Fingerprinting SDKs for Mobile Apps and Where to Find Them: Understanding the Market for Device Fingerprinting

Michael A. Specter*
Georgia Tech
Atlanta, GA, USA
Google LLC
Mountain View, CA, USA

Mihai Christodorescu
Google LLC
Mountain View, CA, USA

Abbie Farr
Google LLC
Mountain View, CA, USA

Bo Ma
Google LLC
Mountain View, CA, USA

Robin Lassonde
Google LLC
Mountain View, CA, USA

Abstract

This paper presents a large-scale analysis of fingerprinting-like behavior in the mobile application ecosystem. We take a market-based approach, focusing on third-party tracking as enabled by applications' common use of third-party SDKs. Our dataset consists of over 228,000 SDKs from popular Maven repositories, 178,000 Android applications collected from the Google Play store, and our static analysis pipeline detects exfiltration of over 500 individual signals. To the best of our knowledge, this represents the largest-scale analysis of SDK behavior undertaken to date.

We find that Ads SDKs (the ostensible focus of industry efforts such as Apple's App Tracking Transparency and Google's Privacy Sandbox) appear to be the source of only 30.56% of the fingerprinting behaviors. A surprising 23.92% originate from SDKs whose purpose was unknown or unclear. Furthermore, Security and Authentication SDKs are linked to only 11.7% of likely fingerprinting instances. These results suggest that addressing fingerprinting solely in specific market-segment contexts like advertising may offer incomplete benefit. Enforcing anti-fingerprinting policies is also complex, as we observe a sparse distribution of signals and APIs used by likely fingerprinting SDKs. For instance, only 2% of exfiltrated APIs are used by more than 75% of SDKs, making it difficult to rely on user permissions to control fingerprinting behavior.

CCS Concepts

• **Security and privacy; Mobile and wireless security**;

Keywords

Mobile Apps, Fingerprinting, Static Analysis, Cross-app Tracking, Transparency, Android

ACM Reference Format:
Michael A. Specter, Mihai Christodorescu, Abbie Farr, Bo Ma, and Robin Lassonde. 2025. Fingerprinting SDKs for Mobile Apps and Where to Find Them: Understanding the Market for Device Fingerprinting. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3744877

*Email address to contact the authors: mobile-fingerprinting-sdks-paper@google.com

CCS '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3744877

1 Introduction

Device fingerprinting is a technique used to identify and track user devices by collecting a wide range of information about device-specific hardware, software, and configuration settings. The combination of these attributes creates a unique, or near-unique, digital "fingerprint" for that device. This process has clear privacy concerns—fingerprinting identifiers can be collected without user control or notice, and persist over the device's lifetime regardless of most user privacy-seeking actions (e.g., clearing one's cookies, rotating advertising IDs, or enabling private browsing).

Both major mobile operating system vendors have undertaken significant efforts to limit the privacy impact of device fingerprinting. Apple has introduced policies that require apps to request user consent to collect tracking-relevant device data [5], and provide human readable explanations for the use of specific high entropy "required reason" APIs [4]. Both Google and Apple's mobile platforms now require developers to provide nutrition label-style privacy information to the user [3, 24], either as metadata submitted to their respective application stores or attached as part of the application itself. Google is developing a privacy sandbox for the web [1], and, on Android, a new sandbox that restricts third-party advertising libraries from accessing sensitive information available to the rest of the application [25]. These interventions are promising—providing much needed transparency and accountability.

The success of such anti-fingerprinting efforts depend on the technical implementation, market fit, and intention of the application's developer. For example, Apple's anti-tracking and app transparency policies explicitly allow the collection of fingerprinting data for anti-fraud purposes [5], and Android's Privacy Sandbox focuses solely on isolating code from third-party advertisers. Apple's "required-reason APIs" approach also has limitations; it currently applies to just 30 APIs, and its effectiveness depends on what other data points are collected across the wider fingerprinting ecosystem. Therefore, characterizing the technical implementation of fingerprinting in the wild and understanding stakeholders involved would provide invaluable insight into the effectiveness of these interventions.

This paper presents a comprehensive, large-scale analysis of device fingerprinting practices within the Android application ecosystem. We adopt an empirical approach centered on the identification of third-party Software Development Kits (SDKs) integrated in mobile applications, measuring their market reach, tracking methodologies, and privacy impact. To the best of our knowledge, this research represents the most comprehensive analysis of SDK behavior regarding privacy-invasive practices like device fingerprinting, with an extensive dataset of over 228,000 unique SDKs and 178,000 Android applications. Our methodology attempts to enable a more nuanced understanding of the scale and scope of device fingerprinting in mobile ecosystems; we avoid applying our own potentially biased or narrow definitions on fingerprinting behavior, and adopt a number of techniques to provide reliability and consistency when subjective analysis is unavoidable.

While many studies have measured the impact of fingerprinting (see §2 for related work), there is a dearth of knowledge surrounding the purpose of fingerprinting behavior. For example, no prior study has attempted to understand what kinds of third parties collect sufficient information to fingerprint a device, and how this fits in with the needs of the first party developer. Characterizing the overall problem from the perspective of developers can help determine why these techniques are used, provide invaluable insight into what is required to better preserve user privacy, and interpret the value of assumptions underlying current and proposed enforcement methods.

There are a number of challenges that significantly complicate our study. Any fingerprinting-detection mechanism will likely be incomplete, as there are many (potentially stealthy) methods of collecting entropy from a device, including timing information, instruction execution quirks, and other hardware-specific sources. Categorization and analysis of SDKs is also a difficult task—while *applications* self-label their use and market-fit, current SDK distribution methods do not require SDK authors to provide significant descriptions of their code. We describe the solutions to these and other challenges in depth in §3.

One important challenge is definitional: The claim that a service is fingerprinting suggests intent of the author of the code, which is most often practically unknowable. Applications may collect sufficient information to uniquely identify a device for any number of reasons, including analytics, crash reporting, anti-fraud, or through normal operation of the application itself. We emphasize that our study is purely observational—we measure *fingerprinting behavior*, and ascribe no motivation to the authors of the code. We also emphasize that our choice to examine the Android ecosystem is entirely due to convenience, and that our results are likely to extend to iOS as well. As noted in prior work [33], whereas Android's open ecosystem allows for scalable analysis, iOS's digital rights management scheme actively hinders the same.

We answer the following research questions:

RQ1: What types of behaviors do *self-identifying* fingerprinting SDKs exhibit?
RQ2: What are the stated purposes of SDKs with likely fingerprinting behavior?
RQ3: What kinds of apps use SDKs with likely fingerprinting behavior, and how prevalent are these SDKs in real world apps?

We find that many kinds of SDKs collect sufficient information to track a user (at least 20 signals exfiltrated per SDK), and that there is a large diversity in signals collected (SDKs exfiltrate 75.5 signals on average, out of a total of 504 unique signals observed across the SDK dataset). Though ads do make up a significant portion ($\approx 30\%$) of the SDKs that exhibit fingerprinting behavior, a surprising number of fingerprinting-like SDKs used in common Android applications have unclear functionality and lack significant description for categorization ($\approx 24\%$). Anti-fraud and analytics services were also prevalent in our dataset, indicating that more research must be done to create privacy-preserving alternatives to fingerprinting as used in such functionality. Finally, SDKs that exhibit likely fingerprinting behavior are disproportionately popular—roughly 10× more installs than non-fingerprinting alternatives—and individual SDKs are likely to exist across multiple application market segments (e.g., health and dating).

Roadmap. We begin in §2 with important background and prior work, as well as a short overview of the fingerprinting threat model. In §3 we describe our dataset, analysis pipeline, and labeling methodology. Next, §4 presents an overview of our results, and we conclude with a discussion and examination of our study's limitations in §5.

2 Background & Related Work

To the best of our knowledge, our work is the first large-scale study of native app-based device fingerprinting in the wild.[1] No prior research has attempted to understand why this phenomenon is common, or the market surrounding the use of these tools. Much of the existing literature stems from examining fingerprinting as an attack, with a focus on novel methods of fingerprinting.

Android applications & SDKs. Android applications may be written in any language, and can be installed from arbitrary sources including the Play store, secondary app stores, side-loading, or may come pre-loaded on-device from the manufacturer. As a result, much of Android's security model revolves around sandboxing applications using a combination of SELinux's SEPolicy and standard Linux UID-style access control mechanisms. In addition to sandboxing, access to certain sensitive data is declared via metadata provided by the app, and enforced via both install and run-time permissions checks [40]. Unless manually sandboxed, Android third party libraries (called *SDKs*) execute in the first-party application context, and therefore enjoy the same permissions as the first-party application.

SDKs may be distributed as either raw code or automatically downloaded at build time from any number of repositories or build systems. A commonly used build system is the *Maven* format, an open standard for Java dependency resolution. In practice, distribution of SDKs is commonly done using a build tool called Gradle, which loads SDKs from any number of public Maven repositories.

Fingerprinting signals. In this paper we define a *signal* to be an individual data point collected from a device. Different papers have different terms for a unique datapoint in a fingerprint, Eckersley [18] calls it a variable. Signals may be obtained from many sources including API calls, common files on the platform, system properties, hardware quirks, or runtime environment values.

Diversity of signals. Though fingerprinting is commonly discussed in the context of browsers [18, 19], there is a rich literature surrounding the many methods of fingerprinting via native code. Fingerprintable components include the microphones and speakers [9, 14, 66], the accelerometer, the gyroscope, and the magnetometer [9, 16, 38, 52, 55, 64, 65], the hardware clock [32, 48], the

[1] We make a distinction here between *web-based* and *on-device* app fingerprinting. E.g., prior studies have looked into the prevalence of fingerprinting on the web. Here we are exclusively interested in fingerprinting that occurs on-device outside of a web context.

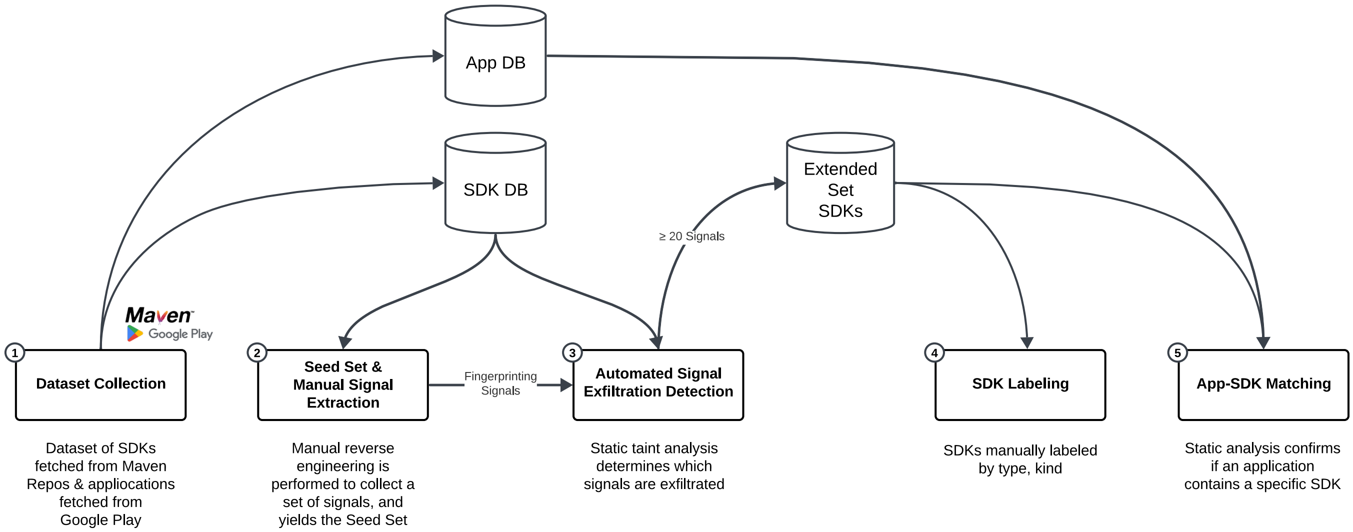


Figure 1: An overview of our analysis pipeline. We begin by ① fetching apps, SDKs, and associated metadata from a series of Maven repositories and the Google Play app store, only selecting applications installed on $> 10k$ active devices (§3.1). We continue by ② extracting a seed set of SDKs whose copy indicates that they are fingerprinting (§3.2). We then ③ use static taint analysis to determine which SDKs exfiltrate these signals (§3.3), and ④ manually label the resulting SDKs to determine their market fit (§3.4). Finally, we ⑤ perform another round of static analysis to determine which applications contain which SDKs (§3.5).

camera [8, 46], the GPU clock [41], and the battery [13, 44]. Fingerprinting of the system or the apps ranges from specific APIs [45], to system configuration (e.g., via `procfs` [50, 53]), user settings [35], and browser configuration [18, 51]. Fingerprinting can also be done by communicating with neighboring devices via short-range radio protocols such as Bluetooth [34]. Distinct signals can be combined to increase accuracy [2, 12].

Detection & prevention. Detection and prevention methods include ML classification of API calls [7, 20, 29], re-calibrating sensors [15], changing system settings [30], adding random noise to collected data [15, 42], or using taint tracking to identify exfiltration of fingerprintable data [37]. Permissions systems do not offer adequate protection against fingerprinting [17, 58].

Prior measurement studies. There have been a number of studies that measure the use of fingerprinting, though the majority focus on the web [19, 43]. A significant challenge appears to be the ever-growing surface of APIs, which have been quickly adopted by fingerprinters [7].

Existing analyses of Android native systems are comparatively rare. Longitudinal analyses not only highlighted this distinctive, mobile-specific flavor of SDK-based tracking in general, but also showed that the privacy risk across time and app versions varies greatly with little correlation to existing enforcement approaches [47]. Han et. al. [27] find that the presence of privacy-risky behavior (including fingerprinting) does not seem to vary by the cost of the app, with free apps and paid apps sharing similar sets of third-party SDKs or dangerous permissions.

The closest study to ours is Torres et al's 2018 work [21] on identifying fingerprinting in applications. They find that fingerprinters on mobile devices rely more on categorical signals and less on side channels, and argue that detection and prevention of fingerprinting on mobile is distinct from web browsers. Our work uses a significantly increased scale in terms of signals, SDKs, and applications (30k vs our 178k), presenting a more complete understanding of the ecosystem, in addtion to SDK labels and further statistics.

3 Methodology

In this section, we provide an in-depth discussion of our analysis pipeline. We depict our overall process in Figure 1, and outline a summary below:

(1) **Dataset Collection (§3.1):** We fetch a dataset of SDKs by crawling popular Maven repositories and a dataset of applications from the Google Play Store.
(2) **Seed Set & Manual Signal Extraction (§3.2):** We use the advertising copy published by each individual SDK to generate a *Seed Set* that in public statements self-announce as fingerprinters. We then manually reverse engineer these SDKs to extract what *signals* each exfiltrate.
(3) **Automated Signal Exfiltration Detection (§3.3):** Using static taint analysis in tandem with data generated from our Seed Set, we determine if an SDK exfiltrates relevant fingerprinting signals. We call all SDKs that perform enough exfiltration to be exhibiting fingerprinting behavior the *Extended Set*.
(4) **SDK Labeling & Analysis (§3.4):** Unlike applications, SDKs are unlabeled, and do not carry metadata associated to market, use-case, or intended audience. To provide adequate statistics and market information, we manually label all SDKs in the Extended Set. To avoid bias and meaningless labels, we borrow coding techniques from the HCI community, iteratively developing a codebook and reaching consensus on SDK label definitions and assignments.

(5) **App-SDK Matching (§3.5):** To provide statistics on the use of each SDK, we must first determine which SDK exists in which application. We accomplish this through a series of static analysis techniques.

There are a number of reasons we focus on SDKs rather than applications as a whole. One may expect an SDK to be self-contained and have explicitly publicized functionalities. As with any modern development environment, SDKs use in applications is incredibly common, with the majority of apps using third-party libraries to support a variety of core functions. Finally, an emphasis on SDKs also makes the potential harm from third parties far clearer: users are more likely to understand and trust an application they have actively installed, but may be unaware of the transitive trust they have placed on the third party SDKs and services used by an app.

3.1 Dataset Collection

Application Dataset. We collected 3,025,417 APKs[2] published on the Google Play store over almost 18 months (from January 2023 to May 2024). We supplement this set of APKs with each application's *total audience size* – the number of active devices that an individual APK has been installed on. An active device is a device that has been turned on at least once in the previous 30 days [23].

To avoid biasing our sample set with applications that lack a significant user base, we limit our analysis to applications active on the Google Play store with a total audience size of over 10,000 from April 13, 2024 to May 13, 2024. In total, this covers 178,054 applications. While we cannot estimate how often these applications were launched, the audience-size metric assures that devices on which the apps were installed were in active use. We approximate the market reach of an app by summing all active 30-day installs of these apps — we note that this may double-count users who install, remove, and then reinstall the same app, as well as installs by the same user on one device under multiple profiles (e.g., personal and work) or on multiple devices (e.g., phone and tablet).

SDK Dataset. There is no single source of information for SDKs, instead developers supply their build system with a URL of a particular Maven repository and library ID. We collate our dataset using a custom crawler, extracting all SDKs from 9 separate large-scale Maven repositories: JCenter, Maven Central, Google, Sonatype, Spring.io, Jitpack, Bintray, and Artifactory. From these repositories, we fetched a dataset of 228,598 SDKs as well as each SDK's associated metadata. We excluded all SDKs whose version label included one of the words {"alpha", "beta", "test", "dev", "debug", "qa"}.

3.2 Seed Set and Manual Signal Extraction

From our dataset of SDKs we select a *Seed Set* of SDKs that, in their advertising copy or other metadata, openly admit to collecting information for the purpose of fingerprinting. We then manually reverse engineered each SDK, confirmed that the SDK was collecting a nontrivial set of device data, and extracted a list of signals that the SDK uploaded to a server. To avoid mislabeling, each candidate SDK is then reverse engineered *again* by a second analyst, who independently confirms the list of signals. In total, our Seed Set contains 14 SDKs, reporting over 500 distinct signals. The results from this effort are reported in more detail in §4.1.

[2]APK is the file format of Android apps. In the rest of the paper we use the term "APK" as shorthand for one Android app.

3.3 Automated Signal Exfiltration Detection

We collected an *Extended Set* of *fingerprinting-like SDKs*, consisting of SDKs that are similar in terms of collected data to the Seed Set. To obtain this set, we developed a static analysis suite that performs information-flow analysis on SDKs and their dependencies, and selected SDKs with sufficient signal overlap with our Seed Set. Note that SDKs in the Extended Set may collect signals beyond those found in the Seed Set.

To avoid over-claiming the existence of fingerprinting behavior in our dataset, we only include an SDK if it exfiltrates *more than the lowest number of signals collected by any SDK in our Seed Set.* Put another way, we only consider an SDK to be exhibiting fingerprinting behavior if it exfiltrates more than what is uploaded by an SDK that openly admits to fingerprinting. Note that this is a conservative estimate—it is likely that more sophisticated estimates of entropy would indicate that an SDK could uniquely identify a user using *less* information than what we examine here. The tradeoff here is intentional; our goal is to provide upper-bar estimates without indulging in more complicated analyses. This step results in 723 distinct SDK families, each with multiple versions, for a total of 14,178 SDK versions.

We built a static-analysis suite for Android APK and SDK analysis, and deployed an interprocedural, context-, field-, object-sensitive taint-flow tracking algorithm for fingerprinting detection. It works by tainting all fingerprinting-related data with meta information and then propagating taint at the instruction level, so that the transparency of the flow information can be achieved, allowing us to reconstruct the taint flow path to independently verify exfiltration. We claim no novelty for this analysis, though some implementation details may be of independent interest, so we include these in an extended version of this work [49].

3.4 SDK Labeling

While developers provide an attestation of the categorical use-case of their *application* as a part of submission to the Google Play store, there is no equivalent process for *SDKs*. Indeed, Maven repositories usually provide only the name of the SDK, a short explanation, and a link back to the originator of the code. This metadata is often incomplete, further obfuscating the use of the SDK. Other datasets are more complete, including the Google Play SDK Index [26], but provide only a small database of SDKs.

Here we borrow techniques from the HCI community and treat the problem as a manual labeling task. Label definitions were collaboratively developed by a team of five expert coders based on the metadata of the SDK, including the SDK's description in Maven and the content of its developer's website, from a sample of 100 random SDKs in our dataset. For brevity, an informal description of these categories is in Table 1, and a full explanation (including subcategories) is in the full version of this paper [49]. Upon reaching saturation, we split the remaining SDKs between reviewers such that each SDK was independently examined twice.

For efficiency, we limited our labeling effort to the 723 SDK families that have been detected in our application dataset, under the assumption that all versions of the same SDK have an equivalent use case and should share the same label. The resultant definitions were robust, with reviewers usually agreeing on SDK labels; the Krippendorff's alpha inter-rater reliability score of the independent

Table 1: SDK Label Definitions. Each SDK receives one label based on its Maven metadata and website description, not based on its code.

SDK Label	*Description*
Advertising	Supports displaying ads, ads bidding, ads targeting, ads mediation, or analytics for the purpose of monetization or conversion (Ex: AppLovin, Teads)
Analytics	Monitors and reports on app health (examples: TOAST Logger, RichAPM Agent), or collects the user's behavior in app (Ex: Pushwoosh, Acoustic Tealeaf)
Security & Authentication	Implements user authentication (E: Passbase, Ondato), detects fraud and related security anomalies (Ex: Incognia, SEON), and payment functionality (Ex: Alipay, PayPal).
Tools / Other	Provides navigation (Ex: Tencent Map Nav, Radar), object or person tracking (Ex: BeaconsInSpace, Foursquare Movement), communication with social networks (Ex: Facebook, Chat SDK), or other well-defined functionality (Ex: iZooto App Push, GameUp)
Unclear / Not Found	Purpose or functionality could not be determined from online metadata

labeling step was 0.804. Finally, all label disagreements were resolved and re-labeled in a meeting of the full group, meaning that any labeling disagreement was addressed by comparing labels from all five coders.

3.5 App–SDK Matching

Inspired by the large body of work in SDK identification for Android apps [6, 28, 36, 36, 39, 56, 59–63], we created an SDK-identification pipeline using a fine-grained code similarity metric that can be aggregated across code units (e.g., classes, modules) and packaging units (e.g., SDKs, SDK versions). The similarity metric relies on identifiers for system APIs (e.g., operating system calls, standard library calls), opcode frequencies, framework APIs, and string constants.

In our design, we choose parameters for this similarity metric, including the percentage of APK code similar to a known SDK sufficient to declare the SDK as present in the APK, to ensure that our results limited false positives at the risk of some false negatives. In other words, we may miss the presence of an SDK in an APK, and thus the statistical analysis in the rest of the paper provides lower bounds for the prevalence of fingerprinting SDKs. We include a detailed description of our approach in the full version of this work [49].

4 Results

We analyzed the Seed Set, Extended Set, and their prevalence in our application dataset to answer our three research questions (§1).

Table 2: Seed Set SDKs. List of SDKs that openly admit to fingerprinting, through advertising copy, developer documentation, or other copy. The number of raw number of signals they collect is on the right. Unique Signals is not a total, but the set union of all signals without repetition.

Name	*Signals*
Seon	43
Forter	69
Kaspersky AntiVirus SDK	20
Accertify (InAuth)	213
Castle	31
Microsoft Dynamics 365	128
IP Quality Score	58
Fingerprint.js	30
Shield	148
ThreatMetrix (Lexus Nexus)	94
Ravelin	30
TransUnion TruValidate	55
Socure	43
Incognia	81
Unique Signals	504

4.1 RQ1: Self-Identified Fingerprinters

What types of behaviors do self-identifying fingerprinting SDKs exhibit?

We find 14 different SDKs that admit to fingerprinting in the wild, listed in Table 2. Our manual analysis found that fingerprinting libraries exfiltrate a minimum of 20 unique signals, and an average of 75.5. Notably, the techniques used by these SDKs were straightforward, with no SDK attempting to collect more than what was available from framework-level API calls. This is a departure from what has been previously measured in the web context (e.g. audio context fingerprinting [19]), as well as the more advanced, hardware-focused techniques discussed by the academic community.

We examine the distribution of signals collected by SDKs in the Seed Set, and present an overview in Figure 2. Though the total number of signals collected is 1043, there are only 504 *unique* signals, with less than half of all signals collected by at least two SDKs. The set of fingerprinting signals collected are relatively sparse, as the individual signals that SDKs choose to select are somewhat dissimilar—Figure 3 displays the cosine similarity between different fingerprinting SDKs, with only two SDKs scoring above 0.5 (Transunion and Ravelin). Of the 504 unique signals collected, we find that only 21 individual APIs were collected by more than half of the SDKs in our Seed Set, by a maximum of 11 SDKs. We conclude that cosine similarity of individual signals (as used in prior work [21]) is unlikely to be an effective detection mechanism.

Some SDKs stand out with unique API usage patterns. For instance, Forter, Accertify, Microsoft Dynamics, and Shield appear to use a much broader range of APIs compared to others. This could suggest that these SDKs are more complex or serve a wider range of functionalities. Conversely, Kaspersky AntiVirus SDK appears to use a very limited set of APIs, possibly reflecting its more focused security purpose. Under the assumption that all of these SDKs perform fingerprinting equally well, such breadth of API usage

Figure 2: Map of signals collected by known fingerprinting SDKs in the Seed Set, with one dot per API exfiltrated. The top plot shows the percentage of Seed Set SDKs that exfiltrate that API; 80% of APIs are exfiltrated by fewer than 50% of SDKs, and only 2% of APIs are exfiltrated by 75% of SDKs.

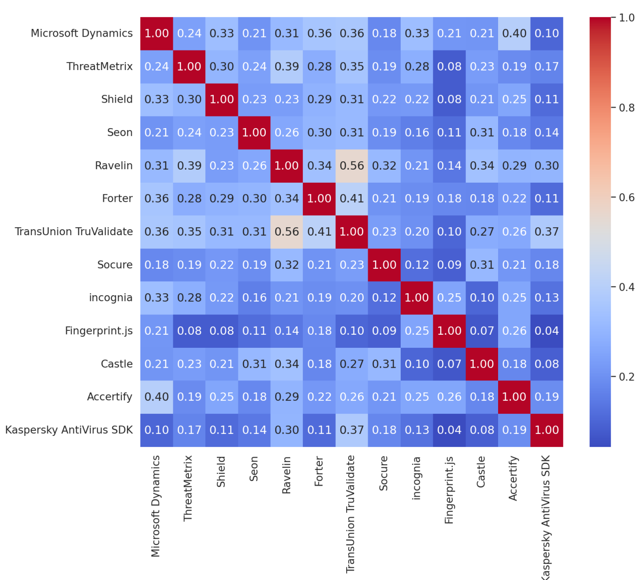


Figure 3: Cosine similarity between Seed Set SDKs, each represented as a one-hot encoding vector of their specific APIs used.

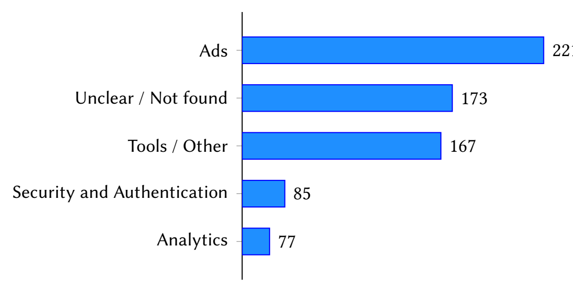


Figure 4: Prevalence of purposes across the Extended Set of 723 likely fingerprinting SDKs.

patterns implies that some APIs provide more valuable (i.e., more fingerprintable) signals than others.

4.2 RQ2: Purposes of Likely Fingerprinters

What are the stated purposes of SDKs with likely fingerprinting behavior?

The automated exfiltration detection (§3.3) yields 723 SDKs that exhibit behavior similar to the known fingerprinting behavior of the SDKs in the Seed Set. Each SDK may have multiple versions and our SDK dataset identifies 14,178 versions for these 723 SDKs, for an average of 19.60 versions per SDK. With the Extended Set labeled as described in Section 3.3 we consider the prevalence of various purposes across SDKs that exhibit fingerprinting-like behavior and plot the resulting distribution in Figure 4.

Several observations are readily available from the plot. First, there is a clear separation between the “Ads” category, the “Tools / Other” category, and the rest of the categories. The “Tools / Other” category being large is expected, as it encompasses a wide variety of functionalities, from cloud storage, to image and video processing, and to consent management. The high number of SDKs with “Ads” as purpose is potentially representative of the complex structure of that particular industry, where ad networks provide their own SDKs and ad platforms act as aggregators and mediators between apps and ad networks, with corresponding mobile SDKs that reflect these relationships. Oftentimes, the ad-mediation systems consists of 5–15 individual SDKs, one for each ad network to which the mediator can connect, easily boosting the number of total SDKs in this vertical. We opted to count these connector SDKs separately — from a security and privacy perspective they are distinct artifacts.

A second observation is there is a large contingent of SDKs whose purpose is not clear from online, public information. The "Unclear / Unfound" category is the second highest source of likely fingerprinting behavior. We know these SDKs are in use in a variety of apps (as we see later in §4.3), so the question is not only what purpose these SDKs serve, but also how to make this purpose information available to security and privacy enforcement mechanisms. One option would be to reverse engineer the code of each SDK and infer from this code its purpose, though this is unlikely to be a scalable long-term solution (and out of scope for this paper).

A further consideration for the use of privacy-preserving alternatives is the long tail of functionalities present in the "Tools / Other" category, which represents 23% of the total number of likely fingerprinting SDKs. While "Ads", "Security and Authentication", and "Analytics" are reasonably well understood and studied in terms of privacy, the "Tools / Other" SDKs cover a broad range of algorithms and data types that may not have readily available privacy-preserving alternatives.

SDK Purpose & Behavior. The challenge of identifying the purpose of likely fingerprinting SDKs of a particular type could be alleviated by focusing on their use of fingerprinting signals, if sufficiently discriminative. For example, if Ads SDKs have distinct fingerprinting behaviors (in terms of signals and API data they exfiltrate) compared to those of Security and Authentication SDKs, one could identify and control fingerprinting appropriately without relying on the SDK's purpose declaration or on its non-fingerprinting functionality, both of which could be adversarially manipulated. We evaluated this hypothesis by expressing the fingerprinting behavior of each SDK as points in a high-dimensional space defined by a one-hot encoding of the APIs exfiltrated. Each API of interest is an independent dimension in this space and an SDK is placed at position 0 along this dimension if it does not exfiltrate the corresponding API, or 1 if it does. This results in 504-dimensional space (one for each API observed in the Extended Set of SDKs) in which we locate the 723 SDKs.

In Figure 5, we provide a visual representation of the similarity between different SDK types in the Extended Set using a t-Distributed Stochastic Neighbor Embedding (t-SNE) [54] plot. t-SNE allows us to cluster the SDKs based on their "natural" similarity in fingerprinting behavior by mapping the high-dimensional space (our 504 signals) to a faithful representation in a lower-dimensional space through non-linear transformations while preserving local and global relationships between the data points. Based on the recommendations from Wattenberg, Viégas, and Johnson [57], we set t-SNE perplexity to 25, learning rate of 10, and iterations to 5,000, resulting in a final KL divergence of 0.614789. The resulting t-SNE output is shown in Figure 5, with one dot per SDK, relatively positioned as determined by t-SNE and color coded based on our five SDK purpose labels.

The t-SNE plot illustrates the diversity of signal/API usage in fingerprinting behavior, as a large number of small clusters formed. At a minimum this leads us to believe that a corresponding larger number of small, focused permission-based policies may be able to address the fingerprinting problem, with the associate risk of enforcement performance (due to the cost of maintaining and evaluating this many policies) and low usability (due to placing the user in the position of making decision based on seemingly similar but privacy-distinct permissions).

The feasibility of automating the anti-fingerprinting/anti-tracking policies put forth by the industry (advertising: no tracking allowed, anti-fraud: tracking allowed) can be reduced to whether "Ads" SDKs (marked as ◆ in Figure 5) are easily separable from "Security and Authentication" SDKs (× in Figure 5). The right third of t-SNE plot contains *most* of the "Security and Authentication" SDKs (×), while the "Advertising" SDKs (◆) are on the left. Yet there are many "Security and Authentication" SDKs on the left side of the plot, not to mention "Analytics" (■) and "Tools / Other" SDKs (+), that appear to have similar fingerprinting-like behavior to the "Ads" SDKs. Thus any automatic enforcement that needs to distinguish between "Ads" SDKs and "Security and Authentication" SDKs will need to rely on non-trivial classifiers that are more expressive than permissions.

Finally we observe that the "Unclear / Unfound" SDKs (●), which are declared in APKs and present in Maven repositories but lack any descriptive information, have fingerprinting-like behaviors similar to *all* other SDK categories. This supports the need for robust categorization and labeling mechanisms for SDKs, and the conclusion that behavioral analysis may be insufficient without additional out-of-band (non-code) information.

Use of Sensitive Signals. We manually identified 24 APIs which could be used to retrieve location data exactly or approximately and then checked how many of the likely fingerprinting behaviors in SDKs from the Extended Set rely in these APIs. We performed a similar analysis for app-usage signals (based on the three APIs we identified to provide information about the apps the user has installed on the device, the apps that are in use, or the usage statistics for installed apps) and for the account-list signals (based on two APIs to retrieve lists of personal accounts registered on the device). We find that of likely fingerprinting SDKs 72% collect coarse-grained location signals, 71.6% collect fine-grained location, and 86.29% collect at least one or the other. Only only 6.15% record account-list signals, and 38.46% collect app usage information.

4.3 RQ3: Market Reach

What kinds of apps use SDKs with fingerprinting behavior, and how prevalent are these SDKs in real-world apps?

To answer this question, we consider the SDK categories described in the RQ2 results, and the app categories assigned by the Google Play Store [22].

We are interested in understanding the presence of fingerprinting SDKs in the mobile-app marketplace. For this, we measured how many apps include fingerprinting SDKs in each app category, which categories of fingerprinting SDK are in most use, and which fingerprinting SDKs co-occur most often in apps. The first measurement seeks to determine whether there are app categories with particularly high prevalence of fingerprinting and thus that should be prioritized for any fingerprinting-reduction intervention. The second and third measurements inform any technical efforts to replace fingerprinting-based solutions with privacy-preserving alternatives.

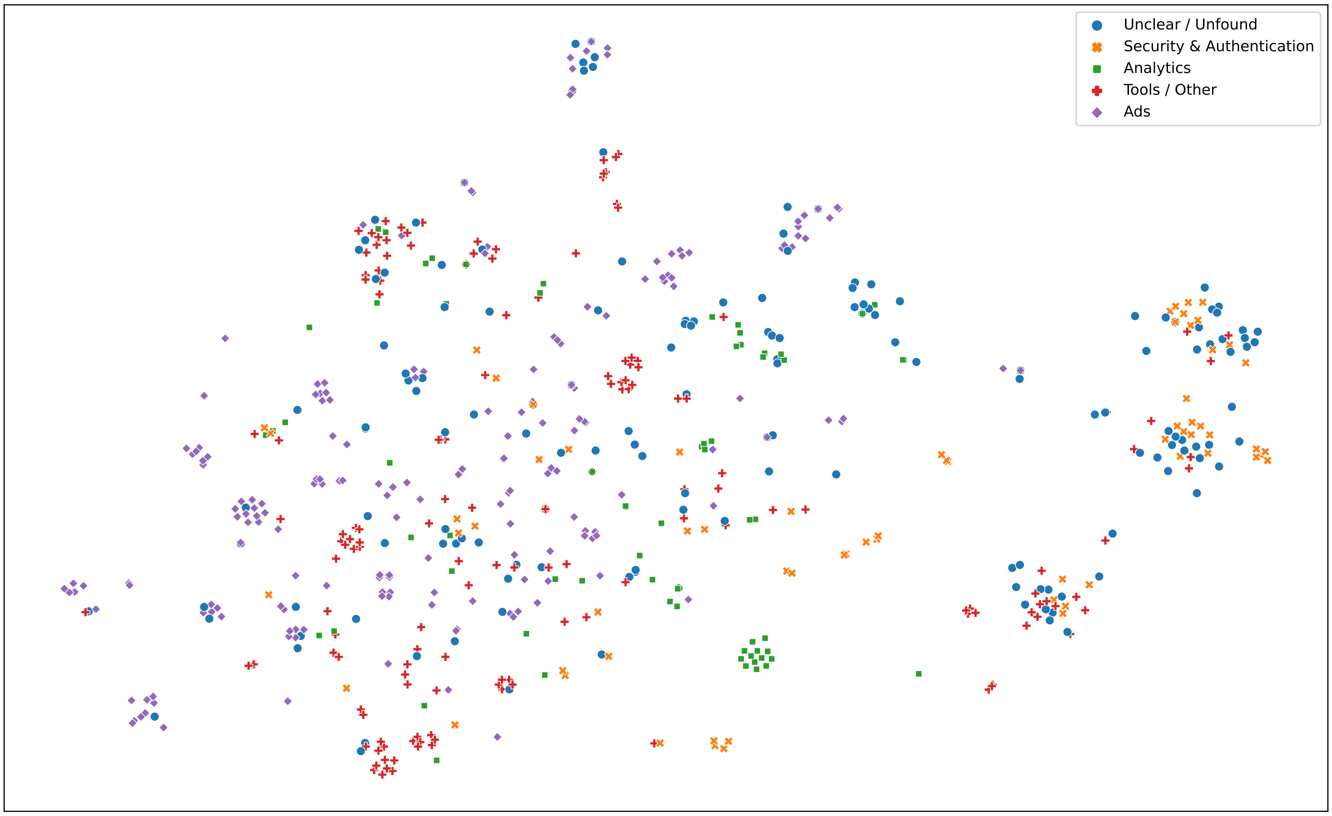


Figure 5: Map of likely fingerprinting behaviors of SDKs in the Extended Set, computed using t-SNE over embeddings constructed by one-hot encoding the exfiltrated APIs. Proximity of SDKs indicates that they exfiltrate data from similar sets of APIs.

Figure 6 shows the prevalence of apps that include fingerprinting SDKs in each app category. Figure 6a illustrates that the raw number of apps using fingerprinting SDKs averages at 3.2% across app categories, ranging between 0.8% (for "Events" apps) and 10% (for "Video Players" apps). Figure 6b takes into account the number of installs each such app had in the 30-day period, and shows that presence of fingerprinting functionality is heavily skewed towards popular apps.

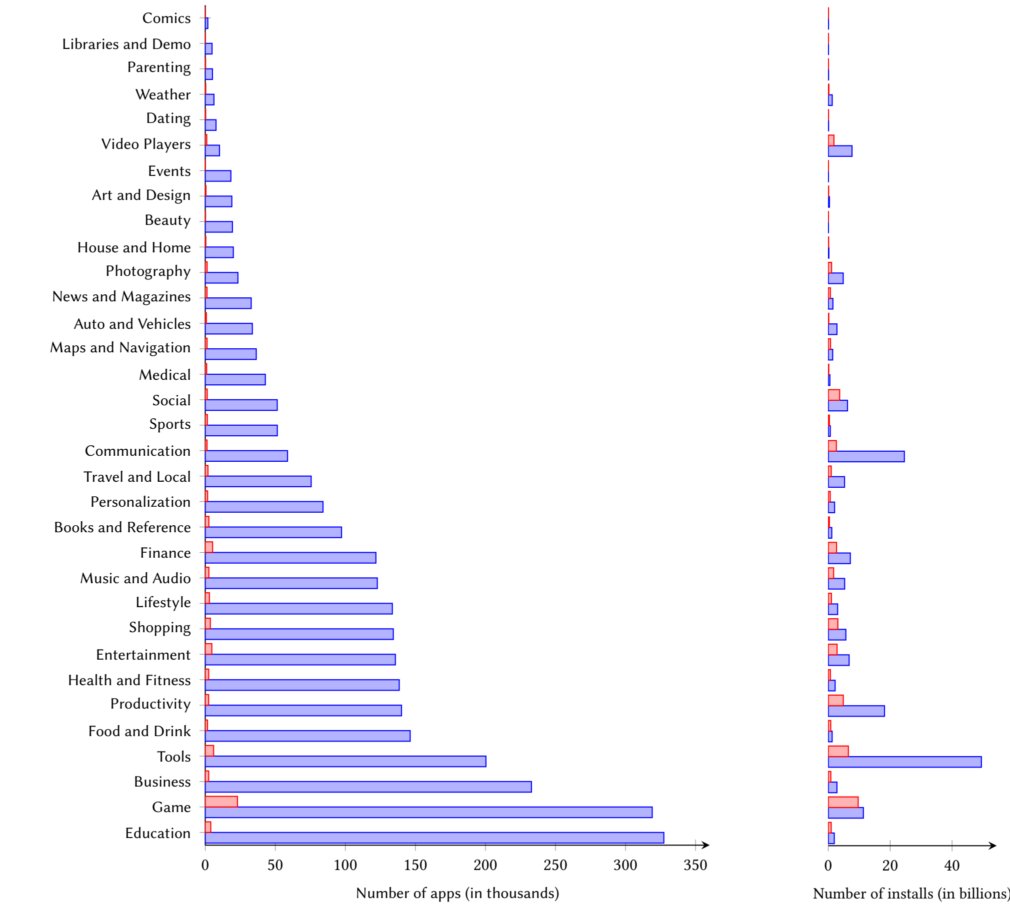
On average 39.4% of apps in a category contain at least one fingerprinting SDK, and while the minimum prevalence is 5.5% of apps (for the "Libraries and Demo" category), a number of app categories have >50% prevalence: "Maps and Navigation", "Beauty", "Shopping", "Sports", "House and Home", "Social", "Food and Drink", "Dating", "Game", and "Comics". From a user's point of view, this indicates that randomly installing a popular app has a 39.4% chance of being fingerprinted and, if they select a dating, game, or comics app, they will be fingerprinted with a 80+% probability.

The "Comics," "Game," and "Dating" categories stand out with the highest number of apps incorporating likely fingerprinting SDKs, respectively in this order. This could be attributed to several factors, such as the prevalence of free-to-use functionality (e.g., free-to-play games) that rely on targeted advertising or in-app purchases, or the need for (frictionless) user identification in online settings.

Fingerprinting Prevalence across App Categories. To further understand the prevalence of likely-fingerprinting SDKs in the mobile-app ecosystem, we use the purpose labels we developed in §4.2 to map app categories to SDK categories. This results in the heatmap shown in Figure 7a, in which a deeper shade of red indicates that the SDK category of that row dominates the app category of that column. For example, any likely-fingerprinting SDKs used by "Art and Design" apps come primarily from the "Ads" SDK category, while likely-fingerprinting SDKs in "Finance" apps are foremost from the "Analytics" SDK category.

The "Ads" SDKs dominate across almost all app categories as a source of likely fingerprinting behavior, with "Unclear / Unfound" SDKs as the second most common. We note that in many cases the absolute number of "Unclear / Unfound" SDKs is close to that of "Ads" SDKs (e.g., in the "Business" app category, the labels for 16,097 SDKs are unclear, and 17,780 SDKs have the "Ads" label) and thus any shift from "Unclear / Unfound" to "Ads" will only further cement the dominance of "Ads" SDKs a source of likely fingerprinting behavior.

A second observation from this heatmap is that there are several app categories ("Finance", "Food and Drink", "Shopping") where "Analytics" likely-fingerprinting SDKs are more prevalent that other categories of likely-fingerprinting SDKs. We hypothesize that in these app categories, the fingerprinting is less used to track user identities (which are known from the user-account information)



7.50%	87.76%
1.58%	5.58%
4.98%	38.26%
5.71%	15.78%
4.09%	82.57%
10.45%	23.91%
0.81%	42.98%
2.74%	33.00%
1.11%	52.45%
2.16%	56.51%
5.69%	22.27%
3.85%	45.40%
2.37%	6.14%
3.56%	51.07%
2.33%	31.66%
2.75%	59.25%
2.90%	56.08%
2.17%	10.63%
2.57%	18.28%
2.02%	31.08%
2.65%	31.68%
4.30%	37.48%
2.11%	32.80%
2.22%	33.74%
2.71%	54.44%
3.46%	42.21%
1.77%	31.27%
1.72%	26.71%
1.10%	63.80%
2.98%	13.12%
1.05%	29.18%
7.21%	85.15%
1.21%	48.77%

(a) Count of apps (blue) and apps with fingerprinting-like SDKs (red) by app category.

(b) Installs of apps (blue) and apps with fingerprinting SDKs (red), by app category.

(c) Percent of apps containing fingerprinting-like SDKs by total count (left) and install volume (right).

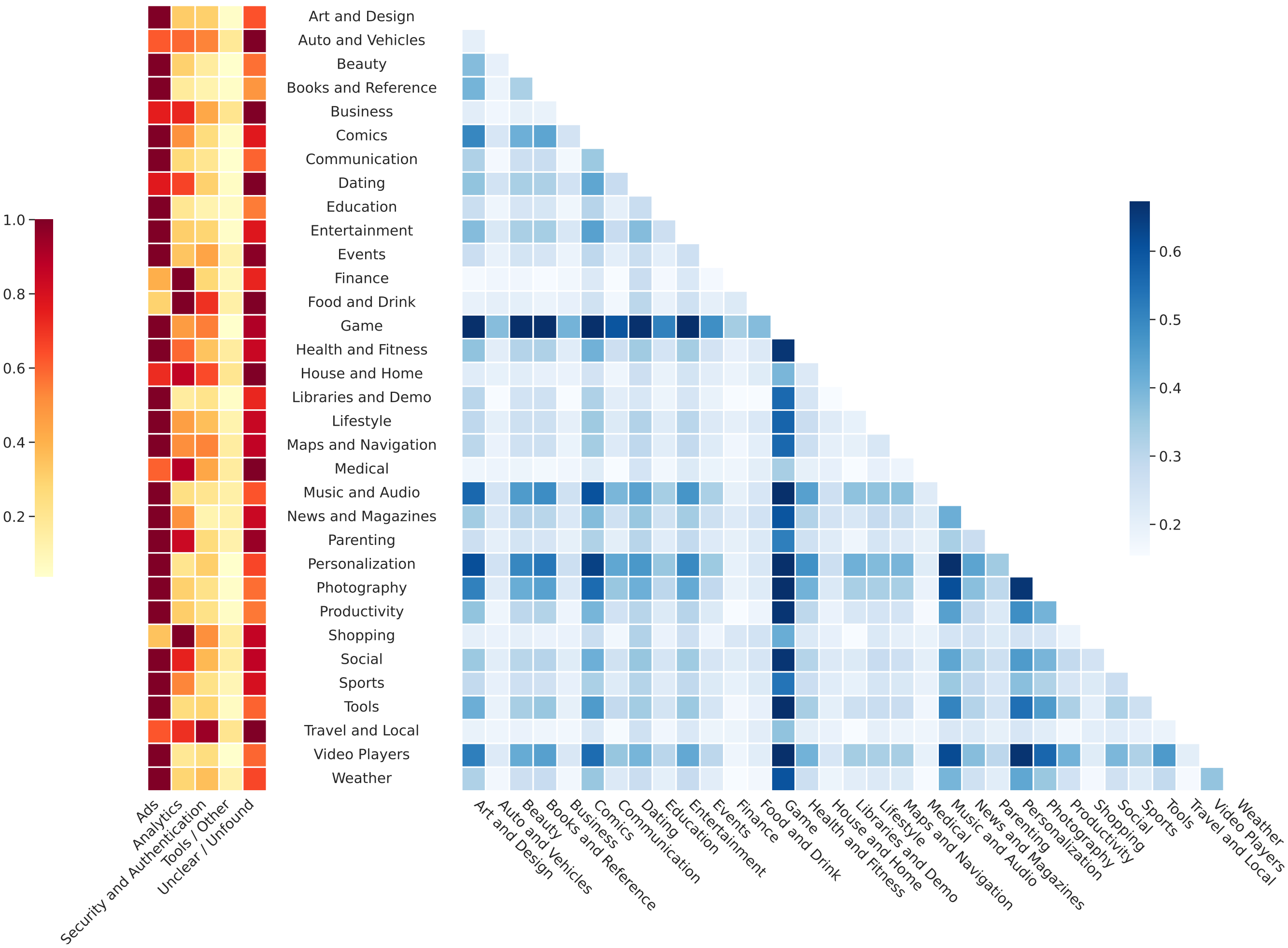
Figure 6: The number of apps that come with fingerprinting SDKs is rather small, on average at less than 5% of the total number of apps in a category (a), yet these apps are some of the most installed (b), giving likely fingerprinting SDKs an outsized presence in the market. In 23 app categories (highlighted in (c)), apps with fingerprinting SDKs are 10x more popular than other apps.

and more used to understand user preferences with respect to the items for sale in the app.

Possible Tracking across App Categories via Fingerprint Sharing. A significant privacy risk brought on by fingerprinting is the potential for a third party to track user activity across applications. This can happen when an SDK included in multiple applications fingerprints a user, and thus allows for user activity to be attributed to the same user for both. A service might then learn, say, that a user that engages in particular style of dating app, and also uses a specific medical or finance application. Such cross-app tracking may take place on device or on the server, in both cases powered by the fingerprinting data obtained from the shared SDK.

To estimate a *lower bound* on the risk of cross-app tracking, we analyze the prevalence of likely fingerprinting SDKs present in distinct categories by computing the probability that two apps randomly selected from each app category share at least one likely fingerprinting SDK. The results are shown in Figure 7b as a heatmap (only the lower diagonal presented, as the heatmap is symmetric). A darker shade of blue in the figure indicates a higher prevalence of shared fingerprinting SDKs, as shown, for example, by the ("Game", "Entertainment") entry compared with the ("Travel and Local", "Comics") entry. We compute these probabilities for the top-1000 apps by total audience size (as defined in §3.1) for each app category, regardless of whether those apps include a likely fingerprinting SDK or not. As a result, the prevalence (and the associated heatmap shown in Figure 7b) reflect both the popularity of apps and the distribution of likely fingerprinting SDKs in such popular apps.

Analysis of this heatmap clearly indicates that a few app categories have likely fingerprinting SDKs in common with many other



(a) Heatmap of the prevalence of likely fingerprinting SDKs in app categories (X-axis) and SDK categories (Y-axis).

(b) Heatmap of the co-occurrence of likely fingerprinting SDKs across pairs of app categories. Each cell describes the percentage of apps in the corresponding app categories (on the X-axis and Y-axis) that share one or more likely fingerprinting SDKs.

Figure 7: Prevalence of likely fingerprinting SDKs within and across app categories. In (a), the color gradient is computed per app category, allowing the comparison of SDK prevalence between app categories. In (b), the color gradient is computed per pair of app categories given by the X-axis and Y-axis coordinates. For raw data, see Appendix A.

app categories. For example, "Game" apps share likely fingerprinting SDKs with "Art and Design" apps, as well as with apps in the "Beauty", "Books and Reference", "Comics", "Communication", "Dating", "Entertainment", "Health and Fitness", "Libraries and Demo", "Lifestyle", "Maps and Navigation", "Music and Audio", "News and Magazines", "Personalization", "Photography", "Productivity", "Social", "Tools", "Video Players", and "Weather" categories. Similarly, "Personalization" apps share SDKs with 7 out of 33 app categories. From a user's point of view, this implies that they are at higher risk of cross-app tracking if they install apps from both the "Game" and "Comics" categories.

Alternatively, some categories of apps rarely share likely fingerprinting SDKs with other app categories. We highlight the "Finance" and "Medical" app categories in particular, as such apps often process highly sensitive data. Without further study, one cannot tell why fingerprinting is not more common here, and we note that such apps often require the user to authenticate to access their bank or investment account or their medical record and as such may not need to fingerprint the user through indirect signals.

5 Discussion & Limitations

Our results indicate that the fingerprinting ecosystem is more complex than previously estimated, both in terms of fingerprinting behavior and fingerprinting purpose. We interpret the results below, discuss limitations of our methodology, and present implications for app security mechanisms.

Challenges for Sector-Specific Solutions. A core observation of this paper is that the current mobile ecosystem has evolved to organically deploy fingerprinting-like behavior in a wide variety of apps through several types of SDKs. Our analysis reveals that while advertising SDKs contribute to fingerprinting, they are not the sole culprits: A significant portion of fingerprinting-like behavior originates from SDKs employed for analytics and anti-fraud purposes, and a large contingent (23.9%) did not have sufficient public information about their purpose or functionality to discern a category. These SDKs are often integrated for reasons directly outside of monetization — understanding user behavior to improve applications or preventing bots and fraud — despite ultimately collecting enough device information to create a fingerprint.

This finding challenges the prevailing notion that fingerprinting is primarily driven by app developer's need to monetize via advertising, and highlights the opportunity for a broader perspective on privacy-preserving alternatives. Research on how developers can be incentivized to adopt privacy preserving analytics, for example, could prove useful. It is also likely that sandboxing efforts (such as Android's Privacy Sandbox [1]) could provide additional benefit for both detection and enforcement against SDKs that over-collect — though systems research into lightweight sandboxing to support this setting is necessary, as scaling current process-based techniques to non-ads SDKs requires untenable overhead for constrained mobile environments.

Challenges for API-Specific or Other Behavioral Defenses. A surprising result of our exploration of the Seed Set (self-identified fingerprinting SDKs) was that the space of APIs used is sparse; the SDKs collected information from dissimilar sets of APIs. This held true regardless of the use-case of the SDK or of its prevalence in our dataset of applications. A potential explanation might be API Proxying [31], which may further complicate any anti-fingerprinting enforcement, though determining the joint entropy or shared entropy of particular API's is beyond the scope of this work.

In any case, it would appear that targeting specific APIs (akin to Apple's required reasons [4]) represents a brittle defense against fingerprinting. Developers have a number of signals at their disposal, and could easily move to other sources of entropy. Further, though we (surprisingly) found no evidence of non-API hardware-based fingerprinting in our Seed Set, one might expect developers to shift more advanced methods if comprehensive enforcement at the API level were introduced.

Potential for Sector-Specific Analysis & Targeting. It is worth noting that certain sensitive application verticals appeared to have an improved privacy stance. Normalized by install volume, only 30% of applications in the medical category used a fingerprinting SDK, and (assuming a normal distribution holds between sample sets) only 19.5% of *those* did so using an ads SDK — the bulk of identifiable fingerprinting behavior appears to come from analytics. Medical applications also appeared to have a lower potential for cross-application tracking. This heartening result, which is largely repeated in the Finance category, highlights the need for future work focusing on solutions for specific sensitive market verticals.

Need for Multi-Platform Analysis. It is likely that our results extend to the iOS ecosystem — indeed, all SDKs in our Seed Set appear to have versions readily available for iOS — a finding consistent with prior work on cross-platform tracking [33]. However, it is difficult to perform such analysis on iOS, as Apple's application and operating-system wide DRM restricts third parties' ability to scalably perform static and dynamic analysis on applications in their App Store. Future work studying the iOS ecosystem would provide invaluable insight into the effectiveness of design choices between the two operating systems.

5.1 Limitations

Any empirical study, including the present paper, is a limited view into real-world conditions and trends and thus it is important to evaluate the factors that threaten its validity. Following the "Campbell Tradition" [11], we consider four types of validity—internal, statistical, construct, and external—and their impact on this study.

Internal validity refers to whether the measured effect (fingerprintable APIs) truly corresponds to the outcome of interest (fingerprinting behavior). A risk is that the use of APIs to retrieve high-entropy data may not be caused by intentional fingerprinting behavior, but instead the result of necessary app functionality. We sidestep this by focusing on documenting the purposes of collection of fingerprintable data. A secondary limitation exists in the selection bias implicit in our Seed Set, which consists of SDKs that *self-identify* as fingerprinting. It may be that SDKs that fingerprint for hidden reasons use alternative techniques, which would not be caught in our later analyses. We assume that a Seed Set SDK's self-reporting is honest and make no further inferences about the SDK's intent.

Statistical validity refers to the risks of underpowered experiments, i.e., without sufficient statistical support. Our large sample size of 228,598 SDKs and 3,025,417 apps mitigates this risk.

Construct validity refers to the choice of metrics to measure the presence of fingerprintable APIs and behaviors. We focused on the number of APIs as an efficient metric of fingerprinting behavior, though we note that not all APIs are equally useful for fingerprinting. For now we make the simplifying assumption that in-the-wild techniques are largely equivalent, and that there is no relationship between signals collected. Using more complex metrics such as collision entropy [10] requires experiments across large sets of devices and users, which we leave for future work.

External validity refers to the generalizability of our results to real-world. Our choice of actual SDKs from popular Maven repositories and mobile apps from the Google Play store ensure minimize this risk. However, we do not attempt to catalog all fingerprinting mobile ecosystem, limiting ourselves to Java-language SDKs that are part of the Android/AOSP framework (excluding non-platform APIs or those from OEMs). It is possible that the Seed Set of fingerprinting SDKs, hand selected through web search, is not representative of all fingerprinting behaviors in the wild, and further study to ensure a comprehensive view is needed.

6 Conclusion

In this paper, we presented the largest-scale analysis of SDK behavior ever conducted, examining over 228,000 SDKs and 178,000 Android applications to understand the prevalence and purpose of fingerprinting-like behavior. Our findings reveal that a significant number of SDKs, beyond those explicitly designed for advertising, collect enough information to potentially track users. This includes SDKs used for analytics and anti-fraud, highlighting the need for privacy-preserving alternatives in these areas. Surprisingly, a large portion of SDKs exhibiting fingerprinting-like behavior lacked clear identification, emphasizing the need for greater transparency in the SDK ecosystem. Moreover, we observed that these SDKs with fingerprinting-like behavior are disproportionately popular and often integrated across diverse application categories. These results underscore the importance of ongoing efforts by Apple and Google to enhance user privacy and emphasize the need for continued research to ensure that such industry efforts are well directed.

References

[1] * * *. The Privacy Sandbox: Technology for a More Private Web. Available online at https://privacysandbox.com/intl/en_us. Last visited: 2024-09-30.

[2] Irene Amerini, Rudy Becarelli, Roberto Caldelli, Alessio Melani, and Moreno Niccolai. Smartphone fingerprinting combining features of on-board sensors. *IEEE Transactions on Information Forensics and Security*, 12(10), 2017.

[3] Apple. Describing data use in privacy manifests.

[4] Apple. Describing use of required reason api. Available online at https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/describing_use_of_required_reason_api. Last visited: 2024-09-30.

[5] Apple. User Privacy and Data Use - App Store. Available online at https://developer.apple.com/app-store/user-privacy-and-data-use/. Last visited: 2024-09-30.

[6] Michael Backes, Sven Bugiel, and Erik Derr. Reliable third-party library detection in android and its security applications. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 356–367, New York, NY, USA, 2016. Association for Computing Machinery.

[7] Pouneh Nikkhah Bahrami, Umar Iqbal, and Zubair Shafiq. Fp-radar: Longitudinal measurement and early detection of browser fingerprinting, 2021.

[8] Flavio Bertini, Rajesh Sharma, Andrea Iannì, and Danilo Montesi. Smartphone verification and user profiles linking across social networks by camera fingerprinting. In Joshua I. James and Frank Breitinger, editors, *Digital Forensics and Cyber Crime*, pages 176–186, Cham, 2015. Springer International Publishing.

[9] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile device identification via sensor fingerprinting, 2014.

[10] Gecia Bravo-Hermsdorff, Róbert Busa-Fekete, Mohammad Ghavamzadeh, Andres Muñoz Medina, and Umar Syed. Private and communication-efficient algorithms for entropy estimation, 2023.

[11] Donald T Campbell and Julian C Stanley. *Experimental and quasi-experimental designs for research*. Ravenio books, 2015.

[12] Yinzhi Cao, Song Li, and Erik Wijmans. (cross-)browser fingerprinting via os and hardware level features. In *Network and Distributed System Security Symposium*, 2017.

[13] Jing Chen, Yingying Fang, Kun He, and Ruiying Du. Charge-depleting of the batteries makes smartphones recognizable. In *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 33–40, 2017.

[14] Anupam Das, Nikita Borisov, and Matthew Caesar. Do you hear what I hear? fingerprinting smart devices through embedded acoustic components. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, New York, NY, USA, 2014. Association for Computing Machinery.

[15] Anupam Das, Nikita Borisov, and Matthew Caesar. Exploring ways to mitigate sensor-based smartphone fingerprinting, 2015.

[16] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*. The Internet Society, 2014.

[17] Antonios Dimitriadis, George Drosatos, and Pavlos S. Efraimidis. How much does a zero-permission android app know about us? In *Proceedings of the Third Central European Cybersecurity Conference*, CECC 2019, New York, NY, USA, 2019. Association for Computing Machinery.

[18] 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.

[19] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1388–1401. ACM, 2016.

[20] Amin FaizKhademi, Mohammad Zulkernine, and Komminist Weldemariam. Fpguard: Detection and prevention of browser fingerprinting. In Pierangela Samarati, editor, *Data and Applications Security and Privacy XXIX*, 2015.

[21] Christof Ferreira Torres and Hugo Jonker. Investigating fingerprinters and fingerprinting-alike behaviour of android applications. In *European Symposium on Research in Computer Security*, pages 60–80. Springer, 2018.

[22] Google. Play Console Help: Example categories. Online at https://support.google.com/googleplay/android-developer/answer/9859673.

[23] Google. Play console help: View app statistics. Online at https://support.google.com/googleplay/android-developer/answer/139628?hl=en&co=GENIE.Platform%3DAndroid.

[24] Google. Provide information for google play's data safety section.

[25] Google. SDK Runtime overview.

[26] Google. Google play sdk index. Online at https://play.google.com/sdks, 2024.

[27] Catherine Han, Irwin Reyes, Álvaro Feal, Joel Reardon, Primal Wijesekera, Narseo Vallina-Rodriguez, Amit Elazar, Kenneth A. Bamberger, and Serge Egelman. The price is (not) right: Comparing privacy in free and paid apps. *Proceedings on Privacy Enhancing Technologies*, 2020(3), 2020.

[28] Hongmu Han, Ruixuan Li, and Junwei Tang. Identify and inspect libraries in android applications. *Wirel. Pers. Commun.*, 103(1):491–503, nov 2018.

[29] Kris Heid, Vincent Andrae, and Jens Heider. Towards detecting device fingerprinting on ios with api function hooking. In *Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference*, EICC '23, New York, NY, USA, 2023. Association for Computing Machinery.

[30] Thomas Hupperich, Davide Maiorca, Marc Kührer, Thorsten Holz, and Giorgio Giacinto. On the robustness of mobile device fingerprinting: Can mobile users escape modern web-tracking mechanisms? In *Proceedings of the 31st Annual Computer Security Applications Conference*, ACSAC '15, 2015.

[31] Somesh Jha, Mihai Christodorescu, and Anh Pham. Formal analysis of the api proxy problem, 2023.

[32] T. Kohno, A. Broido, and K.C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[33] Konrad Kollnig, Anastasia Shuba, Reuben Binns, Max Van Kleek, and Nigel Shadbolt. Are iPhones Really Better for Privacy? A Comparative Study of iOS and Android Apps. *Proceedings on Privacy Enhancing Technologies*, 2022(2):6–24, 2022.

[34] Aleksandra Korolova and Vinod Sharma. Cross-app tracking via nearby bluetooth low energy devices. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, CODASPY '18, page 43–52, New York, NY, USA, 2018. Association for Computing Machinery.

[35] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix C Freiling. Fingerprinting mobile devices using personalized configurations. *Proc. Priv. Enhancing Technol.*, 2016(1):4–19, 2016.

[36] Menghao Li, Wei Wang, Pei Wang, Shuai Wang, Dinghao Wu, Jian Liu, Rui Xue, and Wei Huo. Libd: Scalable and precise third-party library detection in android markets. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 335–346, 2017.

[37] Tianyi Li, Xiaofeng Zheng, Kaiwen Shen, and Xinhui Han. Fpflow: Detect and prevent browser fingerprinting with dynamic taint analysis. In Wei Lu, Yuqing Zhang, Weiping Wen, Hanbing Yan, and Chao Li, editors, *Cyber Security*, pages 51–67, Singapore, 2022. Springer Nature Singapore.

[38] Xiang-Yang Li, Huiqi Liu, Lan Zhang, Zhenan Wu, Yaochen Xie, Ge Chen, Chunxiao Wan, and Zhongwei Liang. Finding the stars in the fireworks: Deep understanding of motion sensor fingerprint. *IEEE/ACM Transactions on Networking*, 27(5):1945–1958, 2019.

[39] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. Libradar: Fast and accurate detection of third-party libraries in android apps. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, page 653–656, New York, NY, USA, 2016. Association for Computing Machinery.

[40] René Mayrhofer, Jeffrey Vander Stoep, Chad Brubaker, Dianne Hackborn, Bram Bonné, Güliz Seray Tuncay, Roger Piqueras Jover, and Michael A. Specter. The android platform security model (2023), 2021.

[41] Gabi Nakibly, Gilad Shelef, and Shiran Yudilevich. Hardware fingerprinting using html5, 2015.

[42] Nick Nikiforakis, Wouter Joosen, and Benjamin Livshits. Privaricator: Deceiving fingerprinters with little white lies. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, 2015.

[43] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *2013 IEEE Symposium on Security and Privacy*, pages 541–555, May 2013.

[44] Łukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. The leaking battery. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Alessandro Aldini, Fabio Martinelli, and Neeraj Suri, editors, *Data Privacy Management, and Security Assurance*, pages 254–263, Cham, 2016. Springer International Publishing.

[45] Gerald Palfinger and Bernd Prünster. Androprint: Analysing the fingerprintability of the android api. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery.

[46] Erwin Quiring, Matthias Kirchner, and Konrad Rieck. On the security and applicability of fragile camera fingerprints. In *Computer Security – ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I*, page 450–470, Berlin, Heidelberg, 2019. Springer-Verlag.

[47] Jingjing Ren, Martina Lindorfer, Daniel J. Dubois, Ashwin Rao, David R. Choffnes, and Narseo Vallina-Rodriguez. Bug fixes, improvements, ... and privacy leaks - a longitudinal study of pii leaks across android app versions. In *Network and Distributed System Security Symposium*, 2018.

[48] Iskander Sanchez-Rola, Igor Santos, and Davide Balzarotti. Clock around the clock: Time-based device fingerprinting. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1502–1514, New York, NY, USA, 2018. Association for Computing Machinery.

[49] Michael A. Specter, Mihai Christodorescu, Abbie Farr, Bo Ma, Robin Lassonde, Xiaoyang Xu, Xiang Pan, Fengguo Wei, Saswat Anand, and Dave Kleidermacher. Fingerprinting sdks for mobile apps and where to find them: Understanding the market for device fingerprinting. Online at https://arxiv.org/abs/2506.22639, 2025.

[50] Raphael Spreitzer, Felix Kirchengast, Daniel Gruss, and Stefan Mangard. Procharvester: Fully automated analysis of procfs side-channel leaks on android. In *Proceedings of the 2018 on Asia Conference on Computer and Communications*

Security, ASIACCS '18, 2018.
[51] O. Starov and N. Nikiforakis. Xhound: Quantifying the fingerprintability of browser extensions. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 941–956, Los Alamitos, CA, USA, may 2017. IEEE Computer Society.
[52] Junze Tian, Jianyi Zhang, Xiuying Li, Changchun Zhou, Ruilong Wu, Yuchen Wang, and Shengyuan Huang. Mobile device fingerprint identification using gyroscope resonance. *IEEE Access*, 9:160855–160867, 2021.
[53] Güliz Seray Tuncay, Jingyu Qian, and Carl A. Gunter. See no evil: Phishing for permissions with false transparency. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 415–432. USENIX Association, August 2020.
[54] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.
[55] Tom Van Goethem, Wout Scheepers, Davy Preuveneers, and Wouter Joosen. Accelerometer-based device fingerprinting for multi-factor mobile authentication. In Juan Caballero, Eric Bodden, and Elias Athanasopoulos, editors, *Engineering Secure Software and Systems*, Cham, 2016.
[56] Yan Wang, Haowei Wu, Hailong Zhang, and Atanas Rountev. Orlis: Obfuscation-resilient library detection for android. In *2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2018.
[57] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.
[58] Wenjia Wu, Jianan Wu, Yanhao Wang, Zhen Ling, and Ming Yang. Efficient fingerprinting-based android device identification with zero-permission identifiers. *IEEE Access*, 4:8073–8083, 2016.
[59] Jian Xu and Qianting Yuan. Libroad: Rapid, online, and accurate detection of tpls on android. *IEEE Transactions on Mobile Computing*, 21(1):167–180, 2022.
[60] Xian Zhan, Lingling Fan, Sen Chen, Feng Wu, Tianming Liu, Xiapu Luo, and Yang Liu. Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications. In *Proceedings of the 43rd International Conference on Software Engineering*, ICSE '21. IEEE Press, 2021.
[61] Xian Zhan, Lingling Fan, Tianming Liu, Sen Chen, Li Li, Haoyu Wang, Yifei Xu, Xiapu Luo, and Yang Liu. Automated third-party library detection for android applications: Are we there yet? In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 919–930, 2020.
[62] Xian Zhan, Tianming Liu, Lingling Fan, Li Li, Sen Chen, Xiapu Luo, and Yang Liu. Research on third-party libraries in android apps: A taxonomy and systematic literature review. *IEEE Transactions on Software Engineering*, 48(10):4181–4213, 2022.
[63] Xian Zhan, Tianming Liu, Yepang Liu, Yang Liu, Li Li, Haoyu Wang, and Xiapu Luo. A systematic assessment on android third-party library detection tools. *IEEE Transactions on Software Engineering*, 48(11):4249–4273, 2022.
[64] Jiexin Zhang, Alastair R. Beresford, and Ian Sheret. Sensorid: Sensor calibration fingerprinting for smartphones. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 638–655, 2019.
[65] Jiexin Zhang, Alastair R. Beresford, and Ian Sheret. Factory calibration fingerprinting of sensors. *IEEE Transactions on Information Forensics and Security*, 16:1626–1639, 2021.
[66] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 429–440. Association for Computing Machinery, 2014.

A Data Tables for Fingerprinting Prevalence in SDK and App Categories

The following tables provide detailed data on our market measurements.

Table 3 presents the prevalence of likely fingerprinting SDKs across various app categories, broken down by SDK type. For instance, in the "Art and Design" app category, 43.3% of apps are likely to contain Ads SDKs that likely engage in fingerprinting. The data shows that "Ads" and "Unclear/Unfound" SDK categories generally have higher prevalence rates across most app categories compared to "Analytics," "Security and Authentication," and "Tools/Other" SDKs.

Tables 4 and 5 detail the proportion of apps within each category that contain SDKs also present in apps of other categories. The tables indicate that many apps utilize SDKs that are also prevalent in apps belonging to different categories. For instance, while 0.401 of "Books and Reference" apps share SDKs with "Art and Design" apps, only 0.095 of "Art and Design" apps share SDKs with the "Food and Drink" category, indicating a much lower overlap in SDK usage between these two specific app types.

Table 3: The prevalence of likely fingerprinting SDKs (by SDK category) in app categories.

	SDK Category				
App Category	Ads	Analytics	Sec. and Authn	Tools / Other	Unclear / Unfound
Art and Design	0.433	0.138	0.132	0.022	0.274
Auto and Vehicles	0.211	0.200	0.183	0.064	0.341
Beauty	0.484	0.146	0.080	0.014	0.277
Books and Reference	0.544	0.093	0.066	0.030	0.267
Business	0.241	0.234	0.138	0.067	0.320
Comics	0.387	0.193	0.099	0.024	0.297
Communication	0.475	0.127	0.096	0.018	0.284
Dating	0.275	0.237	0.108	0.022	0.358
Education	0.516	0.102	0.061	0.036	0.285
Entertainment	0.412	0.128	0.119	0.021	0.320
Events	0.346	0.118	0.154	0.044	0.338
Finance	0.163	0.397	0.111	0.038	0.292
Food and Drink	0.095	0.318	0.225	0.044	0.318
Game	0.339	0.160	0.185	0.010	0.307
Health and Fitness	0.340	0.200	0.117	0.055	0.287
House and Home	0.209	0.252	0.189	0.059	0.291
Libraries and Demo	0.463	0.075	0.100	0.025	0.338
Lifestyle	0.358	0.165	0.129	0.044	0.304
Maps and Navigation	0.325	0.167	0.174	0.051	0.282
Medical	0.195	0.288	0.140	0.052	0.324
Music and Audio	0.452	0.108	0.093	0.063	0.284
News and Magazines	0.386	0.193	0.045	0.052	0.324
Parenting	0.314	0.263	0.082	0.041	0.299
Personalization	0.451	0.094	0.140	0.017	0.298
Photography	0.464	0.140	0.103	0.024	0.268
Productivity	0.463	0.145	0.105	0.028	0.259
Shopping	0.121	0.347	0.177	0.056	0.298
Social	0.317	0.236	0.120	0.051	0.276
Sports	0.375	0.199	0.085	0.038	0.302
Tools	0.454	0.116	0.130	0.029	0.271
Travel and Local	0.179	0.203	0.272	0.059	0.287
Video Players	0.483	0.092	0.121	0.018	0.286
Weather	0.411	0.118	0.148	0.053	0.270

Full Version of the Paper

A full version of this paper is online at

https://arxiv.org/abs/2506.22639

The full version includes:

- Descriptions of the static analyses performed,
- A description of the SDK-identification algorithm,
- A definition of the codebook used to categorize SDKs, and
- A list of the APIs observed in fingerprinting-like behaviors.

Table 4: The sharing prevalence of likely fingerprinting SDKs across app categories [Part 1]. Part 2 is in Table 5.

	App Category																
App Category	Art and Design	Auto and Vehicles	Beauty	Books and Reference	Business	Comics	Communication	Dating	Education	Entertainment	Events	Finance	Food and Drink	Game	Health and Fitness	House and Home	Libraries and Demo
Art and Design	0.469	0.202	0.381	0.401	0.210	0.499	0.322	0.365	0.272	0.381	0.272	0.164	0.195	0.794	0.366	0.214	0.304
Auto and Vehicles	0.202	0.182	0.197	0.187	0.174	0.238	0.166	0.253	0.178	0.233	0.193	0.174	0.203	0.378	0.211	0.197	0.143
Beauty	0.381	0.197	0.332	0.327	0.200	0.410	0.269	0.331	0.243	0.330	0.249	0.174	0.203	0.677	0.313	0.211	0.253
Books and Reference	0.401	0.187	0.327	0.347	0.192	0.435	0.278	0.326	0.241	0.335	0.242	0.158	0.186	0.714	0.322	0.198	0.260
Business	0.210	0.174	0.200	0.192	0.171	0.251	0.169	0.254	0.177	0.236	0.187	0.173	0.198	0.403	0.214	0.189	0.145
Comics	0.499	0.238	0.410	0.435	0.251	0.577	0.352	0.432	0.309	0.443	0.296	0.226	0.256	0.847	0.407	0.251	0.322
Communication	0.322	0.166	0.269	0.278	0.169	0.352	0.228	0.278	0.203	0.279	0.206	0.145	0.172	0.598	0.266	0.178	0.210
Dating	0.365	0.253	0.331	0.326	0.254	0.432	0.278	0.419	0.277	0.381	0.274	0.274	0.302	0.669	0.344	0.267	0.237
Education	0.272	0.178	0.243	0.241	0.177	0.309	0.203	0.277	0.199	0.267	0.205	0.167	0.194	0.512	0.247	0.192	0.183
Entertainment	0.381	0.233	0.330	0.335	0.236	0.443	0.279	0.381	0.267	0.381	0.263	0.231	0.261	0.714	0.337	0.249	0.244
Events	0.272	0.193	0.249	0.242	0.187	0.296	0.206	0.274	0.205	0.263	0.227	0.165	0.201	0.483	0.251	0.210	0.192
Finance	0.164	0.174	0.174	0.158	0.173	0.226	0.145	0.274	0.167	0.231	0.165	0.210	0.225	0.338	0.197	0.185	0.108
Food and Drink	0.195	0.203	0.203	0.186	0.198	0.256	0.172	0.302	0.194	0.261	0.201	0.225	0.253	0.381	0.227	0.217	0.134
Game	0.794	0.378	0.677	0.714	0.403	0.847	0.598	0.669	0.512	0.714	0.483	0.338	0.381	0.994	0.659	0.394	0.557
Health and Fitness	0.366	0.211	0.313	0.322	0.214	0.407	0.266	0.344	0.247	0.337	0.251	0.197	0.227	0.659	0.320	0.227	0.243
House and Home	0.214	0.197	0.211	0.198	0.189	0.251	0.178	0.267	0.192	0.249	0.210	0.185	0.217	0.394	0.227	0.217	0.154
Libraries and Demo	0.304	0.143	0.253	0.260	0.145	0.322	0.210	0.237	0.183	0.244	0.192	0.108	0.134	0.557	0.243	0.154	0.212
Lifestyle	0.296	0.203	0.265	0.265	0.203	0.349	0.224	0.318	0.221	0.303	0.227	0.202	0.230	0.568	0.276	0.219	0.196
Maps and Navigation	0.301	0.190	0.261	0.265	0.188	0.338	0.223	0.295	0.212	0.287	0.220	0.174	0.205	0.560	0.268	0.203	0.200
Medical	0.177	0.182	0.183	0.168	0.174	0.217	0.153	0.247	0.172	0.225	0.188	0.178	0.208	0.334	0.200	0.198	0.130
Music and Audio	0.561	0.247	0.457	0.488	0.260	0.605	0.395	0.440	0.335	0.472	0.328	0.201	0.239	0.886	0.445	0.263	0.369
News and Magazines	0.341	0.232	0.309	0.306	0.231	0.381	0.260	0.355	0.257	0.339	0.265	0.222	0.254	0.599	0.318	0.251	0.237
Parenting	0.269	0.203	0.248	0.242	0.200	0.319	0.208	0.308	0.214	0.290	0.222	0.200	0.229	0.516	0.260	0.219	0.181
Personalization	0.610	0.254	0.499	0.530	0.269	0.640	0.430	0.465	0.355	0.494	0.350	0.197	0.237	0.912	0.479	0.268	0.408
Photography	0.511	0.218	0.417	0.442	0.231	0.553	0.356	0.410	0.299	0.423	0.291	0.191	0.220	0.838	0.404	0.228	0.332
Productivity	0.364	0.178	0.299	0.315	0.183	0.398	0.255	0.306	0.224	0.310	0.225	0.155	0.182	0.661	0.298	0.190	0.237
Shopping	0.202	0.190	0.203	0.190	0.190	0.273	0.172	0.318	0.189	0.267	0.182	0.235	0.253	0.416	0.229	0.199	0.128
Social	0.351	0.213	0.305	0.310	0.217	0.411	0.257	0.357	0.245	0.345	0.242	0.217	0.243	0.661	0.312	0.226	0.224
Sports	0.290	0.199	0.264	0.259	0.198	0.328	0.220	0.312	0.219	0.294	0.225	0.197	0.224	0.535	0.272	0.214	0.195
Tools	0.412	0.193	0.334	0.356	0.200	0.457	0.287	0.342	0.248	0.352	0.245	0.170	0.198	0.738	0.333	0.205	0.264
Travel and Local	0.192	0.177	0.190	0.179	0.172	0.237	0.160	0.261	0.174	0.231	0.184	0.185	0.211	0.366	0.208	0.190	0.134
Video Players	0.515	0.222	0.421	0.445	0.234	0.553	0.358	0.398	0.304	0.427	0.299	0.178	0.212	0.841	0.405	0.238	0.337
Weather	0.322	0.169	0.268	0.279	0.172	0.355	0.227	0.276	0.204	0.283	0.209	0.144	0.170	0.603	0.268	0.183	0.210

Table 5: The sharing prevalence of likely fingerprinting SDKs across app categories [Part 2]. Part 1 of this data is in Table 4.

	App Category														
App Category	Lifestyle	Maps and Navigation	Medical	Music and Audio	News and Magazines	Parenting	Personalization	Photography	Productivity	Shopping	Social	Sports	Tools	Travel and Local	Video Players
Art and Design	0.296	0.301	0.177	0.561	0.341	0.269	0.610	0.511	0.364	0.202	0.351	0.290	0.412	0.192	0.515
Auto and Vehicles	0.203	0.190	0.182	0.247	0.232	0.203	0.254	0.218	0.178	0.190	0.213	0.199	0.193	0.177	0.222
Beauty	0.265	0.261	0.183	0.457	0.309	0.248	0.499	0.417	0.299	0.203	0.305	0.264	0.334	0.190	0.421
Books and Reference	0.265	0.265	0.168	0.488	0.306	0.242	0.530	0.442	0.315	0.190	0.310	0.259	0.356	0.179	0.445
Business	0.203	0.188	0.174	0.260	0.231	0.200	0.269	0.231	0.183	0.190	0.217	0.198	0.200	0.172	0.234
Comics	0.349	0.338	0.217	0.605	0.381	0.319	0.640	0.553	0.398	0.273	0.411	0.328	0.457	0.237	0.553
Communication	0.224	0.223	0.153	0.395	0.260	0.208	0.430	0.356	0.255	0.172	0.257	0.220	0.287	0.160	0.358
Dating	0.318	0.295	0.247	0.440	0.355	0.308	0.465	0.410	0.306	0.318	0.357	0.312	0.342	0.261	0.398
Education	0.221	0.212	0.172	0.335	0.257	0.214	0.355	0.299	0.224	0.189	0.245	0.219	0.248	0.174	0.304
Entertainment	0.303	0.287	0.225	0.472	0.339	0.290	0.494	0.423	0.310	0.267	0.345	0.294	0.352	0.231	0.427
Events	0.227	0.220	0.188	0.328	0.265	0.222	0.350	0.291	0.225	0.182	0.242	0.225	0.245	0.184	0.299
Finance	0.202	0.174	0.178	0.201	0.222	0.200	0.197	0.191	0.155	0.235	0.217	0.197	0.170	0.185	0.178
Food and Drink	0.230	0.205	0.208	0.239	0.254	0.229	0.237	0.220	0.182	0.253	0.243	0.224	0.198	0.211	0.212
Game	0.568	0.560	0.334	0.886	0.599	0.516	0.912	0.838	0.661	0.416	0.661	0.535	0.738	0.366	0.841
Health and Fitness	0.276	0.268	0.200	0.445	0.318	0.260	0.479	0.404	0.298	0.229	0.312	0.272	0.333	0.208	0.405
House and Home	0.219	0.203	0.198	0.263	0.251	0.219	0.268	0.228	0.190	0.199	0.226	0.214	0.205	0.190	0.238
Libraries and Demo	0.196	0.200	0.130	0.369	0.237	0.181	0.408	0.332	0.237	0.128	0.224	0.195	0.264	0.134	0.337
Lifestyle	0.254	0.238	0.199	0.366	0.285	0.244	0.384	0.329	0.247	0.228	0.280	0.246	0.276	0.203	0.330
Maps and Navigation	0.238	0.232	0.182	0.369	0.273	0.227	0.396	0.331	0.247	0.201	0.263	0.233	0.275	0.186	0.333
Medical	0.199	0.182	0.187	0.218	0.225	0.200	0.217	0.191	0.162	0.193	0.204	0.194	0.174	0.179	0.196
Music and Audio	0.366	0.369	0.218	0.669	0.413	0.332	0.711	0.611	0.446	0.247	0.434	0.353	0.505	0.234	0.618
News and Magazines	0.285	0.273	0.225	0.413	0.339	0.276	0.438	0.373	0.287	0.251	0.312	0.288	0.313	0.229	0.378
Parenting	0.244	0.227	0.200	0.332	0.276	0.241	0.344	0.297	0.228	0.225	0.266	0.239	0.252	0.201	0.299
Personalization	0.384	0.396	0.217	0.711	0.438	0.344	0.765	0.659	0.485	0.250	0.457	0.373	0.545	0.239	0.663
Photography	0.329	0.331	0.191	0.611	0.373	0.297	0.659	0.567	0.403	0.239	0.396	0.320	0.458	0.212	0.564
Productivity	0.247	0.247	0.162	0.446	0.287	0.228	0.485	0.403	0.289	0.186	0.287	0.242	0.326	0.172	0.405
Shopping	0.228	0.201	0.193	0.247	0.251	0.225	0.250	0.239	0.186	0.275	0.252	0.224	0.205	0.207	0.217
Social	0.280	0.263	0.204	0.434	0.312	0.266	0.457	0.396	0.287	0.252	0.322	0.271	0.324	0.214	0.393
Sports	0.246	0.233	0.194	0.353	0.288	0.239	0.373	0.320	0.242	0.224	0.271	0.248	0.266	0.199	0.321
Tools	0.276	0.275	0.174	0.505	0.313	0.252	0.545	0.458	0.326	0.205	0.324	0.266	0.372	0.187	0.460
Travel and Local	0.203	0.186	0.179	0.234	0.229	0.201	0.239	0.212	0.172	0.207	0.214	0.199	0.187	0.181	0.209
Video Players	0.330	0.333	0.196	0.618	0.378	0.299	0.663	0.564	0.405	0.217	0.393	0.321	0.460	0.209	0.574
Weather	0.227	0.224	0.158	0.398	0.262	0.212	0.431	0.355	0.256	0.166	0.259	0.220	0.290	0.161	0.362