such as calling an API in order to read them, or *passive*, meaning that the site just gets them as part of normal operation, such as an the browser type.[1]

The browser vendor must then measure the information content I_i of each surface. Chrome suggests using browser telemetry for this purpose:

> Each browser can start by instrumenting itself to estimate the information revealed by each surface and report it back through telemetry. This data will be used for the next phase.

What matters is the distribution of values, not the raw possible informational content. As a concrete example, QUIC version numbers are 32 bits long, but each browser uses the same QUIC version (`0x00000001`), so this surface actually has effectively no information content even though potentially it could be very large. Thus, measuring the information content of a given surface requires collecting the values of a large number of clients and estimating the distribution of those values. Note that as long as it is possible to distinguish between different browser types, it is not sufficient to examine the global value distribution, but instead one must measure distribution across instances of a given browser type.

[1]Currently, sites also learn the specific browser version, but there have been proposals to stop providing that. It is generally not practical to conceal the browser engine from sites because different engines have different networking behavior (see, for instance [FW]), though potentially multiple browsers based on the same engine such as Chromium and Edge might be hard to distinguish on the same OS.

Not all *values* of the same surface have the same informational content. For instance, because Chrome is so popular, knowing that a given user uses Chrome yields very little information, but knowing that the user uses Tor browser is very informative because Tor usage is so infrequent (formally, the information content is $I(E) = -log_s(p(V))$ where $p(V)$ is the probability of obtaining that value). Google does not specify how to address this (though see `https://github.com/bslassey/privacy-budget/issues/5`), but it seems like there are two main approaches to measuring information content:

- Label each surface S_i with the overall Shannon entropy [Sha48] for the surface. This would yield a constant value for each surface.
- Estimate the information content of each potential value for S_i and use the appropriate value as the information content when surface is read. This would require providing the browsers with a table of the distributions/information contents for each potential value.

Neither of these turns out to be really satisfactory, as described in Section 4 below.

In order to use this information, browsers would have an overall privacy budget value B (presumably expressed in bits). They would then track which surfaces $\mathbf{R} \subset \mathbf{S}$ each site had read and use that to calculate the total amount of budget that the site had consumed. The budget would be maintained for the current page load and as long as the site had stored explicit state (e.g., cookies) that would allow it to connect multiple values.[2] As long as the site had not exceeded its budget, when it tried to read surface i it would get the true value V_i. Once the site exceeded the privacy budget, it would get a degraded answer. E.g.,

$$Result = \begin{cases} V_i & \sum_j I(R_j) < B \\ degraded(V_i) & \sum_j I(R_j) >= B \end{cases} \tag{1}$$

The term "degraded" here is ours. Google's document says:

> Once we're ready to enforce the privacy budget, subsequent API calls that violate the budget will either result in an error being thrown or, if possible, will be replaced with a privacy-preserving version of the API that either returns imprecise or noisy results or a generic result that doesn't vary between users. Another option to consider is to deny storage and network requests to the page after the budget is hit such that it cannot exfiltrate any new information it learns.

Each of these options has somewhat different implications.

- If access returns an error, then it is possible to determine which surfaces have been previously read and use that as a fingerprint, as described in Section 7.
- If access returns a generic value, then this still leaks *some* information about whether a value was read. I.e., if it is *not* the generic value, then the query was within budget.
- Imprecise results still leak some information, which must be accounted for in the budget. E.g., if you quantize screen width by multiples of 1024, it allows for distinguishing smartphones from large screen displays.

[2] The reasoning here is presumably that if the site cannot connect two different readings, then they are not able to use them to build a single fingerprint.

- Noisy results seem problematic. Either the result is stable or semi-stable for a given site (though perhaps giving different answers for each site), in which case it is usable for fingerprinting, or it is inconsistent for that site, in which case it is trivial to determine whether the query succeeded or failed by just making repeated queries. In addition, if it is randomly distributed around the true value, then repeated queries can approximate the true value.

It is also necessary to adjust the budget for passive fingerprinting vectors; they must be treated as already read by the site because it is not possible to know if the site is using them.

3.1 Multiple Reads of the Same Surface

One challenge here is multiple reads of the same surface. For instance, consider a site which asks for `window.screen` on each load in order to properly size its content (aka "responsive design"). As window size is a classic fingerprinting property, this surface be incorporated into the budget calculation, but how do we handle multiple reads? Google's explainer does not specify.

There are at least two obvious alternatives:

1. Treat each read as distinct and decrement the budget each time.
2. Treat each read as duplicative and decrement the budget only the first time.

The first of these is clearly unsatisfactory: existing sites routinely read properties on each load and even relatively low information properties would quickly exceed the budget, thus resulting in breakage after repeated loads. The second option seems somewhat more practical, especially for values which are entirely or at least mostly static (e.g., operating system or hardware platform): repeated reads of truly static values do not provide additional information. However, it is unclear what to do about values which are somewhat but not completely stable, such as screen size, as repeated reads do provide some additional information. This is especially true if different values for the same surface have different budget impacts. It may be necessary to treat some surfaces one way and some another. In a final proposal, this would need to be fleshed out, but for the purposes of this document, we assume that repeated reads of the same property succeed as long as the initial read was permitted.

3.2 Enforcement Domain

The budget could be enforced either based on the origin or at the top-level site.[3]. As observed by Snyder and Livshits [SL19], the origin is unworkable because the top-level page can simply instantiate multiple subresources that measure different surfaces and join the results. Our understanding from discussions with Google is that the intent is to have the the budget tied to the top-level site. Unfortunately, this has the potential to create unpredictable breakage because third-party subresources out of the control of the top-level page will be in a position to consume the page's budget. It is unclear how to address this issue. See Section 6 for more detail on this.

[3]The First Party Sets proposal recommends applying the policy to all sites in a set; see `https://privacycg.github.io/first-party-sets/#applications`

4 Estimating Information Content

As implied by the discussion in Section 3, estimating the information content of a given value is not straightforward even if we know the distribution. If we use a fixed value for the entropy of the surface, we will often be overestimating the amount of information available to the site (because there are likely to be a few common values) and then occasionally will underestimate (if there are a few unique values). On the other hand, if we use the information content of the specific value, then we must have detailed knowledge of the distribution and be continuously adjusting it for new values, as wall as regularly updating clients. Moreover, having the amount of budget consumed by a surface vary depending on the value returned creates issues for sites, as discussed in Section 6.

In addition to difficulties determining the information content of individual surfaces, it is not straightforward to *combine* the information content of multiple surfaces, because those values are not independent. Thus, if you naïvely add up all the informational values of a given fingerprinting surface, as indicated in Equation (1), the result grossly overestimates the amount of information available to the site. As a trivial concrete example, the `window.screen` property denotes the dimensions of the display that the browser is on and contains `width` and `height` properties. However, they are highly correlated and thus if we treat them as separate fingerprinting surfaces, we are effectively double charging the site.

A lot of fingerprinting characteristics depend on the choice of browser, OS, and hardware. Though there are many different metrics that provide fingerprinting information, these metrics are often linked back to one choice, from which there are a limited number of options. For example, networking behavior is largely dictated by the implementation of the networking stack, parts of which are provided by the browser or OS; the true information conveyed there is best captured by observing the choice of browser or OS, not the properties that a site might observe such as how HTTP headers are populated. Overestimating the budget consumed by sites is especially problematic because the available budget is actually quite low, as described below. This could lead to failures (see Section 6).

Accurately measuring the amount of information provided to the site requires measuring the *joint* distribution of all of the fingerprinting surfaces in the browser. With that information in hand, one could in principle estimate how many new bits of information each value read has provided and appropriately debit the budget. This would be an extremely large table[4] that would need to be distributed (or at least summarized) to every client. Moreover, because it might contain sensitive values it would have its own privacy issues. For these reasons, actually taking this approach seems impractical.[5] Indeed, even collecting this data is somewhat problematic, as it requires the browser vendor to fingerprint the client.

One challenge with any attempt to set a fingerprinting budget is that the IP address itself is sufficient to identify many users; even in cases where it is not actually unique, it can often be combined with a small amount of other fingerprinting information to produce a unique identifier. Thus, any effective anti-tracking mechanism must have some way to prevent IP-address based tracking. There have been a number of mechanisms proposed to address IP-based tracking, including proxy-based approaches [Las20b] [App21] and auditing/policy based approaches such as Google has proposed in Willful IP Blindness [Las21]; we leave discussion of those mechanisms to future work. Without this kind of mitigation, the major defense against IP address tracking is for clients to block connections to trackers, in which case the need for a separate anti-fingerprinting defense would be significantly lessened.

[4]It would also be sparse and we would need to figure out how to handle value combinations that were not present in the table.

[5]An additional problem here is handling new values which have never been seen before. These necessarily represent a very small fraction of the user population and so leaking that such a value is present leaks information.

5 Sizing the Budget

A key parameter of a system like this is the size of the budget itself and the Google explainer does not specify how to set this value. Given that the total number of Internet users is on the order of 2^{32}, 32 bits seems like the upper limit. However, in practice this is very likely excessive, for two reasons.

First, in the "best case" scenario where users are independently distributed with respect to fingerprints, the anonymity set for a given fingerprint is $\frac{|U|}{2^b}$ where U is the set of users and b is the size of the fingerprint. For instance, if there are 2^{32} users and 20 bits of fingerprinting, then each anonymity set is of size 4096 even without any additional information that would allow profile construction[6] There is not consensus on the minimum size of the anonymity set, but it seems likely that at least 8 bits is necessary which gives a new limit of closer to 24 bits.

However, users are not independently distributed with respect to fingerprints: some fingerprints will have more or less users, with the result that we must solve for the worst case. This can be potentially addressed by weighting each value as described in the previous section, but any inaccuracies in weighting may cause anonymity sets to be smaller than expected.

6 Impact on Sites

As described in Section 3, the assumption is that the browser will allow sites to access a given fingerprinting service until they have consumed their privacy budget and then they will receive degraded answers. This seems like it has a number of issues.

The first issue is the impact that applying a privacy budget will have on existing sites. It's very common for sites to routinely request large amounts of data that can be used for fingerprinting. Table 1 shows the frequency at which the JavaScript attributes measured by AmIUnique [7] appear in the Chrome Platform Status Dashboard [8].

Without tracing through sites it is difficult to know how many of these APIs are being used for non-fingerprinting purposes (e.g., customizing site behavior). However, it seems likely that as-is many sites will exceed any plausible fingerprinting budget.

This is especially true because much of the fingerprintable surface of browsers is in information which is passively provided to the site (e.g., User-Agent, Accept.); as described in the Privacy Budget Explainer, it is necessary to assume that sites are using this information because it is not possible to determine if they are accessing it. Table 2 shows the set of information carrying values in an HTTP request from Firefox along with the corresponding entropy estimates from [GBLB18].

The explainer proposes moving some of these values to Client Hints [GW21], which would allow it to be an active surface, however there are other vectors (e.g., TLS cipher suites, QUIC congestion control behavior, etc.) which can be remotely observed and are proxies for UA and version, so it's unclear how successful this will be.

The impact of sites going over the privacy budget depends on the order in which the various information surfaces are accessed: earlier accesses will get correct information while later accesses will get degraded

[6]E.g., if only one user with fingerprint X is interested in cars, then a visit to *Motortrends* and a visit to *Car and Driver* with the fingerprint X probably identify the same user.

[7]`https://www.amiunique.org/fp`

[8]`https://www.chromestatus.com/metrics/feature/popularity`

[9]The relevant API is reading from the canvas, but this does not appear to be in the dashboard. HTMLCanvasElement has a usage of 40.435532

[10]There are alternative, older ways of enumerating browser fonts through trial and error.

Attribute	Chrome Feature	Frequency (%)
User agent	NavigatorUserAgent	92.387269
Platform	NavigatorPlatform	46.275684
Cookies enabled	NavigatorCookieEnabled	44.008556
Timezone	DateTimeFormat	12.575308
Content language	NavigatorLanguage	49.736865
Canvas	- [9]	
List of fonts (JS)	DocumentFonts	24.052605[10]
Do Not Track	NavigatorDoNotTrack	17.986316
Product sub	NavigatorProductSub	5.808992
Vendor	NavigatorVendor	37.643527
Vendor sub	NavigatorVendorSub	2.562682
Hardware concurrency	NavigatorHardwareConcurrency	28.711425
Device memory	NavigatorDeviceMemory	28.565478
List of plugins	NavigatorPlugins	48.135067
Screen width	V8Screen_Width_AttributeGetter	64.822413
Screen height	V8Screen_Height_AttributeGetter	64.713506
Screen depth	V8Screen_ColorDepth_AttributeGetter	49.922662
Screen available top	V8Screen_AvailTop_AttributeGetter	13.389985
Screen available Left	V8Screen_AvailLeft_AttributeGetter	9.048967
Screen available Height	V8Screen_AvailHeight_AttributeGetter	33.345196
Screen available width	V8Screen_AvailWidth_AttributeGetter	33.215547
Screen left	WindowScreenLeft	15.138352
Screen top	WindowScreenTop	14.090885
Permissions	V8Permissions_Query_Method	16.698676
WebGL Data	V8WebGLRenderingContext_GetExtension_Method	18.240619
Use of local storage	OBSOLETE_LocalStorageFirstUsedBeforeFcp	62.378766
Use of session storage	SessionStorageFirstUsedBeforeFcp	44.318258
Use of IndexedDB	UnprefixedIndexedDB	49.3671
Audio formats	V8HTMLMediaElement_CanPlayType_Method	20.396475
Audio context	V8AudioContext_Constructor	2.38653
Frequency analyser	WebAudioDezipperOscillatorNodeFrequency	0.7015
Audio data	V8AudioContext_Constructor	2.38653
Video formats	V8HTMLMediaElement_CanPlayType_Method	20.396475
Media devices	MediaDevicesEnumerateDevices	7.923699
Accelerometer	LinearAccelerationSensorConstructor	0.0031
Gyroscope	GyroscopeConstructor	0.139054
Battery	BatteryStatusGetBattery	9.700758
Connection	NetInfo	42.901908

Table 1: Frequency of fingerprinting surface access

information. If the later accesses are used for site tuning (e.g., responsive design) then the impact may be minimal, but if they are used for site features such as videoconferencing or WebGL, then the site may just fail. Some of these APIs (e.g., `getUserMedia()`) already come with permission prompts which Google suggests may be sufficient to grant a budget exemption, but others, such as WebGL do not; it seems

Value	Header	Entropy
Platform	`User-Agent`	1.200
Do Not Track	`DNT`	1.919
Header Accept	`Accept`	1.383
Content Encoding	`Accept-Encoding`	1.534
Content Language	`Accept-Language`	4.918
User-Agent	`User-Agent`	9.779
List of HTTP Headers	N/A	4.918

Table 2: Entropy values for HTTP headers

implausible to add new permission prompts to a large fraction of sites even if we ignore the difficulty of explaining them to users. This is especially true for APIs where there is no natural way to provide degraded functionality.

The breakage problem is exacerbated by being unpredictable. First, because earlier API calls fail while later calls succeed, changes in a site — or its dependencies — can break otherwise working sites when a critical access moves from inside to outside the budget. Moreover, sites cannot afford to consume their entire privacy budget at any given time lest they might later decide to add a new script/feature which might access some information surface and would then break.

Sites may even experience inconsistent breakage depending on network conditions: some browsers will load script A first and some will load script B first. Similarly, because JavaScript can execute asynchronously, the order in which API calls happen is indeterminate. Addressing this would require substantially restructuring the site JavaScript which would be disruptive and might have performance consequences. The unpredictability problem becomes even worse if we use the true information content of a given value rather than the entropy of a given surface: a site might work with common browsers but then fail with unusual configurations where each surface carries more information and thus the budget is exceeded more quickly. The unpredictable nature of breakage under these scenarios could make such breakage difficult for web developers to debug and for users to understand, potentially leading to a significant support burden as well.

7 Usability for Tracking

As Snyder and Livshits [SL19] observe, it is potentially possible to exploit the privacy budget itself for tracking purposes. The most serious application for this is for cross-site tracking in which the tracker is loaded on multiple sites and wants to measure user behavior across those sites. Historically, such trackers would use cross-site cookies, but as browsers increasingly are blocking or partitioning such cookies, fingerprinting becomes a more attractive approach. The Privacy Budget idea is intended to prevent this kind of cross-site tracking but because the budget itself is state that can be both manipulated and read but is *not* partitioned, it can be used for cross-site tracking.

The general idea is to measure which pieces of the budget have been *used* by a given domain. Consider the simplified case where every surface has a single bit of entropy and it is possible to determine whether a query to a surface succeeded or failed. Assume that the privacy budget is B bits. The attacker randomly (without replacement) queries B surfaces out of $\mathbf{S}$. There are now two cases:

- *New client.* In the case where the attacker has never seen this client before, all the queries will succeed and the privacy budget will be exhausted.

- *Return client.* In the case where the client has talked to the attacker before, then only queries which the attacker made during the initial visit will succeed.

If all the attacker's queries succeed, this is a new client[11] and can be noted as such, with the probabilistically unique identifier consisting of the list of successful queries and their values. There are two components to this identifier: (1) the set of queries which succeeded (randomly chosen out of a set of $\binom{|S|}{B}$ values) and (2) the actual values themselves. In this naïve version, the attacker thus gets $2B$ bits of entropy.

If any of the attacker's queries fail, then this is a repeat client. The attacker then queries every surface in **S** and notes which queries succeed. This list is the identifier for the user.

Of course, in practice the situation is somewhat more complicated because each surface has a different amount of entropy, so the optimal query pattern is non-obvious — especially if the budget is debited by different amounts for each response. In addition, the browser might make things more difficult by attempting to conceal failed attempts to access a given surface rather than just generating an error. However, because that value needs to be deterministic for a given browser/site pair, it will still leak some information. Moreover, some surfaces (e.g., WebGL) are not consistent with silent failure.

8 Alternative Approaches

For the reasons above, we have concerns about the feasibility of the current Privacy Budget proposal. The proposal is fairly high level and is possible that there are some variations which would improve the situation, but in general our analysis does not overly depend on the precise implementation details and it is unclear how they could be improved to address the issues we identify above.

While it is not clear that a generic solution to fingerprinting is possible, there are a number of known approaches that can help limit the risk of fingerprinting.

Minimize fingerprinting surface in new Web platform features. When designing new Web platform features we should strive to minimize new fingerprinting surfaces, especially for passive fingerprinting. It is not always possible to avoid leaking all information but with care the amount of leakage can be limited, as with the design of the MediaDevices API [JAB+21].

Reduce existing fingerprinting surfaces. Although more difficult, we can gradually reduce the fingerprinting surfaces that are already exposed to sites. One potential approach is to create new ways of accomplishing tasks that do not depend on the release of information that might be used for fingerprinting, e.g., by pushing more logic onto the client. Other approaches include converting passive surfaces to active surfaces, thus making it potentially more possible to detect abuse.

Detect and block fingerprinting trackers. While it is difficult to automatically enforce fingerprinting limits on the client side, it is possible to detect fingerprinting behavior and then block the offending scripts. Iqbal, Englehardt, and Shafiq [IES21] describe a machine-learning system for this based on global Web scraping. It may also be possible to use privacy-preserving measurement techniques [CGB17] [BBCG+21] to remotely detect suspicious sites.

Browsers are already using these approaches and while they are not a perfect solution, we believe they still are more promising than expecting to use the Privacy Budget as a general defense against fingerprinting.

[11]Unless the number of surfaces far exceeds B, then the chance that at least one query will fail for a repeat client is very high (lower-bounded by $1-(\frac{|S|-B}{|S|})^B$.)

9 Summary

Browser fingerprinting is a major threat to user privacy. However, based on our analysis the Privacy Budget proposal is unlikely to be a practical approach to addressing this issue. We have a number of specific concerns:

- Estimating the amount of information in a single surface seems difficult: neither the overall entropy of the surface nor the information of a given value seem suitable.
- Estimating the aggregate amount of information from multiple measurements is difficult because there is a high degree of correlation between values. This makes it hard to know how much to debit the budget when multiple values are taken.
- Any design in which queries made after the budget is exceeded fail will lead to unpredictable breakage and website compatibility issues and will make it much harder to build a site.
- Sites already query a large number of potential fingerprinting surfaces, leaving very little — if any — headroom before they have exceeded their budget; this makes transition difficult.
- The Privacy Budget mechanism itself can be used for tracking based on exhausting the budget in a different pattern for each user and then measuring which surfaces work on future visits.

Rather than attempting to contain the level of fingerprinting that any given site is allowed to do, we believe a better approach is to (1) limit the amount of fingerprinting surface exposed by new Web platform features (2) gradually remove fingerprinting surfaces — especially passive fingerprinting services — from existing features and (3) detect individual abusive patterns and block the sites responsible.

Acknowledgements

Thanks to Bobby Holley and Tom Ritter for gathering usage data. Thanks to Steven Englehardt for many useful comments. A number of the points in this document were previously made by Snyder and Livshits [SL19].

References

[App21] Apple. Apple's improved privacy controls. WWDC, 2021. `https://youtu.be/vqwtmq5ctLE`.

[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`.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 259–282, Boston, MA, March 2017. USENIX Association.

[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.

[FW] Sergey Frolov and Eric Wustrow. The use of tls in censorship circumvention.

[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.

[GW21] Ilya Grigorik and Yoav Weiss. HTTP Client Hints. RFC 8942, February 2021.

[IES21] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1143–1161. IEEE, 2021.

[JAB$^+$21] Cullen Jennings, Bernard Aboba, Jan-Ivar Bruaroey, Henrik Bōstrom, and Youenn Fablet. Media Capture and Streams. `https://www.w3.org/TR/mediacapture-streams/`, aug 2021.

[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.

[Las21] Brad Lassey. Willful ip blindness. `https://github.com/bslassey/ip-blindness/blob/master/willful_ip_blindness.md`, 2021.

[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. Don't count me out: On the relevance of ip address in the tracking ecosystem. In *Proceedings of The Web Conference 2020*, pages 808–815, 2020.

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[SL19] Pete Snyder and Ben Livshits. Brave, Fingerprinting, and Privacy Budgets. `https://brave.com/brave-fingerprinting-and-privacy-budgets/`, nov 2019.