Mixed Signals: Analyzing Software Attribution Challenges in the Android Ecosystem

Kaspar Hageman, Álvaro Feal, Julien Gamba, Aniketh Girish, Jakob Bleier, Martina Lindorfer,
Juan Tapiador, Narseo Vallina-Rodriguez

Abstract—The ability to identify the author responsible for a given software object is critical for many research studies and for enhancing software transparency and accountability. However, as opposed to other application markets like iOS, attribution in the Android ecosystem is known to be hard. Prior research has leveraged market metadata and signing certificates to identify software authors without questioning the validity and accuracy of these attribution signals. However, Android app authors can, either intentionally or by mistake, hide their true identity due to: (1) the lack of policy enforcement by markets to ensure the accuracy and correctness of the information disclosed by developers in their market profiles during the app release process, and (2) the use of self-signed certificates for signing apps instead of certificates issued by trusted CAs.

In this paper, we perform the first empirical analysis of the availability, volatility and overall aptness of publicly available metadata for author attribution in Android app markets. To that end, we analyze a dataset of over 2.5 million market entries and apps extracted from five Android markets for over two years. Our results show that widely used attribution signals are often missing from market profiles and that they change over time. We also invalidate the general belief about the validity of signing certificates for author attribution. For instance, we find that apps from different authors share signing certificates due to the proliferation of app building frameworks and software factories. Finally, we introduce the concept of attribution graph and we apply it to evaluate the validity of existing attribution signals on the Google Play Store. Our results confirm that the lack of control over publicly available signals can confuse the attribution process.

Index Terms—Android, Attribution, Attribution graph, Mobile apps

◆

1 INTRODUCTION

Software *attribution* is the process of matching a piece of software to its author. This concept has gained attention in the research community as it is critical for software analysis, platform measurements, security and threat analysis, transparency, and regulatory enforcement [1]–[12]. Every software platform implements different attribution mechanisms. Windows relies on a Public Key Infrastructure (PKI) that provides authenticity guarantees about the organization offering the software through the use of X.509 certificates issued by trusted Certificate Authorities (CAs) [13]–[15]. In the case of iOS, apps must be signed with a developer certificate issued by Apple [16], [17]. These certificates are part of Apple's developer program, which involves the verification of developers' legal identity [16], [17].

Android implements laxer attribution schemes regardless of the app market. During the development and publication process of apps, developers can disclose attribution data both during the app signing process and on their market profile (*e.g.,* developer name, email and website) [18]. This information is *self-declared by the developer* and is not endorsed nor validated by a trusted authority. Even the cryptographic signing certificates can be self-signed [19]. While other software platforms also distribute software under potentially unverified, self-declared attribution data, their PKI ensures some form of control by the platform operator which is absent in the Android ecosystem. To complicate things further, the diversity of publication policies across Android app markets translates into a lack of a robust Android-wide attribution mechanisms [6], [20]. This state of affairs impedes external actors, such as researchers and regulators (and, possibly, the market operators themselves) from automatically studying developer practices, enhancing software accountability, or effectively detecting harmful, cloned and deceptive apps [1], [21]–[25]. As a result, end users are potential victims of impersonation attacks, such as repackaged malware [22] or phishing attacks [26], which may also have a negative impact on the revenue streams and reputation of legitimate developers.

Prior research has relied on self-declared data, such as the app certificate [3], [6], [10], [20], [22], [27]–[36], app name [3], [8], [36], [37], the package name [20], [32], [38], or market metadata [3], [5], [7], [8], [10], [36]–[40] for author attribution. In some cases, authors combined multiple signals hoping to increase their strength. However, none of these approaches have been sufficiently validated and the general attribution problem remains poorly understood by the community. This paper fills this knowledge gap in the Android ecosystem by assessing the validity of publicly available attribution signals, including app market metadata and signing certificates. Specifically, we answer the following research questions:

RQ1. How available and volatile are attribution signals on Android markets?

RQ2. How consistent are attribution signals within apps, within markets and across them?

RQ3. How is the Google Play Store affected by the lack of signal availability and consistency?

To address these questions, we follow an empirical approach using large-scale, real-world data, and discuss the implications of these measurements. First, we review the

Android app signing process and the policies defined by different markets to report app authors (§2). Our findings inform the definition of a set of attribution signals (§3) that we use to conduct a large-scale measurement on a dataset containing 2.5M sets of signals and 1.4M apps (the difference between both figures is due to the same app being distributed across different markets). Second, we gather app and market metadata (when available) from five Android app markets between December 2019 and October 2021 (§4). The resulting dataset constitutes the basis of our study, whose main contributions are:

- We empirically study the factors that impact on accurate author attribution at the app and market levels, both within and across Android markets (§5). We measure and demonstrate that the use of market metadata and signing certificates as attribution signals is unsound due to the inaccuracy, volatility, and incompleteness of the data (**RQ1**).
- We introduce the notion of an *attribution graph* (§6) to study signal consistency. By applying this concept to our datasets, we observe that attribution signals often conflict with each other both at the app-level (same author using different signals in different apps) and across market (same app using different signals in different markets). These conflict impede the accurate identification of app authors (**RQ2**).
- We conduct a case study of the Google Play Store (§7). We demonstrate that its app vetting process is unable to detect forged metadata during the publication process despite Google's strict publication policies. Our work reveals that (1) the belief that the signing certificate relates to a single company is invalid; and (2) even a combination of several signals is insufficient for sound attribution in the Google Play Store (**RQ3**).

We conclude with a discussion of the scientific, operational, and regulatory implications of our findings. We argue that the lack of supervision over Android apps' release and signing process not only hinders both software transparency and accountability, but also impedes developer-oriented measurement studies. We believe that effective platform control is necessary to solve the predicament of attribution on Android, *e.g.,* by moving away from self-signed certificates and introducing a trusted authority (§8).

Code and data. To foster reproducibility and further research, we provide both the source code of our crawler, and the app and market metadata [41].

2 BACKGROUND

Both the software development and the release processes implemented by every Android market involve multiple parties. We note that the **owner** or **author** entity which is accountable for the product can be different than the entity (or entities) that took part in its development —*i.e.,* the **developer**, which could be a software factory—; and the one releasing it on a market (*i.e.,* the **publisher**. Each of these stakeholders can leave their own fingerprint in the software and market presence, which can translate into incongruous signals that confuse the attribution process. Therefore, we define author attribution as follows:

Definition 1 (Attribution). Author attribution is the ability of a user to determine which company is behind a given app and is thus accountable for this product.

The following example illustrates the challenges behind Android app attribution. In the app "Punk Music Radio," the app's certificate is signed by "Andromo App" (a development framework) and the developer name on the Play Store is a company called "Yottabyte Enterprise Mobile." The package name of the app (`com.andromo.dev271569.app366038`) also points to Andromo. However, the privacy policy (http://turtlefarmboost.simplesite.com/421262547) is a broken website hosted on a domain unrelated to any of these two companies. Given these mixed and contradicting signals, how can we know which company is liable for this app? More generally, is it possible to automatically and accurately identify the authors of Android apps at market scale? Even though there are legitimate use cases for introducing ambiguous attribution signals, if this ambiguity is allowed without oversight, when it is not communicated to the end user properly, or it hinders the ability for accurate attribution by analysts, it potentially becomes problematic.

The remainder of this section explains how the development and signing process (§2.1), and the publication of the Android app through a market (§2.2) can influence software attribution.

2.1 Android App Signing Process

Android apps are distributed as Android Package (APK) files through app markets such as Google Play and alternative app stores like Huawei or Tencent. In order to provide *integrity* (*i.e.,* to prevent tampering of the content of the APK) and *authenticity* (*i.e.,* to prove the identity of the author), each package must be cryptographically signed [19]. Google's official policy states: "*the certificate associates the APK [. . .] to you and your corresponding private key. This helps Android ensure that any future updates to your app are authentic and come from the original author*" [42]. Therefore, the signing certificate supposedly plays a vital role as an indication of authorship.

Android's signature scheme has been revised over time. For the sake of backward compatibility, APKs may be signed using one or more signature schemes. Most crucially, signing certificates typically are self-signed (99% according to one study on over 1M apps conducted in 2014 [1]). Yet, in contrast to Windows [14] and macOS/iOS [16], [43], Android lacks a trusted authority that confirms the validity and veracity of the certificates.

The Google Play Store also introduced new features with a direct impact on the signing process and the ability to identify the entity accountable for an app. Since 2017, it offers *"Play App Signing"* as a way to protect signing keys from being lost or compromised by allowing app authors to delegate the key management and signing process to the market. This service further constrains the already limited function of the signing certificate as an indicator of authorship, since these apps might be signed by Google itself and not by the actual app author. In fact, since August 2021, Google requires all new apps to be published as app bundles, which are essentially built and optimized by the Play Store and,

TABLE 1: Publishing policies for the 6 studied markets. Note that we could not find Baidu's policy.

	Google Play	APKMonk	Tencent	Baidu	APKMirror
Publisher info	✓	✓	✓	—	✓
Anti-malware	✓			—	
Anti-clones	✓			—	✓
False identity	✓			—	✓
Reference to policy	[49]	[50]	[51]	—	[52]

consequently, forces this signature delegation [44]. In this case, authors can use self-signed certificates for "*inspection by developers and end users, who want to ensure that code they're running matches the code that was originally built and signed by the app developer*" [45].

Alternative Android markets implement their own (typically less restrictive) policies. For example, developers should upload an already-signed APK to APKMonk [46], Baidu [47] or APKMirror [48].

2.2 Market Metadata & Policies

Once signed, authors can publish their apps on one or several markets. Most markets, with the exception of Baidu,[1] use the app package name for indexing apps. Google's documentation stresses the importance of the package name as an attribution signal and recommends following the Java package naming convention (*i.e.*, to "*use Internet domain ownership as the basis for package names (in reverse), to avoid conflicts with other developers*" [53]).

App markets allow authors to disclose data about their app (*e.g.*, its description and category) and themselves (*e.g.*, name and contact information) in their market profiles. However, publishing policies and profile metadata are not consistent across markets as shown in Table 1. Furthermore, the markets' terms of services (ToS) might set policies that can influence author attribution. Specifically, they often contain explicit policies to prohibit impersonating other authors or distributing malicious software and clones. For example, Google's Developer Program Policies [54] prohibit misrepresentation and impersonation of other apps and developers [55], [56]. These restrictions also apply to the market metadata, such as the developer name, title and screenshots. However, it is unclear if markets enforce these policies so the accuracy of this data depends on the authors' will to honor the platforms' best practices and guidelines. Out of the markets that we study, only Tencent requires proof of identification for registration, but only for Chinese citizens, residents, or companies. We do not consider F-Droid in our study as it does not impose any requirements on authors. Because it downloads the source, F-Droid builds, and signs the apps itself so several apps exist with only a link to the code repository listed with no further metadata about the author.

3 ATTRIBUTION SIGNALS

A thorough analysis of (i) Google's official documentation; (ii) prior research in this area (see 3.1); and (iii) the app

1. Apps with the same package name can be published in a game and non-game category.

TABLE 2: Attribution signals available per market. The symbols denote the origin of the signal: the app's manifest (📄); the app's signing certificate (🏅); market metadata (🛒).

	Origin	# Markets	Google Play	APKMonk	Tencent	Baidu	APKMirror
Package name	🛒 + 📄	6	✓	✓	✓		✓
App name	🛒 + 📄	6	✓	✓	✓	✓	✓
Developer name	🛒 + 🏅	5	✓	✓	✓		✓
Developer website	🛒	2	✓				
Privacy policy URL	🛒	1	✓				
Developer email	🛒	2	✓				
Developer address	🛒	1	✓				

signing and release policies across five Android markets, allowed us to identify four potential attribution signals: (a) the package name, (b) the app name, (c) developer details from market profiles; and (d) the signing certificate. However, the availability of these signals across markets is not consistent as we show in Table 2, with the exception of the signing certificate, which is mandatory for any Android app.

Package name. The package name serves as the unique (string) identifier of an app for the Android OS. All markets use the package name as a unique and visible identifier, or as an internal identifier, meaning that its value has to be unique per app within each market, and should follow Android's naming convention described in §2. One example of an app correctly following this naming convention is Facebook (`com.facebook.katana`). However, since this convention is not enforced, many apps introduce their own naming schemes as in the case of those built with Andromo's development platform (*e.g.*, `{com,net}.andromo.dev<dev_id>.app<app_id>`).

App name. In some cases, the app name can be considered an attribution signal itself (*e.g.*, Facebook). Howeer, it is not very robust as this field might neither be directly connected with its author nor be unique (*e.g.*, consider generic names such as "Music Player"). Despite its weakness, we include this signal in our analysis to assess its validity and consistency since it can be extracted either from the apps' market profile or from their manifest file.

Developer details. Authors publishing apps on Android app markets are identified by a *developer name* and, depending on the market, they can also disclose contact information such as an *email address*, a *website*, or their *physical address*. Additionally, they can provide a *privacy policy URL* which must contain legal and contact information of the author as required by current legislation [57]–[59]. Therefore, the value of market developer data for attribution purposes depends on the accuracy of the data disclosed by the author. Moreover, its validity varies depending on the intended usage—specifically, on whether it is used to attribute two apps *published on the same market (i.e., intra-market attribution)*, or *across markets (i.e., inter-market attribution)* to the same author.

Signing certificate. Since the private key associated with the certificate is supposedly kept secret by its owner, there is a general belief that two apps signed by the same certificate belong to the same author. Each X.509 certificate contains a *subject* field, which indicates the *owner* of the certificate, and an *issuer*, which serves as an indication of the entity

that provided the certificate to the owner. However, the subject and issuer field in self-signed certificates are the same and this information is filled in arbitrarily by the party creating the certificate. In addition, for apps that delegate their signing process (see §2.1), all certificates generated by each platform share the same subject field.

3.1 Attribution Challenges in Prior Research

Researchers have used the signing certificate, market metadata or a combination of both for attribution. However, no prior study has analyzed the pitfalls of such attribution techniques and how their accuracy could compromise the interpretability and validity of results. Only the limitations of signature-based attribution have been reported in prior work. An empirical analysis by Barrera *et al.* [27] on the security aspects of Android app sandboxing relied on certificates and app information as a proxy for authorship. They showed that malicious actors often create multiple self-signed certificates to prevent link-ability. Similarly, Oltrogge *et al.* [32] showed that app building frameworks, which automate app development and distribution, invalidate the assumption that apps with the same certificate belong to the same company.

A particularly relevant line of research are market-level measurements and cross-market comparative analysis [8]–[12], [39]. All of these studies relied on attribution signals such as the developer's name to link app authors across markets. Wang *et al.* [5]–[7] used market-level metadata and certificate information to track authors across Chinese app markets and the Google Play Store. More recent efforts analyzed the Android supply chain and firmware-level customizations, also faced attribution challenges [28], [35], [60]. Specifically, Gamba *et al.* reported that many preloaded apps are signed with Debug certificates (thus violating Google policies) or with vague subjects such as "Android," thus impeding attribution [35].

The security and privacy community has also used app-level information for attribution purposes [20], [29], [61]–[63]. A crucial aspect in security research is the ability to responsibly disclose vulnerabilities. As most authors often rely on market metadata (*i.e.,* developer address or privacy policies) to contact app developers [40], [64], the correctness of developer information is critical for accountability. However, due to the inability to access accurate contact information easily, researchers often opt to disclose their findings to market operators like Google instead [65], [66]. In other cases, researchers disclose a vulnerability in Android itself to Google directly [67] and could use developer information to inform affected app developers.

4 DATASET

We use a custom-built Scrapy crawler [68] to download APKs and their corresponding app metadata (when available) from five markets (see Table 2): Google Play Store, APKMonk [69], APKMirror (an aggregator of apps published in other markets [70]), and two markets with a large user base in Asia: Baidu [71] and Tencent [72]. We refer to the full set of signals extracted from each app listing in a market as a *market entry*.
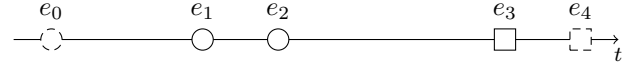

Fig. 1: Timeline of the collected market entries for a given package name and market ($\circ$ = first crawl, $\square$ = second crawl). The longitudinal analysis investigates the differences between e_0 and e_4, whereas the other analyses consider e_4 only.

Crawling strategy.

We kickstarted each market crawl with a seed of all unique package names from AndroZoo [73], totaling about 5M package names. Since AndroZoo does not link APK files with the associated market metadata, we crawled the apps and metadata separately. We opportunistically explore the six markets by following links to similar or recommended apps shown in a given app's profile page. Similar crawling strategies have proven effective in prior work [6], [74]. We deployed the crawler on two separate occasions (using the same 5M apps as a seed) between December 2019 and October 2021 to (1) discover newly published apps, and (2) collect new version of market entries for a subset of apps. We extended our crawler between the two crawling windows to collect the developer address from Google Play. We relied on twelve HTTP proxies located in two EU countries to parallelize the crawling.

Dataset statistics. Table 3 provides an overview of our dataset. The first crawl represents a snapshot of apps collected between December 2019 and May 2021. The second crawl represents the snapshot of apps collected between June 2021 and October 2021. These two snapshots give us a longitudinal perspective of attribution challenges in Android markets. Overall, we crawled metadata for 1.36M different package names across all markets and 1.45M unique APK files (as identified by their SHA-256 file hash), covered by a total of 2.49M market entries. We cannot evaluate the coverage of our crawler due to the inability to know the markets' full catalogue. The discrepancy between the number of market entries and APKs crawled across markets can be attributed to artifacts such as rate limiting, timeouts, geo-restrictions, and other errors when downloading the APKs. Even though we used multiple IP addresses for the crawler, we were unable to crawl APKMirror at scale, as it lies behind CloudFlare's DDoS protection system. Still, for Google Play, we collected 1.40M market entries with 804k unique apps out of the estimated 2.65M available at the time of writing [75].

Longitudinal analysis. Performing a longitudinal analysis requires us to handle both different versions of apps and changes in their market profiles. Table 3 reports the overlap of package names across markets for our two crawling campaigns. The overlap varies from market to market due to app removals and our best-effort crawling strategy: we note that some packages were unlisted from the stores between crawls. Still, the second crawls for APKMonk and Google Play contain 19% and 27% of the apps collected in the first one, respectively. While apps and market metadata are frequently updated, these changes might occur at different times, *i.e.,* not every market listing for a given package name results in a different APK file. This phenomenon explains

TABLE 3: Dataset overview. The overlap column reports the number of unique package names (PkgN) collected in both crawls.

Market	First crawl (December 2019 – May 2021)			Second crawl (June 2021 – October 2021)			PkgN Overlap (%)
	Market entries	PkgN	APK SHA-256	Market entries	PkgN	APK SHA-256	
Google Play	903,385	661,421	66,2019	540,361	434,884	440,060	150,731 (18.7%)
APKMonk	324,054	298,133	300,341	500,870	293,351	293,469	153,472 (26.5%)
Tencent	182,492	126,994	127,354	20,319	4,379	4,049	368 (0.3%)
Baidu	5,235	4,304	3,881	13,907	2,533	2,536	38 (0.5%)
APKMirror	752	716	722	2,846	1,113	1,113	6 (0.3%)

the mismatch between app package names and APKs in Table 3. In order to conduct our longitudinal analysis in such a variable scenario (§5 and §7), we use the latest collected market entry except for our longitudinal analysis (§5.2), in which we consider the earliest entry from the first crawl and the latest from the second for each app (see Figure 1).

5 ANALYSIS OF ATTRIBUTION SIGNALS

Each Android app market defines its own policies and enforcement mechanisms for publishing apps (§2). Yet, all attribution signals are self-declared by the author or developer of the software, thus they could be missing or misleading. Moreover, attribution data is not necessarily persistent. The Android ecosystem is highly dynamic with frequent company acquisitions, re-brandings, and new apps and versions being released regularly [63], [76], as well as short-lived impersonifications (App-Squatting) [77]. As a result, attribution data might not be consistently updated, thus leading to confusion. In this section, we measure the *availability and volatility* of attribution signals across markets in order to reason about their validity and coverage, answering **RQ1**.

5.1 Signal Availability

As a first step, we empirically measure to what extent individual attribution signals are missing on apps across markets. This preliminary analysis is critical to assess the enforcement of market-specific publication policies and then to reason about their individual validity. Later in §6.2–§7, we analyze combinations of signal values and how the soundness of such approaches is negatively impacted when markets fail to provide signal values for published apps.

For this analysis, we consider the latest version for each app (per market listing) in our dataset. Table 4 shows the percentage of unique package names for which there are missing signals, across all markets in our entire dataset. For market entries from Google Play, several signals were not collected for the full data collection period and we report (§4) those results as a percentage of packages collected after we started collecting them.

Market metadata. Nearly all market entries collected across the markets are published under a developer name. Of the eight app entries on Google Play without a developer name, only one (`br.com.mksolutions.mksac.redebr`) remains listed at the time of writing. More significant, however, is the extent to which developer websites (34%), developer addresses (44%), and privacy policy URLs (18%) are missing from Google Play profiles, and the extent of

TABLE 4: Percentage of unique market entries throughout the dataset with missing attribution signals on the different markets (— indicates we did not collect a specific signal).

	Attribution Signal	Google Play *	APKMonk	Tencent	Baidu	APKMirror
Market	Developer name	<0.01%	<0.1%	<0.01%	—	0%
	Developer website	33.7%	—	—	—	—
	Developer email	<0.01%	—	—	—	—
	Developer address	44.4%	—	—	—	—
	Privacy policy URL	17.8%	—	—	—	—
Cert RDN	commonName	7.1%	9%	9.1%	6.5%	11.9%
	organization	17.6%	25.4%	21.1%	20.9%	8.5%
	org.Unit	27.6%	36.7%	30.9%	21.9%	23.5%
	locality	27.2%	35.9%	29.3%	20.9%	16.9%
	state	28.8%	38.8%	32.6%	24.3%	20.3%
	country	21.8%	30.3%	27%	24%	15.4%

*Developer name, address and email as % of markets entries crawled after 2021/09/21

signal absence on the rest of the markets. The contact information is clearly not seen as a necessity for app publication on the Play Store, neither by Google nor the developers.

Signing certificates. When looking at the signing certificates' subject, we find that relative distinguished name (RDN) components are missing for a significant fraction of unique market entries, ranging from 7% for the common names on Google Play to 39% of the state missing on APKMonk. In Android, the RDNs do not serve any purpose for app developers, hence omitting them from their signing certificates comes with no drawbacks. The lack of oversight on the self-signed certificates results in the inconsistent availability of these RDNs.

5.2 Signal Volatility

We measure volatility of market metadata and signing certificates by monitoring changes across the two dataset snapshots as described in §4: we select the earliest entry from the first snapshot and its latest entry from the second snapshot (see Figure 1). Overall, we observe that the coverage of package names varies widely across markets, ranging from less than 1% for Tencent, Baidu and APKMirror to 27% for APKMonk. We report the results for the three low-coverage markets, but note that the results are not representative for the market as a whole. Table 5 shows the results for the signals collected in both crawls.

Market metadata. All markets allow app and developer names of published apps to change: the app name volatility ranges from 0.9% of package names on both APKMonk to 7.9% of package names on Baidu. The volatility of the developer name ranges from 0.6% on APKMonk to 18.8%

on Tencent. For the signals captured on Google Play only (*i.e.,* the developer website, email address, and privacy policy URL) we see a volatility of 1.7%, 1.0%, and 1.6%, respectively.

Notable cases of highly volatile signals include the app 'Electronic Dance Music Radio' and the developer 'Cube Apps Ltd' who underwent six and four name changes over the course of our measurement setup respectively. Note that these results do not include cases in which signal values differed during the measurement period but ended up with their original values. A manual inspection suggests that such cases occur only sporadically and nearly all of these identified cases consist of a single change that was reverted.

Changing the information under which an app is published has its legitimate use cases, but our results suggest that volatility should be considered as a potential issue in research studies. Conclusions drawn from signals at a certain point in time may not hold at a later stage for a given body of package names.

Signing certificates. Signing certificates are volatile too: 58% of all apps in our dataset are signed with multiple certificate schemes (such as v1, v2, v3). Specifically, 23.1% of the apps are signed with v3 certificates, hence allowing certificate rotation (the number rises to 29.1% on Google Play). The certificate percentages in Table 5 represent the fraction of package names that added a new certificate to the APK or completely changed certificates across their market entries. The latter can contain re-published apps by either their original or by a completely different author. The self-signed nature of the signature does not allow us to differentiate between these two cases due to the lack of a ground truth. For example, our dataset contains two market entries for package name "`ua.iread.android`" on Google Play, published under a different developer name ("*<pineconeapps>*" and "*xl-games*") and different app names ("*Learn to read for kids free*" and its seemingly Ukranian translation). The different developer names do not provide any confidence that both apps were published by the same organization.

We also find apps from different authors that have a signature that suggests the delagation of the signing process to Google through "Play App Signing". However, the lack of ground truth prevents us from certifying whether the app was signed by the Play Store or the publisher just created a signature with the same subject information as the Play Store's. We find that this happens in all markets, ranging from less than 1% of package files on Baidu to 16.0% on APKMonk and 32.4% on Google Play itself. These exclude apps signed by certificates with a similar subject to the Google Play's default subject field (*e.g.,* the developer 'Yippity Doo LLC' publishing an app using a certificate with "*Google*" and "*Cupid*" as the subject organization and common name, instead of "*Google Inc.*" as the common name used by the Play Store for their certificates), which are apps published under bogus certificates. Without analyzing additional signals extracted from the APK, this ambiguity affects the soundness of studies into signing delegation.

RQ1: How available and volatile are attribution signals on Android markets? Our empirical results highlight that the developer contact signals are relatively unreliable for

TABLE 5: Percentage of package names per market for which we observe a change in signals over time. App coverage reports the percentage of apps present in both crawls.

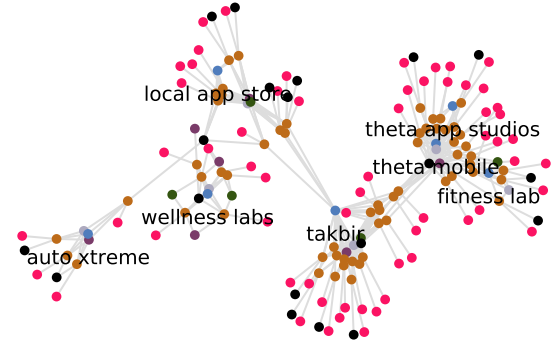|	Attribution Signal	Google Play	APKMonk	Tencent	Baidu	APKMirror
Market	App name	2.5%	0.9%	6.8%	7.9%	0%
	Developer name	3.9%	0.6%	18.8%	—	16.7%
	Developer website	1.7%	—	—	—	—
	Developer email	1.0%	—	—	—	—
	Privacy policy URL	1.6%	—	—	—	—
Cert	Added	0%	0%	0.3%	0%	16.7%
	Fully replaced	<0.01%	<0.01%	2.7%	0%	0%

Fig. 2: A cluster from the Google Play attribution graph annotated with developer names (● = market listing, ● = developer name, ● = developer website, ● = developer email, ● = privacy policy URL, ● = app name, ● = certificate).

attribution, either because they are not available at all or, in the case of Google Play, often missing. The developer name – besides the app name and package name – is essentially the one reliably available signal in app markets. The lack of imposed and enforced restrictions on the (self-signed) certificates is visible in the fraction of RDNs that are missing across signing certificates. Furthermore, whereas certificate remain relatively stable over time, market signals tend to be more volatile.

6 SIGNAL CONSISTENCY

Beside the availability and volatility of individual signals, their self-declared nature and lack of market enforcement directly affects their consistency. The signals associated with an app published by a given developer in a given market (and across markets) may conflict with one another, thus potentially impacting attribution efforts. By evaluating these conflicts, we answer **RQ2** in this section.

6.1 Attribution graph

We introduce the novel concept of *attribution graph* to study and measure signal consistency at the app level (§6.2), within markets (§6.3), and across markets (§6.4).

Definition 2 (Attribution Graph). We define an attribution graph as a bipartite graph $G = (S, A, E)$ in which:

- S is the set of market entries, *i.e.,* the set of all the different signals that can be extracted for each market listing (as described in § 3). These are uniquely identified by the lowercase value of most signals (*i.e.,* developer name, website and email, the privacy policy URL, package name and the app name) and by the SHA256 digest of the signing certificates.
- A is the set of all market listings, *i.e.,* every unique collection of metadata for a given app and market.
- $E = \{(x,y) \mid x \in S \land y \in A\}$ is the set of edges that connect signals with the market entries in which they are found.

This attribution graph allows extracting and measuring the relationship between the different signals and market listings, and their consistency by extension, by analyzing different subgraphs. Figure 2 illustrates this concept and its application to study signal consistency by showing a cluster of apps published on Google Play apps.

6.2 Signal Consistency Within Apps

Due to the fact that attribution signals can be potentially introduced by different actors at different stages of the publication process (§3), it is possible that this information is not consistently introduced or updated for a given app. One such examples is the app name, which is present both in the market and in the APK file. We measure the prevalence of such inconsistencies by comparing the app name disclosed in the market metadata with the information present in the manifest file for every market listing of a given app (per package name). We represent the market listing and its two app names in our attribution graph as nodes, and compare the label of the app name nodes.

App authors might use other alphabets, leading to potential mismatches if not handled correctly. Therefore, to avoid bloating our results, we discard pairs of market and app data where one of the two signals is not in the Latin alphabet. For the remainder of apps, we observe that only in 45.3% of the cases the name is exactly the same. All markets suffer from these inconsistencies – ranging from 28.9% and 46.0% on APKMirror and Tencent respectively – which hints towards poor software maintenance practises and lack of enforcement by market operators.

Such signal inconsistencies complicate the comparison of research studies and attribution efforts at the app-name level. In fact, they make external analysis of these apps harder as one has to decide whether to rely on market metadata or the app's manifest (*e.g.,* when researchers make their results available to the public). Furthermore, the possibility to create an inconsistency between the market and the installed app can be abused by malware and phishing attacks [78]. Users might be tricked into downloading a potentially privacy-intrusive app that shares the name with a well-known app [79], [80].

For the apps for which we do not find an exact match, we measure how similar both names are. To do so, we rely on the normalized Levenshtein similarity [81]. We find that around 17.4% of the apps have a similarity below 50%. We observe cases in which one name is a substring of the other (*e.g.,* "*racing lap timer & stopwatch*"—"*laptimer*", "*localwifinlpbackend*"—"*wifi location service*"), but also cases

in which both names appear to be completely unrelated (*e.g.,* "*marshmallow adventure*"—"*flappy candy*" or "*filebox*"—"*myfaves*"). The package name and app icon of "*marshmallow adventure*" reveals that the app was previously promoted under the name in the app manifest as a clone of the well-known *Flappy Bird* app, whereas the origin of "*myfaves*" is unclear. Our findings confirm the belief that relying on its name to identify an app (as users tend to do [79]) can lead to the installation of undesired apps. We also show the potential applications of attribution graphs to detect clone apps.

6.3 Signal Consistency Within Markets

We now assess signal consistency within a given market from an author perspective, *i.e.,* the extent to which the same signals are used across all apps by a publisher. The company developing an app can be different to the app's author and even to the company publishing it (see §2). Still, signal consistency is critical for researchers, users, and regulators to correctly and unambiguously attribute authorship.

We evaluate the consistency by measuring the re-use of signals by a particular publisher, as well as across publishers, focusing on those signals available on most markets: the app name, the developer name (per market metadata), and the signing certificate. Since these signals are not all available on Baidu, we exclude it from the analysis.

First, we evaluate the degree to which certificates are uniquely used by a single publisher. We analyze our attribution graph, identifying all developer name nodes that are connected to a particular certificate fingerprint node, which represents the number of developer names publishing apps signed with this certificate. We find that a relatively small number of certificates on the markets are used across multiple developer names (ranging from 1.2% on Google Play to 4.9% on Tencent). However, these certificates are used to sign a large fraction of market entries (ranging from 15.2% on Google Play to 22.6% on Tencent). Some of these market entries are associated to apps from the same developer that are published under different developer names (*e.g.,* international subsidiaries, business units, or development teams), but most cases are associated with app building frameworks like Andromo [82] or AppyPie [83]. These development frameworks offer software authors mechanisms for outsourcing the app building process either through automatic app creation techniques, or by developing the app for the publisher [32]. The final app, however, is signed directly by the app building framework. These artifacts result in many unrelated apps sharing the same signing certificate. We study app building frameworks further in §7.2.

We also quantify the opposite aspect: the number of certificate fingerprint nodes connected to a developer name which measures the number of certificates used by the developer to sign their apps. We find that it is common practice for authors to release apps with more than one certificate: the percentage of developer accounts of apps signed with multiple certificates ranges from 9.3% for APKMirror to 44.5% for Google Play, and affect 34.2% and 66.1% of market entries on the two markets, respectively. If two apps by the same developer operate fully independent, there is currently
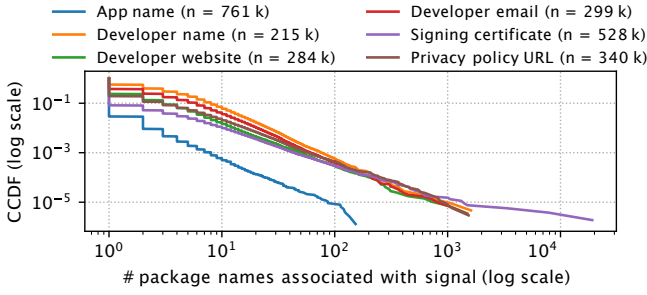
Fig. 3: Complementary CDF of the number of package names associated with individual signals on Google Play.

no incentive for a developer to publish them under the same certificate.

Using the same method, we also group market entries by app name and count the number of apps published under more than one developer name. We find that the percentage of listings that share their app name with another listing but are published by different developers varies from 0% on APKMirror to 3.4% on APKMonk. A notable example on Google Play is the app name *Messages*, which has been published under 25 different package names, with Google's version being the most popular one with more than 1B downloads.

Google Play. We direct our efforts towards analyzing those attribution signals that are only consistently available on Google Play: the developer website, the privacy policy URLs, and developer email. There is a relatively low correlation between the developer name and other developer-related signals: 26.3% of developer names publish apps with more than one website, 23.7% with multiple email addresses, and 31.2% with multiple privacy policies. The latter is an expected outcome, as two apps by the same developer can collect different types of personal data and hence may have a different privacy policy. Note that the automatic analysis of the privacy policies to extract attribution information is an orthogonal research challenge [84]–[86]. The extent to which email addresses and websites differ is more surprising. The developer *"Rad3 Limited"* published two apps *"Wakachangi GO"* and *"Smales Farm"* under the email and websites *"contact@wakachingi.com"* and *"http://www.wakachangi.com/"*, and *"mark@smalesfarm.co.nz"* and *"https://smalesfarm.co.nz"*). These findings show that contact information is in some cases *app*-related and not *developer*-related.

We further examine the distribution of package names published under a particular signal value shown in Figure 3 as a complementary cumulative distribution function (CCDF). The figure shows that for each signal type, there are outliers of signals under which a large volume of apps have been published. These cases – which include the aforementioned *Messages* app name – highlight the inability of relying on a single signal for sound attribution.

6.4 Signal Consistency Across Markets

An app's package name uniquely identifies an app within the Android OS and within markets. However, the association of the unique package name with a particular app might not prevail from market to market since a market operator can only enforce this policy on their own market.

Signal inconsistency across markets has implications in research efforts trying to compare the catalogue of markets and developer publishing and release strategies: the package name can be unreliable for identifying unique apps across markets. Hypothetically, two different entities sharing the same package name might publish an app each under the same developer name but on two different markets. The signals would suggest the same underlying developer even though this is not the case, thus leading the user to install a different app (with a different behavior) depending on the market where it is downloaded. This is also important for researchers, as the market of origin becomes a relevant variable when mining repositories and reasoning about the data and measurement results, or when studying market catalogues and their risks for the end users. In other words, measurements on Google Play might not extrapolate to other markets, or cross-market datasets might be polluted by unrelated apps.

We calculate the number of apps (by package name) listed on more than one market to measure how inconsistent attribution signals are across markets. Out of the total 1,355,186 unique package names present in our dataset, 158,735 (11.71%) have been published on multiple markets. We compare the certificate nodes in the attribution graph used to sign apps across markets sharing a package name. Figure 4 shows the percentage of these apps per market pairs that are signed by the same certificate on both markets. Notably, APKMonk and Google Play have a full overlap. This suggests that APKMonk operates as a mirror of Google Play. The overlap between Baidu and the western markets is relatively poor (between 82% and 90%), which raises the question whether market entries that share a package name are the same app and are published by the same author.

For apps co-published on both Google Play and another market, we see that between the percentage of apps signed by a Google certificate (as identified by its subject) on the alternative market ranges between 2.4% and 24.3% for Baidu and Tencent respectively. Those are apps that are likely to have first been published on the Play Store, after which their signed APKs were published on the other market as recommended by Google [87]. As such, Google Play's signing policy does not only affect their own market, but has also started to impact on attribution of apps across markets at the certificate level.

In a similar fashion as the certificates, we compute the percentage of package names published under the same app name between market pairs (*i.e.,* every possible combination of two markets). We run this analysis only for those app names sharing the same alphabet to avoid noise caused by language differences. We observe that there are significant differences in the app names depending on the origin of the market. We find that Chinese markets depict a relatively higher overlap of app names. 77.0% of package names published on a pair of Chinese markets are published under the same app name. The percentage is even higher for pairs of Western markets (91.7%). However, the percentage is much lower for those package names published across a Western and a Chinese market (52.7%).

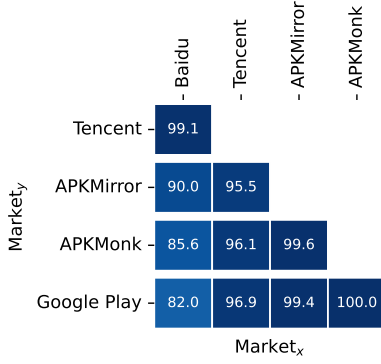Similarly, we compute these percentages for the devel-

Fig. 4: Percentage of package names published on both market$_x$ and market$_y$ signed with the same certificate.

oper name. We exclude Baidu from this analysis since this market does not report developer names (§4)). Across Western markets, package names are published under the same developer name in 92.2% of the cases, and only 66.5% across Tencent and Western markets. We find examples of apps in which the developer name is related across markets (*e.g.,* "*naver webtoon corp.*" and "*webtoon entertainment*"), while others are completely unrelated (*e.g.,* "*gomo limited*" and "*go launcher dev team*"). Possible explanations for the disparity between Chinese and Western signal values include more rampant app repackaging, publication of apps under a subsidiary (or different) company or poor practises of synchronizing metadata on markets across regions. However, these hypothesis are difficult to validate with our dataset.

RQ2: How consistent are attribution signals within apps, within markets and across them? We identified a significant lack of overlap between app names declared in the app's manifest and the published app name. We find both a significant duplicate signal usage between developer accounts (*e.g.,* identical certificates and app names used to publish app from different developer accounts) and multiple signal usage by a single developer account within markets (*e.g.,* publishing apps under different certificates or email addresses). Moreover, our results highlight the challenge of tracking authorship across markets, due to their package names being published under different app names and - perhaps more importantly - different certificates. This problem is magnified by tracking package names across Western and Chinese markets.

7 CASE STUDY: GOOGLE PLAY STORE

Our results in §5 and §6 confirm that publicly available attribution signals are unsound and can lead to confusion. Other signals available in market profiles, such as the developer website or their email address could improve attribution, but these signals are only reliably available on Google Play. The absence of these signals in most alternative markets not only harms attribution of app authors (*i.e.,* transparency), but also automatic and large-scale accountability and comparative market analysis. For example, users might not be able to easily find the author's contact information about the company prior to installation in order to decide whether to trust or not the developer, or even exercise their data rights

in case they need to. Similarly, researchers might not be able to responsibly disclose vulnerabilities without contact details.

In this section, we set out to answer **RQ3**. We focus our analysis explicitly on the Google Play Store, as (1) it is the official and most prevalent Android marketplace [88]; (2) it is the richest in attribution signals at the market level; and (3) Google Play has a significant number of publication policies (§2.2) that implicitly or explicitly affect attribution.

7.1 Enforcement of Publication Policies

We start by investigating the enforcement of Google Play's publication policies through different experiments involving the publication of potentially deceptive apps on Google Play. Particularly, we focus on those policies and recommendations related to attribution signals. According to Google's documentation, each app submitted to Google Play goes through an automated vetting process that flags an app for manual inspection if it is found to contain suspicious behaviors [89]. As impersonation and tampering are explicitly prohibited (§2.2), apps violating these policies can be expected to be rejected by the market. The primary goal of our experiment is to confirm this hypothesis and assess the robustness of Google Play's vetting process to detect and prevent the publication of apps that attempt to impersonate or tamper with other apps or developer profiles. In doing so, we also shed light onto the fidelity of the attribution signals available in the market.

For each experiment, we compiled and attempted to publish a repackaged and completely benign app that includes partially fake developer information. We chose two open source apps with licenses that allow their redistribution to avoid infringing on any intellectual property rights. Namely, we leverage *Open Sudoku* [90], a game with a moderate number of downloads (10,000+), and the *Signal Private Messenger* [91], a popular privacy-focused messenger, with more than 100 million downloads as of March 2022. Specifically, we tested whether Google Play (1) conducts any code similarity analysis between new and existing apps; (2) evaluates the similarities between market signals; and (3) carries out a more exhaustive vetting if the affected app has a high profile (*i.e.,* a larger number of downloads). We released each app under a new developer account to prevent apps from being rejected based on developer reputation.

Ethical Considerations. We took several steps to ensure that our experiments would cause no harm to users or to the platform itself. We received approval from our institutional ethical review board before conducting the experiments. We note that our intention is not measuring whether users install clone apps but whether these apps get flagged during the review process. All apps that we published for this experiment display a notification to the users, clearly stating that they are part of a research project and they are not the original one. Additionally, this notification redirected users to the original apps and advised them to uninstall our clone. Furthermore, we did not collect any personal information about the users that installed our apps. Finally, we removed the apps from the Play Store after our experiments to prevent any confusion. Our experiment might have affected the platform, though we consider the impact to be negligible.

Since all apps go through an automatic vetting process when published, and with an estimated 100k apps [92] published monthly, the overhead of two apps is minimal. There are over 3 million apps [93] publicly available on the Google Play Store, so the release of three apps (*i.e.,* one app was released on two different occasions) will not affect any potential measurement or research conducted on the whole spectrum of the Google Play Store in any meaningful way. Thus, we estimate the benefits of understanding the vetting process of the Google Play Store and its shortcomings to be far greater than the potential overhead of publicly releasing two apps.

Experiment #1: Code similarity. Performing any code similarity for every new app submission might be too computationally expensive given the scale of the market, even excluding different app versions. To test if Google Play performs such a heavy-weight analysis, we released an unmodified clone of the Open Sudoku app, with all the market signals completely modified. This app was accepted for publication, suggesting that the Play Store vetting process does not perform code-level analysis. We left the app on the market for three months before removing it ourselves.

Experiment #2: Metadata similarity. Google Play's Developer Program Policy specifically prohibits impersonation [94], and publishing an app with highly similar market signals should therefore be prohibited. We released another clone of Open Sudoku, this time with signals that were equal to the original apps whenever possible, or highly similar when not (*i.e.,* the package name and the developer name). We used `org.moire.attribution.opensudoku` instead of the original `org.moire.opensudoku` package name, and we re-used the original name of the developer without tildes. The Play Store accepted the app, which shows that impersonation is still possible despite the official policy. We left the app on the market for three months before removing it ourselves.

Experiment #3: Protection of high-profile apps. We hypothesize that Google Play prioritizes the protection of popular or high-profile apps. To test this hypothesis, we release a clone of the Signal Private Messenger with all the attribution signals resembling the original app as closely as possible. The Play Store rejected our first clone of the app for violating the impersonation policy, specifically referring to the app's name. We note that in a subsequent submission, they reported issues with re-using the app's icon. Compared to the previous experiments, these rejections suggest that the Play Store is indeed prioritizing selected, high-profile apps. Ultimately, our cloned app seemed to have become part of a manual vetting process. Just like the original, it contained a PayPal donation link which went against the store policies of only using Google's payment system for in-app purchases. These links were attached as screenshots to our rejection, which likely did not come from an automatic process. Meanwhile, the original app removed the PayPal payment in the app.

Our results indicate that the mechanisms currently in place to enforce publication policies on Google Play are not perfect and that self-reported market signals are not trustworthy for attribution purposes, particularly for non-popular apps.

7.2 Multiple Signal Attribution Graphs

As shown in § 6.3, attribution signals tend to correlate poorly with each other on Google Play. In this section, we explore the soundness and validity of using a combination of signals for attribution. We rely on our previously defined attribution graph (§6) to represent the dataset as a graph. This large attribution graph consists of connected components, or clusters, that we can consider to be associated with a particular author entity. While we acknowledge that there might be a legitimate reason for one company to have more than one value for its metadata (*e.g.,* different departments or subsidiaries), this reduces the attribution power of observed signals and makes it hard to group all apps belonging to the same owner.

The resulting graph contains 158,879 clusters for 804,041 market entries. We find that 76,054 (9.5%) market entries are isolated in their own cluster, and that the largest cluster comprises 288,218 (35.8%) market entries. Furthermore, 5,277 (3.3%) clusters are *fully consistent* (*i.e.,* all apps in these non-isolated clusters are published with the same signals, except for the app name), indicating that the majority of the clusters contain ambiguous information. Whereas most APKs are signed with a single certificate, we find 38 and 52 apps signed with two or three certificates, respectively. In these cases, the app signed by multiple certificates can bridge two, otherwise disconnected, apps with each other.

Node centrality. The size of the one large cluster suggests that a naive attribution graph creation can lead to substantial over-attribution. To identify the signal causing this over-attribution, we compute the *betweenness centrality* of each signal, a metric expressing the fraction of the shortest paths in the graph going through a particular node [95]. Our intuition is that the most central node must be key for connecting vertices in the graph that would otherwise not be connected, and therefore is a likely cause for over-attribution. When looking at the centrality value of nodes, we see that the vast majority have a centrality close to zero. There are, nonetheless, a few outliers exhibiting a relatively high centrality.

By investigating highly-central signals, we find patterns that make attribution challenging. App names are not unique on the Play Store, thereby making it difficult to differentiate between apps sharing a name. Examples include *BMI Calculator* (published 87 times by 86 different developer names), *Flashlight* (published 153 times by 149 developer names) and *Music Play* (published 113 times by 110 developer names). We also observe that the highest centrality corresponds to a set of highly prevalent signing certificates. Relying on the subject information, we manually analyze the top 10 companies (by centrality of the certificate) and observe that all are related to frameworks or companies that build apps for others [32]. The signal with the highest centrality is a certificate associated with Andromo, which is used to sign 19,096 apps. As previously mentioned, Andromo is an app development framework to build apps based on pre-existing components [82] that are all signed by the same certificate and are ready to submit for publication on the market. Andromo does not mention this signing practice in their terms of service [96], and in fact recommends against enrolling for

signing delegation to the Play Store [97]. Apps built with Andromo have been published under 1,712 different developer names, which—other than the signing certificate—have little in common with each other. In this particular case, the package name also indicates the use of this development framework, as 86.2% of Andromo apps follow the package name scheme {com,net}.andromo.dev<dev_id>.app<app_id>. Using the same strategy of finding other highly-central certificate nodes, we find other popular app builders, who are collectively responsible for signing tens of thousands of apps (*e.g.,* Seattle Cloud, Bizness Apps or Mobincube sign over 7k, 3k and 1k apps respectively).

Our clustering approach reveals another app builder through the privacy policy associated with AppsGeyser [98], which is shared by 458 package names that otherwise share no other signal. We also find two privacy policy URLs that serve generic privacy policies related with products (mostly SDKs) intended to be used across different apps (*i.e.,* Firebase [99] and MyAppTerms [100]), which are used by 226 and 390 package names, respectively.

One consequence of this practice is that customers are fully reliant on the app builder to provide updates for the app, as the initial signing certificate is required to do so. In addition, as we discuss in §8, certificates have use cases beyond enforcing update integrity. As a result, the app builders have full control over the declared and used permissions, and can automatically grant permissions across apps from different authors without user awareness. Note that in Android `signature permissions` are automatically granted to apps signed with the same certificate [101]. From an attribution standpoint, these certificates become meaningless to identify the author behind an app. This highlights the separation of roles that makes attribution and accountability extremely hard on Google Play. The company in charge of publishing—and, presumably, the one that should be accountable for potential privacy and security issues—is not the same as the one that has developed, or even signed, the app. In cases such as Andromo, the developing process of the app happens automatically. This leaves a gap between what is mandated by current legislation and market policies and the ecosystem of Google Play apps. Specifically, it is typically assumed that the developer is the same as the company publishing the app and thus the one liable for potential privacy and security violations.

Large organizations. The attribution graphs allows us to study the attribution signals of prominent tech companies. We select a curated list of developer names publishing popular apps with more than 1B installs. The large download count serves as the ground-truth for the legitimacy of these apps and developer accounts. For each developer name, we collect the number of unique signals used across apps published under the same name (*i.e.,* account on Google Play) and measure how many other developer names use the same certificate as the main developer account to sign their apps. Table 6 shows the results of our analysis. Most of these organizations use multiple certificates to sign their apps with the exception of Snap, TikTok, WhatsApp, Instagram, and Skype. An extreme case is HP, which nearly has a separate certificate per published app. For Google, Samsung, WhatsApps and HP, we see that one or more alternative developer name has used one of the certificates

TABLE 6: Uniqueness of signals by large organizations and the developer names they use with the same signing certificate.

Developer name	# apps	# emails	# websites	# certs	Other developer names
Google LLC	136	48	103	85	Google Fiber Inc.; The Infatuation Inc.
Facebook	17	7	13	7	—
Microsoft Corporation	83	62	54	23	—
Samsung Electronics Co., Ltd.	58	19	24	24	Logmein, Inc.; Maas360; Sidi; Teamviewer; Nsl Utils; Sophos Limited; Barco Limited (Awind)
Twitter, Inc.	2	2	2	2	—
Snap Inc	1	1	1	1	—
TikTok Pte. Ltd.	2	1	1	1	—
Netflix, Inc.	5	5	2	4	—
WhatsApp Inc.	2	2	2	1	Whatsapp LLC
HP Inc.	37	23	28	30	Hewlett Packard Enterprise Company; Printeron Inc
Instagram	5	2	2	1	—
King	20	19	18	7	—
Skype	2	1	2	1	—

to publish an app. Some of these names are related to the company, pointing to different units of the company releasing the app (*e.g.,* "Hewlett Packard Enterprise Company" for HP). However, as Table 6 shows, most alternative names are unrelated to the original company (*e.g.,* "Logmein, Inc" for Samsung). Conversely, both WhatsApp and Instagram are subsidiary companies of Facebook Inc., but do not share any cryptographic link with their parent company. A similar relationship holds for Microsoft and Skype. Note that all these cases refer to companies acquired and merged into a larger organization. Overall, these results paint a diverse picture of publication and development strategies across companies that can impede automatic attribution, even for software released by well-known companies.

RQ3: How is the Google Play Store affected by the lack of signal availability and consistency? Our case study reveals the ubiquity of app development frameworks operating on the Play Store. The signals that are re-used across apps created by these frameworks (*e.g.,* certificate or privacy policy) hamper accurate attribution. Conversely, our analysis of the publishing behaviour of high-profile demonstrates a large variety of different signals under which apps are published, hampering attribution too.

8 CONCLUDING REMARKS

Our empirical results show that automatic author attribution in the Android ecosystem is a hard problem. The answers to the research questions highlight the roadblocks that hamper this attribution:

- Not only is developer contact information sporadically provided on markets, the markets that do provide it do not enforce the availability of this information. This problem is amplified by the fact that a non-negligible fraction of apps change their market signals over time (**RQ1**).
- We identified frequent signal inconsistencies of apps within the publication signals of an individual app,

across apps from a developer account on a specific market, and across apps on multiple markets (**RQ2**).

- The publication behavior of app development frameworks and larger companies results in the association of individual signals with multiple developers accounts, and vice versa (**RQ3**).

By introducing the concept of attribution graphs, we have shown that the software developer or author is not always the same as the company accountable for the app. The proliferation of app development frameworks and companies that build apps for others calls for a re-definition of roles and for a re-adjustment of how researchers approach attribution for different studies.

Implications. Imprecise app attribution has negative consequences for a number of research areas and applications as discussed in §3.1. It affects measurements and analyses of the app ecosystem, market dynamics, or automatic detection of deceptive actors and practices. The unavailability of signals limits the software attribution community to rely on a combination of signals to improve upon attribution based on single signals. The volatility of signals threaten the validity of studies on the long term, as drawn conclusions may not hold years later. Our results challenge methods of attribution that prior research has resorted to. Using unique signatures to count unique developers will undercount them as app-frameworks sign apps with the same certificate for different developers (See §7.2) [31], [35], [37]. However, using the developer name or website from the market might overcount entities such as large corporations that created multiple accounts with slightly different names or domains (§7.2) [5], [38], [102]. Our results also show that certificate-based tracking of apps across markets leads to inaccurate results (§6.4) [20].

Imprecise attribution also damages transparency for users, as the absence of clear signals about which company is accountable for an app limits their ability to take an informed decision about whether to install it or not or when exercising GDPR rights. In the case of privacy abuses and security vulnerabilities, it makes disclosures harder. Furthermore, inconsistencies across markets can lead users to install an app by mistake, only because it presents an ambiguous signal (*e.g.,* the app name).

Signing Certificates. We found that the assumption that certificates can be used for attribution (§ 3.1) is flawed, as one author does not necessarily use a single certificate and a single certificate can be used to sign apps from different organizations. The adoption of Google's signing delegation process by developers also reduces the value of signing certificates as attribution signals. The unreliability of the signing certificate has profound consequences for the platform security, as it has long been used as a proof of authorship in prior work and even for threat intelligence. The installation of two apps from different organizations signed with the same certificate also has privacy implications: Android automatically grants permissions requested by one app to the other app [35], [103]. Moreover, certificates are also used to link apps to websites [104], thus enabling the website to check for *"their"* installed apps [105] and even share credentials without direct consent [106]. These implications impact not only unaware end users, but also developers that choose to hand over the certificate signing to another entity.

Recommendations. Only profound platform and market changes can solve the predicament of Android attribution. First, the Android ecosystem should move away from self-signing certificates and vet the information disclosed in apps' market profiles. Other prominent software ecosystems already rely on more sound signature mechanisms, such as Apple issuing valid certificates for its app store [16] and Windows relying on a PKI [14]. While not perfect [107], we argue that these approaches limit the number of certificates with incomplete or invalid information while also raising the bar for malicious actors. The open nature of the Android ecosystem should be preserved (*e.g.,* free and automatic certificate issuance) and should allow for the current business practises of developer to remain in place (*e.g.,* certificate reuse across publishing developers and multiple certificate use by developers). In practise, however, the certificate issuance process would enforce (the very least) for certificate-related fields to be available, but could also include verification on the values of these fields. Showing this information on the market listing of an app on markets would also expose this currently unavailable information to the end user. An alternative option would be to rely on independent third-party verification for attribution, but we argue that this would simply shift responsibility from the market to third-party entities. An implementation could be developed without requiring changes to the current ecosystem and could range from automatic authorship attribution to producing warning for conflicting signals to users.

Our work also has implications for large app repositories compiled for research purposes. Placing app metadata in existing repositories such as AndroZoo and antivirus engines like VirusTotal would benefit the research community and provide intelligence to contextualize observed software behaviors. Prominent markets like Google Play should include veracity checks for developer metadata included in Google Play Protect, while ensuring the identity of the developer as iOS does.

Future work. Current legislation [57] mandates software that collects and shares user-data to provide a comprehensive privacy policy that should include, among other information, data about the company behind the software [58]. Therefore, we argue that automatically parsing privacy policies can yield information useful for attribution. However, automatic parsing of legal texts is a complex problem, specially at scale [84]–[86]. In a preliminary analysis, we find that it is not common for policies to contain contact information as only 59% and and 12% of them have an email and an address respectively. Similarly, other more complex and expensive methods such as UI inspection and code analysis could be used for extracting attribution data [37]. Finally, we believe that the generalizable concept of attribution graphs has potential applications beyond attribution in the Android ecosystem, including large-scale market analysis, study of developer trends and practices, and clone detection.

ACKNOWLEDGMENTS

REFERENCES

[1] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis–1,000,000 apps later: A view on current android malware behaviors," in *2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*. IEEE, 2014.

[2] S. Alrabaee, P. Shirani, M. Debbabi, and L. Wang, "On the feasibility of malware authorship attribution," in *International Symposium on Foundations and Practice of Security*. Springer, 2016.

[3] L. B. Platon Kotzias, Juan Caballero, "How did that get in my phone? unwanted app distribution on android device," in *IEEE Symposium on Security and Privacy (SP)*, 2020.

[4] "Chapter 3 – rights of the data subject," https://gdpr-info.eu/chapter-3/, 2018, "Accessed May 2021".

[5] H. Wang, Z. Liu, Y. Guo, X. Chen, M. Zhang, G. Xu, and J. Hong, "An explorative study of the mobile app ecosystem from app developers' perspective," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2017.

[6] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, "Beyond google play: A large-scale comparative study of chinese android app markets," in *Proceedings of the Internet Measurement Conference (IMC)*, 2018.

[7] H. Wang, H. Li, and Y. Guo, "Understanding the evolution of mobile app ecosystems: A longitudinal measurement study of google play," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2019.

[8] M. Ali, M. E. Joorabchi, and A. Mesbah, "Same app, different app stores: A comparative study," in *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*, 2017.

[9] N. Zhong and F. Michahelles, "Google play is not a long tail market: An empirical analysis of app adoption on the google play app market," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013.

[10] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," in *The 2014 ACM international conference on Measurement and modeling of computer systems*, 2014.

[11] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," *Telemat. Inf.*, 2011.

[12] N. d'Heureuse, F. Huici, M. Arumaithurai, M. Ahmed, K. Papagiannaki, and S. Niccolini, "What's app? a wide-scale measurement study of smart phone markets," *Proceedings of the SIGMOBILE Mobile Computing and Communications Review*, 2012.

[13] B. Kaliski, "Pkcs# 7: Cryptographic message syntax version 1.5," 1998.

[14] D. Kim, B. J. Kwon, and T. Dumitraş, "Certified malware: Measuring breaches of trust in the windows code-signing pki," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[15] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman, "Towards a complete view of the certificate ecosystem," in *Proceedings of the 2016 Internet Measurement Conference*, 2016.

[16] "Apple Developer Program: What You Need To Enroll," https://developer.apple.com/programs/enroll/, 2021.

[17] "Identity Verification," https://developer.apple.com/support/identity-verification/, 2021.

[18] "Google Play Policy Center," https://support.google.com/googleplay/android-developer/answer/9898842, 2021.

[19] "Sign your app," https://developer.android.com/studio/publish/app-signing, 2021.

[20] M. Lindorfer, S. Volanis, A. Sisto, M. Neugschwandtner, E. Athanasopoulos, F. Maggi, C. Platzer, S. Zanero, and S. Ioannidis, "Andradar: fast discovery of android applications in alternative markets," in *Proceedings of the International conference on detection of intrusions and malware, and vulnerability assessment (DIMVA)*, 2014.

[21] E. Okoyomon, N. Samarin, P. Wijesekera, A. Elazari Bar On, N. Vallina-Rodriguez, I. Reyes, Á. Feal, and S. Egelman, "On the ridiculousness of notice and consent: Contradictions in app privacy policies," *Workshop on Technology and Consumer Protection (ConPro)*, 2019.

[22] J. Crussell, C. Gibler, and H. Chen, "Attack of the clones: Detecting cloned applications on android markets," in *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2012.

[23] B. Kim, K. Lim, S.-J. Cho, and M. Park, "Romadroid: A robust and efficient technique for detecting android app clones using a tree structure and components of each app's manifest file," *IEEE Access*, 2019.

[24] H. Wang, Y. Guo, Z. Ma, and X. Chen, "Wukong: A scalable and accurate two-phase approach to android app clone detection," in *Proceedings of the International Symposium on Software Testing and Analysis*, 2015.

[25] K. Chen, P. Liu, and Y. Zhang, "Achieving accuracy and scalability simultaneously in detecting application clones on android markets," in *Proceedings of the International Conference on Software Engineering*, 2014.

[26] M. Sun, M. Li, and J. C. S. Lui, "DroidEagle: Seamless detection of visually similar Android apps," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2015.

[27] D. Barrera, J. Clark, D. McCarney, and P. C. van Oorschot, "Understanding and improving app installation security mechanisms through empirical analysis of android," in *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2012.

[28] L. Wu, M. Grace, Y. Zhou, C. Wu, and X. Jiang, "The impact of vendor customizations on android security," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2013.

[29] K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, W. Zou, and P. Liu, "Finding unknown malice in 10 seconds: Mass vetting for new threats at the google-play scale," in *Proceedings of the USENIX Security Symposium*, 2015.

[30] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Authorship attribution of android apps," in *Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2018.

[31] V. Kalgutkar, N. Stakhanova, P. Cook, and A. Matyukhina, "Android authorship attribution through string analysis," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018.

[32] M. Oltrogge, E. Derr, C. Stransky, Y. Acar, S. Fahl, C. Rossow, G. Pellegrino, S. Bugiel, and M. Backes, "The rise of the citizen developer: Assessing the security impact of online app generators," in *IEEE Symposium on Security and Privacy (SP)*, 2018.

[33] L. Li, T. F. Bissyandé, and J. Klein, "Rebooting research on detecting repackaged android apps: Literature review and benchmark," *IEEE Transactions on Software Engineering*, 2019.

[34] P. Liu, L. Li, Y. Zhao, X. Sun, and J. Grundy, "Androzooopen: Collecting large-scale open source android apps for the research community," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*, 2020.

[35] J. Gamba, M. Rashed, A. Razaghpanah, J. Tapiador, and N. Vallina-Rodriguez, "An analysis of pre-installed android software," in *IEEE Symposium on Security and Privacy (SP)*, 2020.

[36] S. Sebastian and J. Caballero, "Towards attribution in mobile markets: Identifying developer account polymorphism," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[37] G. Xu, C. Zhang, B. Sun, X. Yang, Y. Guo, C. Li, and H. Wang, "Appauth: Authorship attribution for android app clones," *IEEE Access*, 2019.

[38] S. Aonzo, A. Merlo, G. Tavella, and Y. Fratantonio, "Phishing attacks on modern android," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[39] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, and T. Karagiannis, "Rise of the planet of the apps: A systematic

study of the mobile app ecosystem," in *Proceedings of the Internet Measurement Conference (IMC)*, 2013.

[40] S. Farooqi, A. Feal, T. Lauinger, D. McCoy, Z. Shafiq, and N. Vallina-Rodriguez, "Understanding incentivized mobile app installs on google play store," in *Proceedings of the Internet Measurement Conference (IMC)*, 2020.

[41] "Dataset sharing | DropBox," https://www.dropbox.com/sh/dc8aa5qggto7z82/AADY7oyzt9Z7RP-c2pNQ98eXa?dl=0, 2021.

[42] "App Signing Considerations," https://developer.android.com/studio/publish/app-signing#considerations, 2021.

[43] "Signing Mac Software with Developer ID," https://developer.apple.com/developer-id/, 2021.

[44] "Recent Android App Bundle improvements and timeline for new apps on Google Play," https://android-developers.googleblog.com/2020/08/recent-android-app-bundle-improvements.html, 2020, "Accessed May 2021".

[45] "Code Transparency for App Bundles," https://developer.android.com/guide/app-bundle/code-transparency, 2021.

[46] "Submit android apps," https://www.apkmonk.com/submit-app/, 2021.

[47] "Baidu Mobile Application Platform - Application Submission," https://app.baidu.com/newapp/docs/%E5%BA%94%E7%94%A8%E5%88%86%E5%8F%91/%E5%BA%94%E7%94%A8%E5%8F%91%E5%B8%83/%E5%BA%94%E7%94%A8%E6%8F%90E4%BA%A4, 2021.

[48] "APK Upload - APKMirror," https://www.apkmirror.com/apk-upload/, 2021.

[49] "Google Play Policy Center," https://support.google.com/googleplay/android-developer#topic=3450769, 2021.

[50] "Apkmonk - market policy," https://www.apkmonk.com/submit-app/, 2021, "Accessed May 2021 archived at http://archive.today/2GVPl".

[51] "Tencet - market policy," https://open.qq.com/eng/reg, 2021, "Accessed May 2021 archived at http://archive.today/Nbusz".

[52] "Apkmirror - market policy," https://www.apkmirror.com/faq/, 2021, "Accessed May 2021 archived at http://archive.today/mXflW".

[53] "Android Developers — manifest element," https://developer.android.com/guide/topics/manifest/manifest-element.html#package, 2021.

[54] "Google Play Policy Center," https://support.google.com/googleplay/android-developer/topic/9858052, 2021, archived at http://archive.today/gBMYU.

[55] "Google Play Policy Center," https://support.google.com/googleplay/android-developer/answer/9888689, 2021.

[56] "Google Play Policy Center," https://support.google.com/googleplay/android-developer/answer/9888374, 2021.

[57] 2018 reform of eu data protection rules. European Commission. [Online]. Available: https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf

[58] E. Commission. Directive 2000/31/ec of the european parliament and of the council of 8 june 2000 on certain legal aspects of information society services, in particular electronic commerce, in the internal market ('directive on electronic commerce'). [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32000L0031&from=EN

[59] S. of California Department of Justice, "California consumer privacy act (ccpa)," https://www.oag.ca.gov/privacy/ccpa, 2018.

[60] E. Blázquez, S. Pastrana, Á. Feal, J. Gamba, P. Kotizias, N. Vallina-Rodriguez, and J. Tapiador, "Trouble over-the-air: An analysis of fota apps in the android ecosystem," in *IEEE Symposium on Security and Privacy (SP)*, 2021.

[61] N. Marastoni, A. Continella, D. Quarta, S. Zanero, and M. D. Preda, "Groupdroid: Automatically grouping mobile malware by extracting code similarities," in *Proceedings of the 7th Software Security, Protection, and Reverse Engineering / Software Security and Protection Workshop*, 2017.

[62] V. Rastogi, Y. Chen, and W. Enck, "Appsplayground: Automatic security analysis of smartphone applications," in *Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2013.

[63] J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez, "Bug fixes, improvements,... and privacy leaks: A longitudinal study of pii leaks across android app versions," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[64] T. T. Nguyen, M. Backes, N. Marnau, and B. Stock, "Share First, Ask Later (or Never?) - Studying Violations of GDPR's Explicit Consent in Android Apps," in *Proceedings of the USENIX Security Symposium*, 2021.

[65] R. Li, W. Diao, Z. Li, J. Du, and S. Guo, "Android custom permissions demystified: From privilege escalation to design shortcomings," 2021.

[66] J. Reardon, Á. Feal, P. Wijesekera, A. E. B. On, N. Vallina-Rodriguez, and S. Egelman, "50 ways to leak your data: An exploration of apps' circumvention of the android permissions system," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019.

[67] G. S. Tuncay, S. Demetriou, K. Ganju, and C. Gunter, "Resolving the predicament of android custom permissions," 2018.

[68] "Scrapy | A Fast and Powerful Scraping and Web Crawling Framework," https://scrapy.org/, 2021.

[69] "APKMonk," https://www.apkmonk.com/, 2021.

[70] "APKMirror," https://www.apkmirror.com/, 2021.

[71] "Baidu App Store," https://shouji.baidu.com/, 2021.

[72] "Tencent App Store," https://android.myapp.com/, 2021.

[73] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting Millions of Android Apps for the Research Community," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*, 2016.

[74] M. Ikram, N. Vallina-Rodriguez, S. Seneviratne, M. A. Kaafar, and V. Paxson, "An analysis of the privacy and security risks of android vpn permission-enabled apps," in *Proceedings of the 2016 internet measurement conference*, 2016, pp. 349–364.

[75] "Number of available applications in the google playstore," https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2021, "Accessed November 2021".

[76] P. Calciati, K. Kuznetsov, X. Bai, and A. Gorla, "What did really change with the new release of the app?" in *Proceedings of the 15th International Conference on Mining Software Repositories*, 2018.

[77] Y. Hu, H. Wang, R. He, L. Li, G. Tyson, I. Castro, Y. Guo, L. Wu, and G. Xu, "Mobile app squatting," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2020.

[78] Z. Chen, "Thousands of HiddenAds Trojan Apps Masquerade as Google Play Apps," https://www.mcafee.com/blogs/other-blogs/mcafee-labs/multi-tricks-hiddenads-malware/, 2020.

[79] "Android users looking for Elon Musk on Clubhouse caused an identically named app to pull itse," https://9to5google.com/2021/02/01/clubhouse-android-app-elon-musk/, 2021, "Accessed April 2021".

[80] H. Wang, H. Li, L. Li, Y. Guo, and G. Xu, "Why are Android apps removed from Google Play?: A large-scale empirical study," in *Proceedings of the 15th International Conference on Mining Software Repositories*, 2018.

[81] A. Marzal and E. Vidal, "Computation of normalized edit distance and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, 1993.

[82] "Andromo - Mobile App builder for Android. No coding," https://www.andromo.com/, 2021.

[83] "Appypie - homepage," https://www.appypie.com/, 2021, "Accessed May 2021".

[84] S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. Sadeh, S. Bellovin, and J. Reidenberg, "Automated analysis of privacy requirements for mobile apps," in *2016 AAAI Fall Symposium Series*, 2016.

[85] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer, "Polisis: Automated analysis and presentation of privacy policies using deep learning," in *Proceedings of the USENIX Security Symposium*, 2018.

[86] B. Andow, S. Y. Mahmud, W. Wang, J. Whitaker, W. Enck, B. Reaves, K. Singh, and T. Xie, "Policylint: investigating internal privacy policy contradictions on google play," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 585–602.

[87] "Use Play App Signing," https://support.google.com/googleplay/android-developer/answer/9842756?visit_id=637574359279602919-373176865, 2021.

[88] "A look at the android market (aka google play) on its 10th anniversary," https://techcrunch.com/2018/10/22/a-look-at-the-android-market-aka-google-play-on-its-10th-anniversary/, 2018, "Accessed May 2021".

[89] https://developers.google.com/android/play-protect/cloud-based-protections, 2021.

[90] "Open sudoku," https://play.google.com/store/apps/details?id=org.moire.opensudoku, "Accessed Jan 2022".

[91] "Signal," https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms, "Accessed Jan 2022".

[92] "Average number of new android app releases via google play per month from march 2019 to february 2021," https://www.statista.com/statistics/1020956/android-app-releases-worldwide/, 2021, "Accessed May 2021".

[93] "Number of available applications in the google play store from december 2009 to december 2020," https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2020, "Accessed May 2021".

[94] "Google Play Policy Center — Impersonation," https://support.google.com/googleplay/android-developer/topic/9969539, 2021.

[95] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, 1977. [Online]. Available: http://www.jstor.org/stable/3033543

[96] "Terms and Conditions - Andromo," https://www.andromo.com/en/terms/, 2021.

[97] "How to put your app in Google Play | Andromo Support," https://support.andromo.com/i47-how-to-put-your-app-in-google-play, 2020.

[98] "AppsGeyser: Free App Creator & App Maker. Create Android Apps No Code," https://appsgeyser.com/, 2021.

[99] App privacy policy generator. [Online]. Available: https://app-privacy-policy-generator.firebaseapp.com/#

[100] Privacy policy. [Online]. Available: https://web.archive.org/web/20210613192231/http://myappterms.com/reader.php?lang=en

[101] "Permissions on android," https://developer.android.com/guide/topics/permissions/overview, "Accessed Jan 2022".

[102] H. Wang, J. Si, H. Li, and Y. Guo, "Rmvdroid: Towards a reliable android malware dataset with app metadata," in *Proceedings of the 16th International Conference on Mining Software Repositories*, 2019.

[103] "Shared User Id," https://developer.android.com/guide/topics/manifest/manifest-element#uid, 2021.

[104] "Verify Android App Links," https://developer.android.com/training/app-links/verify-site-associations, 2021.

[105] "Is your app installed? getInstalledRelatedApps() will tell you!" https://web.dev/get-installed-related-apps/, 2021.

[106] "Enable automatic sign-in across apps and websites," https://developers.google.com/identity/smartlock-passwords/android/associate-apps-and-sites, 2021.

[107] P. Kotzias, S. Matic, R. Rivera, and J. Caballero, "Certified pup: abuse in authenticode code signing," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2015.

Álvaro Feal is a PhD student working at IMDEA Networks Institute under Prof. Narseo Vallina-Rodriguez's supervision. His research revolves around analyzing privacy threats in the mobile and web ecosystem using static and dynamic analysis techniques as well as network measurements. He has published his research in different venues such as the IEEE Symposium on Security and Privacy, USENIX Security, ACM IMC, PETS Symposium, IEEE ConPro, and CPDP. He has received several awards such as the Distinguished Paper Award at Usenix Security'19 and the "Premio de Investigación en Datos Personales: Emilio Aced" in 2021 and 2022.

Julien Gamba is a PhD student in the Internet Analytics Group at the IMDEA Networks Institute. His research revolves around user's security and privacy in Android devices. In his work, Julien uses both static and dynamic analysis, as well as other techniques specifically designed to understand the behavior of mobile applications. Recently, Julien was the first author of the first large-scale analysis of the privacy and security risks of pre-installed software on Android devices and their supply chain, which was awarded the Best Practical Paper Award at the 41st IEEE Symposium on Security and Privacy. This study was featured in major newspaper such as The Guardian (UK), the New York Times (USA), CDNet (USA) or El País (Spain). Julien was also awarded the ACM IMC Community Contribution Award in 2018 for his analysis of domain ranking services, and was awarded the NortonLifeLock Research Group Graduate Fellowship, the Google PhD Fellowship in Security and Privacy and Consumer Reports' Digital Lab fellowship.

Aniketh Girish is PhD student working at the Internet Analytics group of IMDEA Networks Institute and Universidad Carlos III de Madrid advised by Prof. Narseo Vallina-Rodriguez. Aniketh empirically measure the privacy and security risks of smart home and Mobile ecosystem with a particular focus on consumer data protection and regulatory compliance. Aniketh has published his research in international peer-reviewed conferences such as USENIX Security'20.

Kaspar Hageman is a postdoctoral researcher at Aarhus University, Denmark. He obtained his PhD degree at Aalborg University in 2021 on the analysis of the domain name system, digital certificates and network traffic for security. His research interests further include applying machine learning to network security and more recently collaborative drone fleets. His work was awarded the IETF/IRTF Applied Networking Research Award in 2017.
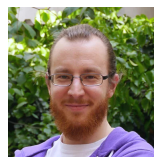

Jakob Bleier is a PhD student at the TU Wien Security and Privacy Research Unit under the supervision of Martina Lindorfer. He is working in the area of system security and is developing new methods and tools for measuring similarities between programs, as well as identifying the libraries used. His work has contributed to a publication presented at the AsiaCCS'22 conference.

Martina Lindorfer is an Assistant Professor at TU Wien, Austria, and a key researcher at SBA Research, the largest research center in Austria which exclusively addresses information security. Prior to starting her tenure track she spent two years as a postdoc at the University of California, Santa Barbara. Her research focuses on applied systems security and privacy, with a special interest in automated static and dynamic analysis techniques for the large-scale analysis of applications for malicious behavior, security vulnerabilities, and privacy leaks. Her research and outreach activities have been recognized with the ERCIM Cor Baayen Young Researcher Award, the ACM Early Career Award for Women in Cybersecurity Research, as well as the Hedy Lamarr Award from the City of Vienna.

Juan Tapiador is Professor of Computer Science at Universidad Carlos III de Madrid, Spain, where he leads the Computer Security Lab. Prior to joining UC3M he worked at the University of York, UK. His research interests include systems security, malware, privacy, and network measurements. He has served in the technical committee of conferences such as USENIX Security, ACSAC, DIMVA, ESORICS and AsiaCCS. He has been the recipient of the UC3M Early Career Award for Excellence in Research (2013), the Best Practical Paper Award at the 41st IEEE Symposium on Security and Privacy (Oakland), the CNIL-Inria 2019 Privacy Protection Prize, and the 2019 AEPD Emilio Aced Prize for Privacy Research. His work has been covered by international media, including The Times, Wired, Le Figaro, and The Register.

Narseo Vallina-Rodriguez is an Associate Research Professor at IMDEA Networks and a co-founder of AppCensus Inc. Narseo obtained his Ph.D. at the University of Cambridge and his research interests fall in the broad areas of network measurements, privacy, and mobile security. His research efforts have been awarded with best paper awards at the 2020 IEEE Symposium on Security and Privacy (S&P), USENIX Security'19, ACM IMC'18, ACM HotMiddlebox'15, and ACM CoNEXT'14 and Narseo has received prestigious industry grants and awards such as a Google Faculty Research Fellowship, a DataTransparencyLab Grant, and a Qualcomm Innovation Fellowship. His research in the mobile security and privacy domain has been covered by international media outlets like The Washington Post, The New York Times, and The Guardian and it has influenced policy changes and security improvements in the Android platform. Narseo's work has received in multiple occasions the recognition of EU Data Protection Agencies with the AEPD Emilio Aced Award (2019, 2020, and 2021) and the CNIL-INRIA Privacy Protection Award (2019 and 2021). He is also the recipient of the IETF/IRTF Applied Networking Research Award in 2016 and the Caspar Bowden Award in 2020.