



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

School of Computer Science and Statistics

Investigating data collection by Google Analytics for Firebase on an Android phone

Mark Connor

Supervisor: Douglas Leith

April 15, 2025

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Computer Science and Business

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at <http://www.tcd.ie/calendar>.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at <http://tcd-ie.libguides.com/plagiarism/ready-steady-write>.

Signed: Mark Connor _____ Date: 9/04/2025 _____

Abstract

This report investigates the data collection practices of Google Analytics for Firebase on Android devices. The focus is on how consent settings related to data privacy affect the transmission of personal identifiers to Google servers. Despite the widespread adoption of Google Analytics for mobile apps, its documentation is not comprehensive. In particular, it is uncertain how consent settings affect identifiers and how these identifiers link to users. To explore this uncertainty, a sample Android application was created that integrates Google Analytics. A man-in-the-middle attack was implemented using a tool called Mitmproxy to intercept and record encrypted network traffic between the app and Google servers. The recorded traffic was decoded by reverse engineering the Google Play Services APK to extract the encoding schema. Additional analysis was conducted on women's health apps to observe how Google Analytics handles sensitive data on real-world applications. The findings of this report reveal significant discrepancies between Google's documentation and actual data practices. Notably, identifiers such as the Google Advertising ID, Firebase Instance ID, and DSID cookie are collected even when consent settings suggest otherwise. In cases where consent mode is not implemented by the developer, Google defaults to the most permissive data collection behaviour. This means all sensitive identifiers are logged as if full consent was granted. This behaviour was consistently observed across real apps, including those with consent banners that failed to affect the data logged to Google servers. The results indicate that the burden of enforcing privacy falls heavily on app developers, while users often have little to no actual control over the data collected about them. This report highlights the need for improvements to consent management by the service Google Analytics. It also generally contributes to the ongoing discourse on ethical data collection and digital consent.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Doug Leith, for his guidance throughout this project. His feedback and support challenged me, and as a result, I am proud of the work I have produced. It couldn't have been done without him.

I would also like to thank my family for their love and belief in me throughout my studies. Their support meant the world to me and kept me motivated throughout my four years at Trinity College.

Contents

1	Introduction	1
1.0.1	Problem Statement	2
1.0.2	Objectives	2
2	Related Work	3
2.1	Literature Review	3
2.2	Prevalence of Advertising in Mobile Applications	4
2.3	Man-in-the-Middle Attack	5
2.4	Protobuf	5
3	Google Analytics For Firebase	7
4	Experiment Design	10
4.1	Measurement Methodology	10
4.1.1	Sample Firebase App	10
4.1.2	Man-in-the-Middle Attack Setup	12
4.1.3	Reverse Engineering Google Play Services	14
4.1.4	Decoding Protobuf Traffic	16
4.2	Google Analytics Settings	17
4.2.1	Consent Mode	17
4.2.2	Unspecified consent state	18
4.2.3	Consent Settings	19
4.2.4	Women’s Health Apps	20
5	Experimental Results	22
5.1	Consent Mode	22
5.2	Unspecified consent State	24
5.3	Consent Settings	26
5.4	Women’s Health Apps	27
5.4.1	Discussion	29

6 Conclusion	31
6.0.1 Future Work	31

List of Figures

3.1	Graph of active users over time from Analytics dashboard.	8
3.2	Graph of countries with active users from Analytics dashboard.	8
4.1	Prompt to choose a favourite food that appears upon the first launch of the sample Google Firebase app.	11
4.2	Screen A of the sample Google Firebase application.	12
4.3	Command-line prompt for recording network traffic.	13
4.4	Function within the Google Play Services APK that contains the protobuf schema.	15
4.5	Example of identifiers found in the response of a network connection to Firebase.	17
4.6	The meaning of various consent state values visualised.	18
4.7	Summary of the consent settings available for Android users and app developers.	20
5.1	Table of consent mode tags.	22
5.2	DSID cookie recorded in Google Play Services method.	23
5.3	File on the client device where the DSID cookie is saved.	23
5.4	Identifiers logged to Google with unspecified consent state.	25
5.5	Identifiers logged to Google when all consent types are enabled.	25
5.6	Identifiers collected under different consent configurations with Google Signals enabled.	26
5.7	Identifiers collected under different consent configurations with Google Signals disabled.	27
5.8	Data collection of popular women's health apps.	28
5.9	Identifiers collected by Google on apps with a consent banner.	29

1 Introduction

Over the last 25 years, data has become one of the world's most valuable resources [1]. Corporations use data-mining methods applied to massive databases of personal information to plan advertising and new product development and to implement targeted-advertising campaigns [2]. Companies have also developed tools to track user behaviour and personalize their experience using this data [3].

Google is one of the world's largest data-focussed corporations [4]. It has many products that collect data, a notable one is Google Analytics. Google Analytics for Firebase measures app usage and user engagement for mobile apps [5]. As people continue to use their mobile devices more [6], Google products are responsible for managing more sensitive data. Despite managing lots of data, Google Analytics for Firebase documentation is poor. Research has shown that mobile applications often collect a large set of sensitive personal data and transmit it to remote servers without user consent [7]. However, it is unclear how Google handles information from third-party apps and how the collection of this data changes when users' consent settings change.

Despite its widespread use, there is remarkably little public documentation about the nature of the data Google Analytics collects. In particular, it is not well understood how specific identifiers link the analytics data to an individual person or device. The aim of the project is to investigate how data is transmitted by Google Analytics and how the data transmission is affected by app and dashboard settings. It is important to clarify that these identifiers are data points logged to Google servers. Third-party applications, from which the data is collected, cannot see or access these identifiers. This means that the primary concern is not the data collection practices of third-party apps but rather Google's control over this data. Of particular interest is discovering the effect of user consent preferences on the data Google collects. The project used network traffic analysis to capture and interpret the data collected by Google from Android applications.

The findings of this report reveal that Google Analytics' data collection mechanisms may not fully align with their documentation. For example, Google claims that the DSID cookie is used to identify a signed-in user to respect their ad personalization settings [8]. However, the report's findings show that user ad personalization settings may not be respected. On

top of this, there are several cases where user data is transmitted even when privacy settings suggest otherwise. Thus, this report raises serious concerns about transparency, informed consent, and compliance with data protection laws such as the General Protection Regulation (GDPR).

Understanding Google's data collection practices is crucial because personal data protection is a growing political and social concern [9]. Governments and regulatory bodies around the world are increasingly scrutinizing corporate data collection methods, with calls for stricter enforcement of privacy laws [10]. This research contributes to that discussion by exposing discrepancies in Google's approach and reinforcing the need for stronger accountability in the digital economy.

1.0.1 Problem Statement

Users and app developers rely on consent settings to control how personal data is collected and shared. However, poor documentation raises doubts about whether these settings serve their intended purpose. This lack of transparency poses risks to user privacy and could potentially conflict with data protection laws. Thus, this report seeks to answer the following research question:

To what extent do consent settings affect the personal data collected by Google Analytics from Android applications?

1.0.2 Objectives

To answer my research question, my project tracks and analyses personal user information sent to Google servers from an Android phone. The analysis aims to discover how data is linked to individual users and how data collection changes when consent settings are changed. To achieve this, the following three objectives were defined:

1. Create a sample Google Firebase application to record and analyse network traffic while changing consent settings.
2. Reverse engineer Google Play Services to decode data transmission and explore how exactly data is linked to a user.
3. Test real Android applications that collect sensitive information.

2 Related Work

2.1 Literature Review

Academic research has demonstrated that many companies collect users' personal data without explicit consent. This literature review discusses research on network traffic analysis, mobile advertising and apps that collect sensitive information.

In his 2021 paper [11], Leith explains the process of decrypting HTTPS connections. This process involves a proxy called *Mitmproxy*. This proxy acts as an intermediary between the app and target server while logging the traffic. This same implementation of collecting network traffic is used in my project. In another paper [12], Leith furthers research into Android predatory data collection practices. Here, he suggests that Google may violate GDPR legislation based on the data that is being collected by the Google Dialer and Google Messages Apps. This paper explains how uniquely identifiable information is being collected by these applications. It primarily focuses on the collections made via the Google Play Services Clearcut logger service but also touches on those made by Google Firebase. My research expands on this paper by examining how Google Firebase links collected data to an individual user. Neither of these studies consider the data sent to Google through third-party applications. This is a distinctive feature of my project that distinguishes it from previous research.

There has been a significant amount of research in data collection for mobile advertising. In his 2012 paper [13], Grace shows that ad libraries connected to Android applications collect sensitive information, much of which is difficult to justify. It discovers ad libraries that retrieve users' phone numbers, call history, SMS messages and more. This report may be somewhat out-dated because ad libraries have evolved significantly since 2012. However, it provides important context to the predatory history of mobile ad tracking. A more recent paper [14] finds that 379 of 1,134 studied apps do not comply with their respective privacy policies. This finding highlights the importance of investigating how apps can control user data collection. It also gives rise to legal concerns of data transmission on Android apps. The topic of possible legal concerns is also mentioned in Leith's 2022 paper, which is particularly relevant to my project.

In her 2022 paper [15], Alfawzan highlighted the privacy concerns posed by women's health apps. She finds that out of the apps studied, only 52% requested consent from users and 13% collected data before obtaining consent. This report provides useful insight into the privacy concerns of these apps. In another paper [16], Cao et. al builds on this insight by discussing how data from these apps can be used to detect abortions and pregnancies. It argues that the recent overturning of a constitutional right to an abortion in the US means that the collection of this data puts women at a risk of being prosecuted. My project uses this argument as motivation for investigating women's health apps to see if personally identifiable information is being sent via Google Analytics. If so, this would strengthen concerns posed by Cao's paper.

2.2 Prevalence of Advertising in Mobile Applications

Advertising is a fundamental component of the mobile app ecosystem. Reports show that over 95% of Android apps are free [17]. Although they are free, these apps generate nearly 98% of app revenue [18]. In-app advertising is a common approach for monetizing free mobile apps. For example, Instagram is a free social media app that used an advertising revenue model to generate an estimated \$66.9 billion in revenue in 2024 alone [19].

Advertisers pay mobile apps to display their ads to individuals with specific traits. This is called targeted advertising. Targeted advertising is based on estimating these user traits, some of which can be sensitive, such as political preferences, financial status, gender, etc. Targeted advertising is facilitated by advertising trackers, which collect user data to serve personalized ads. A significant portion of mobile applications use advertising trackers. A study found more than 75% of over 300 apps contained the signatures of ad trackers [20]. These trackers gather various forms of user data, which are then used to profile users and optimize ad targeting. Advertising trackers are typically embedded in Software Development Kits (SDKs). The widespread adoption of SDKs has been encouraged by the tracker companies that advertise them. This raises concerns about user privacy, as these services often operate without clear user awareness. For example, Google offers their Google Firebase SDK, which collects data that helps understand user behaviour [5].

Google's business model relies on advertising. More than 80% of Alphabet's (Google's parent company) revenue comes from Google Ads [21]. With 62% of all global internet traffic now coming from mobile devices, much of their revenue relies on mobile advertising. Google's advertising business has been put under scrutiny in recent years. An American antitrust lawsuit posed against Google claims that Google's ad-tech business is an illegal monopoly [22]. Other reports suggest that Google has limited advertisers' control [23]. In light of these criticisms, it seems reasonable to investigate how Google's mobile advertising ecosystem functions. The investigation of this project involves a part of Google's advertising

ecosystem, Google Analytics for Firebase.

2.3 Man-in-the-Middle Attack

The data transmitted by Google Analytics is encrypted. The first step to analysing this data was to decrypt it. For this, a man-in-the-middle (MITM) attack was used. A MITM attack is a technique to intercept network traffic. This attack was implemented using a proxy called *Mitmproxy*. This proxy intervenes between the communication of the app and target server [24]. It pretends to be the remote server to the app and pretends to be the app to the remote server. This tricks both parties into sending the data to it instead of directly to each other. Normally, the user's phone connects to the internet using Wi-Fi or mobile data. However, a VPN configures all internet traffic to be sent through the remote server. Both the app and the target server are unaware of the existence of the proxy and believe they are communicating solely with each other. Once the proxy has received the data from the app, it can be recorded or altered before it is sent to its destination.

Hypertext transfer protocol secure (HTTPS) is a secure protocol for data transmission that has been adopted by most Android applications. Without HTTPS, the communication between a Client and a Server would be in plain text. This means that sensitive information like the user's name, age, and password can be read by anyone that is able to intercept it. HTTPS is designed to solve this problem. It includes a protocol called Transport Layer Security (TLS), which encrypts the data sent between users. If the encrypted data gets intercepted by a hacker, all the hacker can see is muddled data. There are several steps to the TLS handshake. Firstly, the browser establishes a connection to the server. Then, the Client and Server begin to communicate with each other. A key part of the initial communication in the TLS handshake is the Certificate Authority (CA) system, which verifies the identify of websites. This system prevents MITM attacks because an unauthorized proxy (like *Mitmproxy*) would not have a valid certificate signed by a trusted CA.

2.4 Protobuf

Decrypting the data is not enough for analysis, as the decrypted data remains encoded and unreadable in its raw form. Google encodes data using protocol buffers (protobufs) for more efficient transmission [25]. Unlike popular formats such as JSON and XML, protobufs use a compact binary format, making them faster and more lightweight. However, this also means the data is not human-readable by default. To decode the data, the protobuf schema must be understood, as it defines the structure and meaning of the encoded information. A protobuf schema specifies the data types, field names, and field numbers used in the binary-encoded messages. Without access to this schema, it is difficult to accurately

interpret the raw protobuf data, as field numbers alone do not provide meaningful context. The protobuf schema is embedded in the Google Play Services app. This app serves as the central framework for many Firebase and Google services on Android devices [26]. By reverse engineering the Google Play Services app, this protobuf schema can be found and used to decode the protobuf message. This allows for analysis of the data transmitted by Google.

3 Google Analytics For Firebase

Google Firebase is a mobile application development platform. It provides many services for backend development such as authentication, real-time databases, cloud storage and hosting [27]. Firebase also integrates with various Google services, one of these being Google Analytics. Google Analytics is a service that analyses user engagement, retention, and behaviour on a website to provide insights for businesses [28]. Google Analytics for Firebase is the version that integrates with Google Firebase. This version analyses information from mobile applications, instead of websites. Originally called Firebase Analytics, the service was renamed to Google Analytics for Firebase in 2017 to unify app analytics under the Google Analytics brand [29]. This renaming can cause confusion when researching or discussing the service. For simplicity, this report uses the term *Google Analytics* to refer specifically to Google Analytics for Firebase, the mobile app analytics version.

Google Analytics is one of the most widely adopted analytics solutions for mobile applications. It is reported that 58% of Android applications use Google Analytics [30]. This service is part of a larger trend in mobile development, where analytics software development kits (SDKs) are embedded within apps to track user interactions. These services are designed to collect user behaviour data and use it for targeted tracking purposes. Other popular analytics SDKs include Facebook Analytics, Fabric, Flurry Analytics and Amplitude.

The latest iteration of Google Analytics, Google Analytics 4 (GA4), introduced a significant change from session-based to event-based analytics [31]. Essentially, all user interactions within an app are recorded as events [32]. Events capture details about the user interaction such as items searched for, items bought, etc. This change helps developers better understand how users engage with their app, rather than collecting metrics that can be difficult to obtain insights from such as number of page views, or how long a user spends on the app. This allows GA4 to focus on interactions that convey behavioural information. Every time a user interaction occurs, an event is recorded and logged to Google's back end servers. This information can then be used to profile user traits. Google Analytics uses this user information to provide businesses with details about how their users are interacting with their app. Figures 3.1 and 3.2 show some of the analytics displayed on the Home page of the Google Analytics dashboard.

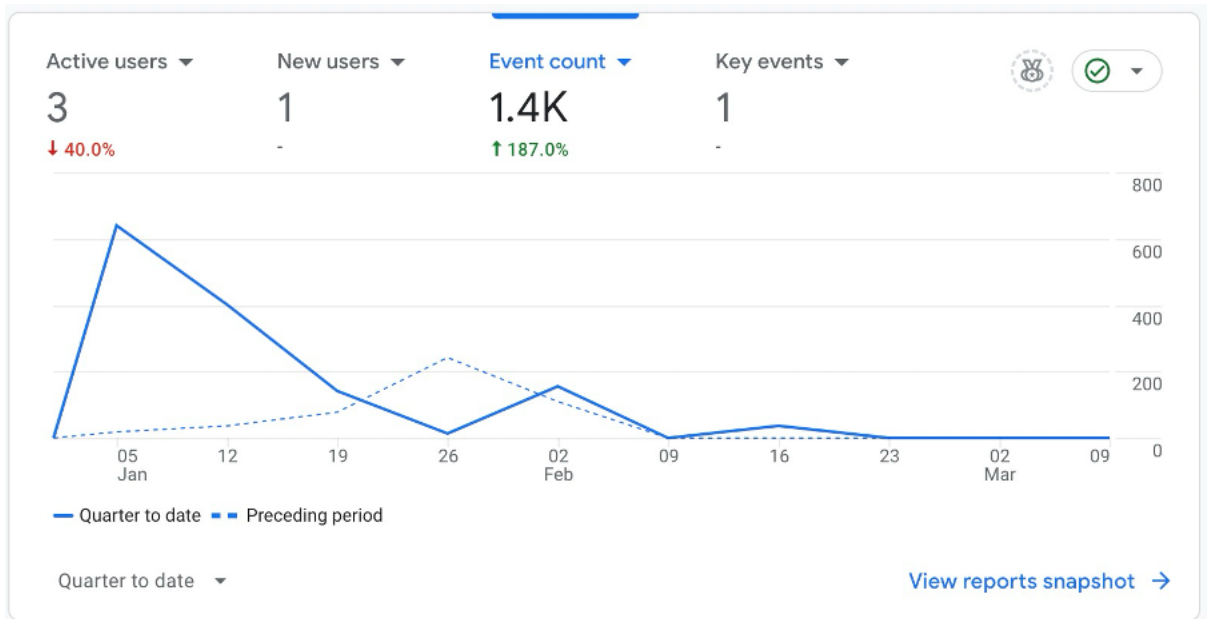


Figure 3.1: Graph of active users over time from Analytics dashboard.

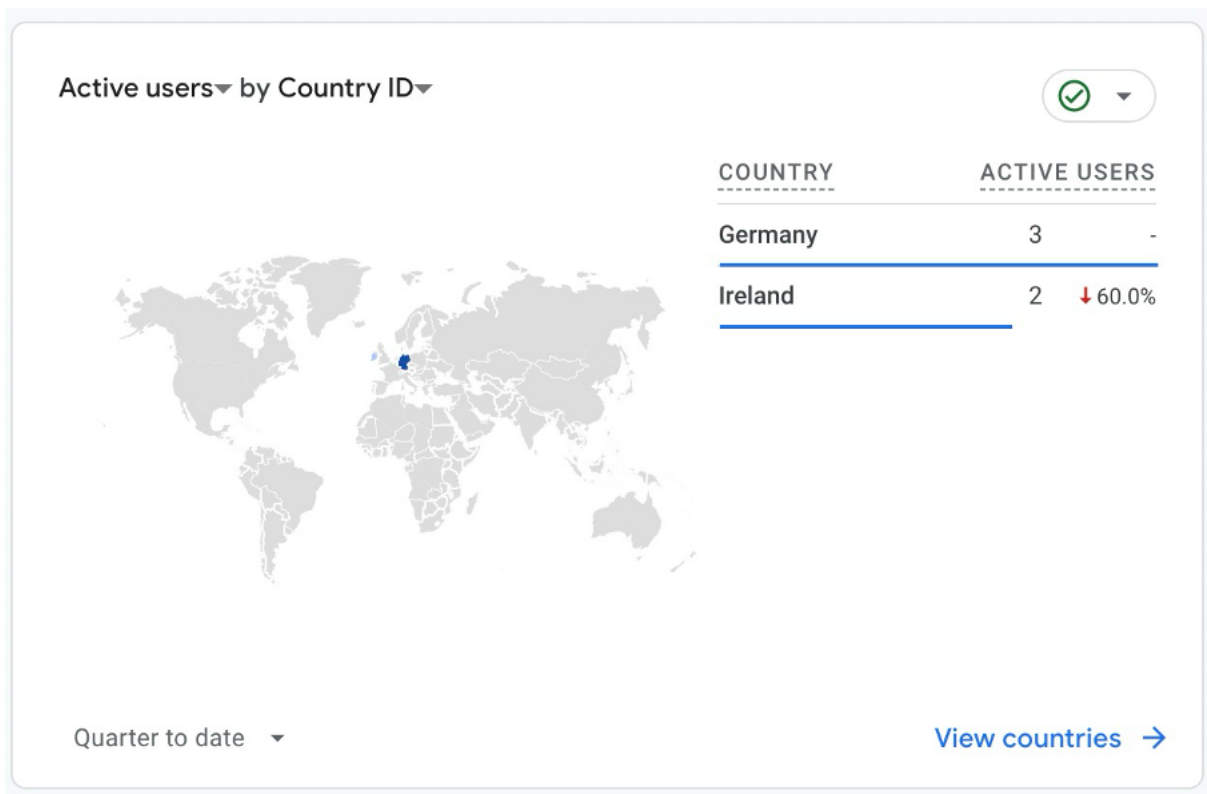


Figure 3.2: Graph of countries with active users from Analytics dashboard.

Google Analytics offers various settings that allow app developers to manage how data is collected on their app. Consent mode is a tool provided by Google Analytics for developers to manage users' consent preferences. It lets developers communicate each user's consent

preferences to Google [33]. Google provides documentation that explains how to set up consent mode on a Google Firebase app [34].

4 Experiment Design

4.1 Measurement Methodology

4.1.1 Sample Firebase App

To explore the extent of user and developer control over data transmission, it was necessary to configure consent settings within a Firebase application that integrates Google Analytics. Consent settings are preferences that determine the types of personal data the app can collect and share with third parties, such as Google Analytics. There are consent settings for both users and app developers. These settings are explained in Section 4.2.3. A sample application that provides the code for integrating Google Analytics was sourced from the official Firebase documentation [35].

Access to Firebase and related services required the creation of a Google account. Within a Google account, an Analytics account can be made, which provides access to the Analytics dashboard. The Analytics dashboard provides data collection preferences that the app developer can manage. In particular, they can use the dashboard to enable Google Signals. Google Signals is a feature that allows Google to track user data from sites and apps and use that data for advertising purposes [36]. Google claims that it is designed for users who have consented to Ad Personalization [37], which allows Google to customise the user's ad experience.

The Google Firebase documentation provides instructions for adding Firebase to an Android application [38]. This involves downloading the `google-services.json` file and placing it in the module root directory. The documentation also recommended using Android Studio for application development and testing, so this was installed and used as the app's development environment. This approach ensured that the Firebase application was correctly linked to Google Analytics, enabling a controlled investigation between a developer's data collection configuration and actual data collection behaviour.

The sample application was accessed on a Google Pixel 4a, which was connected to a laptop via USB. Running the program through Android Studio launched the application on the

device. On the initial launch of the application, the user is prompted to select a favourite food out of a list. This choice is subsequently logged to Google Analytics as a User Property [39]. Figure 4.1 displays this prompt as it appears upon the first launch of the app.

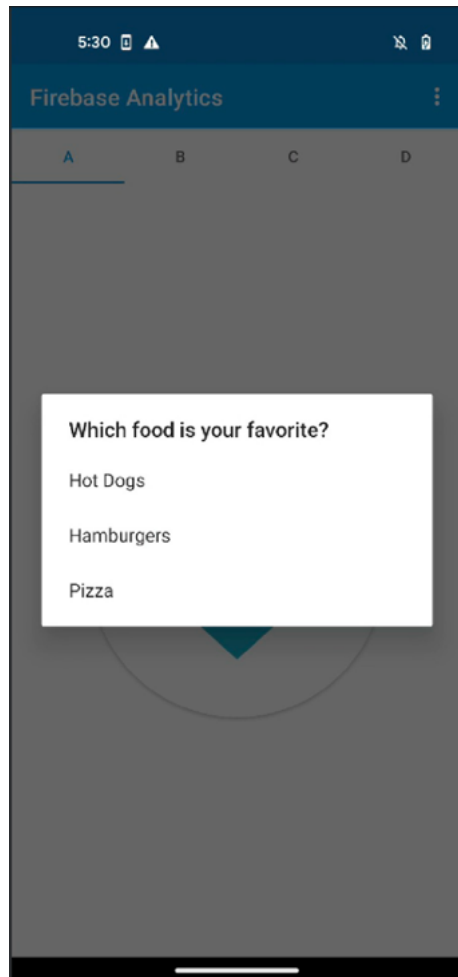


Figure 4.1: Prompt to choose a favourite food that appears upon the first launch of the sample Google Firebase app.

Generating network traffic from the application presented challenges, as events are batched and uploaded approximately once per hour [40]. This means that consistently capturing network traffic would take approximately an hour of recording. However, Firebase offers a debug mode [40], which when enabled, logs events as they happen in real-time. This eliminated the need to record traffic over a prolonged period.

The sample application consists of four screens labelled A, B, C and D. Screen A is shown in Figure 4.2. Each screen displays a symbol and users can navigate between them. Each time the user moves screen, a connection is made to Firebase servers and an event is logged. Event information includes the screen the user moved to and for how long they were on the screen. Along with information about the event, information about the user is also sent. This

includes data that can uniquely identify the user's device or Google account. These uniquely identifying data points (identifiers) are the focus of the network traffic analysis.

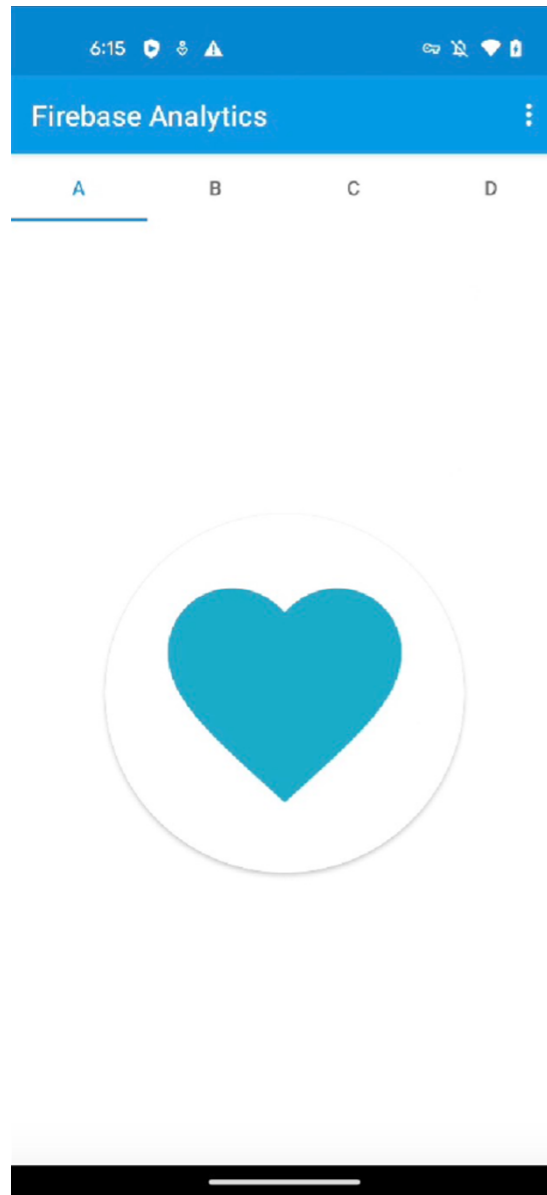


Figure 4.2: Screen A of the sample Google Firebase application.

4.1.2 Man-in-the-Middle Attack Setup

The Client device used for the MITM attack was the Google Pixel 4a running Android 13 rooted using Magisk v27.0. The attacking device used was an Apple MacBook Air operating on macOS Sonoma version 14.3.1, with *Mitmproxy* version 11.1.0.

To initiate the MITM attack, the Client device's CA system had to be surpassed. For this, *Mitmproxy* was used. *Mitmproxy* was installed as a trusted certificate on the device, meaning the device would recognize it as a legitimate CA and allow it to intercept HTTPS traffic. However, simply installing the CA certificate was not sufficient. An update to trusted

certificates on Android in 2016 meant that user installed certificates would not be trusted by mobile apps by default [41]. To bypass this restriction, the *Mitmproxy* certificate had to be converted to a system certificate. To convert the certificate to a system certificate, the Magisk app was used [42]. Magisk provides an easy way to access features of your phone that are blocked by phone providers. One of the modules that can be installed using Magisk is called 'TrustUserCerts' [43]. This module converts all user certificates to system certificates, so they will automatically be trusted by mobile apps.

A command-line version of *Mitmproxy* called `mitmdump` was used. The command line prompt used to record traffic is displayed in Figure 4.3.

```
mitmdump -p 8081 --mode transparent --showhost --ssl-insecure -w file_name
```

Figure 4.3: Command-line prompt for recording network traffic.

The option `-p 8081` instructs *Mitmproxy* to listen on port 8081. This is the point where network traffic is to be intercepted.

The option `-mode transparent` sets *Mitmproxy* to transparent mode. This is necessary because most modern Android applications bypass HTTP proxy settings. They ignore the proxy settings and connect to the internet directly, instead of passing through the proxy. Notably, while using a regular proxy, the client must be configured so that the traffic destination is the proxy. A transparent proxy never changes the destination of the network traffic sent from the client. Instead, traffic is directed at the network layer. Traffic was redirected at the network layer by connecting the client device to a VPN. The VPN routes all traffic through the proxy server. It essentially acts as a custom gateway for all network connections. At this gateway, *Mitmproxy* logs the traffic.

The option `showhost` displays the Host header for each connection while traffic is being recorded. This feature made it possible to identify traffic destinations in real time. In Section 4.2.4 this feature was used to quickly establish if an app integrates Google Analytics for Firebase without having to analyse the traffic. If a connection to the `app-measurement.com/a` endpoint appears, the app integrates Google Analytics.

The option `--ssl-insecure` disables verification of upstream TLS certificates. This makes it possible to intercept HTTPS traffic. If enabled, many connections would fail due to TLS security measures.

Lastly, `-w file_name` logs the traffic to a file called "file_name". This file can later be decoded and analysed.

4.1.3 Reverse Engineering Google Play Services

Google Analytics is dependent on Google Play Services, meaning it can only run on devices that have Google Play Services installed [44]. The definitions of protobufs sent by Analytics are embedded in the Google Play Services Android Package (APK). An APK is the file format that Android uses to distribute and install apps. Similar to a ZIP file, it can contain metadata and multiple files that are compressed for ease of distribution. Analysing the code in the Google Play Services APK serves two objectives:

1. To find and interpret the protobuf schema.
2. To investigate how the data sent within the protobuf message is linked to users' identities.

The Google Play Services APK was reverse engineered using Jadx. Jadx is a tool for producing Java source code from APK files [45]. It converted the Google Play services APK to Java code for manual inspection. The Google Play Services APK was extracted from the client device to the laptop using ADB. ADB is a command-line tool to communicate with a device [46]. Extracting the file involved locating the file path of the Google Play Services APK on the device and using ADB commands to transfer it to the local machine. USB debugging was enabled on the device to establish the necessary connection for data transfer. Once extracted, Jadx decompiled the APK, which allowed for further examination into the structure of protobuf messages and Google Analytics data collection mechanisms.

Google Play Services uses code obfuscation techniques to prevent reverse engineering of the application. The most impactful obfuscation technique used was identifier remaining. This means replacing meaningful class, method and variable names with short, non-descriptive labels (e.g. `aaqq.a`, `gfbp.h`). This was impactful because it made critical classes related to Google Analytics become obfuscated, meaning there were no contextual clues about their functionality. On top of this, the APK contains code that is completely unrelated to Google Analytics as well as legacy code that is no longer active. These reasons as well as the sheer size of the codebase, made it difficult to find the protobuf schema. To make this process easier, the `grep` command-line tool was used to search for key identifiers such as `"DSID"`, `"cookie"`, `"google_ad_id"`, and `"core_platform_services"`. This approach facilitated the pinpointing of lines of code where these key words appeared. Importantly, these key words are identifiers that relate to protobuf messages. Thus, it was expected that they may appear in close proximity to the protobuf schema or have some connection to it within the codebase. Therefore, searching for these key identifiers would make it possible to locate the protobuf schema. The search led to a class that manages the protobuf messages that send Google Firebase event data. This class contains the function displayed in Figure 4.4. This function includes a specific line of code that defines the connection between field numbers and their corresponding names, types and attributes. This line of code represents the protobuf

schema. The binary string within this line of code represents the structure of the data. The accompanying object list matches variable names with their respective field numbers.

```

public final Object dynamicMethod(int i, Object obj) {
    if (i == 0) {
        throw null;
    }
    int i2 = i - 1;
    if (i2 == 0) {
        return (byte) 1;
    }
    if (i2 == 2) {
        return new fvju(instance, "\u0004\u0005\u0000\u0001\u0001\u0005\u0005\u0000\u0001\u0000\u0001\u0001b\u0002qj\u0000\u0003\u0001\u0004\u0002\u0005c\u0003", new Object[]{"b", c.a, bytn.class, "d", "e", "f", "g"});
    }
    if (i2 == 3) {
        return new FirebaseEventProto();
    }
    if (i2 == 4) {
        return new bytk();
    }
    if (i2 == 5) {
        return instance;
    }
    if (i2 != 6) {
        throw null;
    }
    Parser parser = h;
    if (parser == null) {
        synchronized (FirebaseEventProto.class) {
            parser = h;
            if (parser == null) {
                parser = new fvhm(instance);
                h = parser;
            }
        }
    }
    return parser;
}
}

```

Figure 4.4: Function within the Google Play Services APK that contains the protobuf schema.

As well as locating the protobuf schema, the grep search for key identifiers was used to explore the connections between these identifiers and user identities. For example, locating the occurrences of “DSID” within various classes provided context for how this identifier was accessed, and transmitted. Given more time, further analysis could be done on this topic.

4.1.4 Decoding Protobuf Traffic

Once the protobuf schema was found, the next step involved decoding the protobuf messages transmitted by Google Analytics. Understanding these messages is crucial, as they contain the structured data that Google Analytics uses to track user interactions. The protobuf schema provides the necessary details to interpret the data accurately.

With the protobuf schema as a guide, the binary string was interpreted using a systematic decoding approach. A protobuf decoding tool, provided by the project supervisor, was used [47]. This decoding tool breaks down the binary string into its component parts. The remaining segments of the string represent the field numbers and associated data types, which matched the definitions provided in the schema. The field numbers in the protobuf message were then matched with their corresponding variables in the schema. Once field numbers were mapped to variables, the data contained in the protobuf messages was decoded and ready for analysis.

To analyse the protobuf message, the MITM recording had to be sifted through to find the specific connection relevant to the Google Firebase sample application. This was a challenging task because network traffic constantly flows between a user's phone and various servers. Thus, each recording contained many network connections, most of which were not relevant to this research. The relevant connections were the ones sent from the sample Firebase application to Google servers. It was crucial to develop a systematic approach to identifying these specific connections among the large volume of recorded network traffic. Each network connection consists of a request and a response. The app sends a request to the server, which processes it and sends back a response. The request body contains the actual data sent to the server from the app, which is the primary focus of the project. In the case of the sample Firebase app, data is sent from the app to Google servers when an event is recorded. Along with information about the event, there is also data about the user, some of which can be used to identify them. Figure 4.5 shows an example of this user data sent from the Firebase app to Google servers. It was outputted in the terminal and decoded into a human-readable format using a protobuf decoder.

```

upload_timestamp_millis: 1737379106624
event_timestamp: 1737379106118
bundle_end_timestamp: 1737379106118
previous_bundle_end_timestamp: 1737379105137
operating_system: "android"
operating_system_version: "13"
device_model: "Pixel 4a"
user_default_language: "en-us"
app_store: "manual_install"
package_name: "com.google.firebase.quickstart.analytics"
app_version: "1.0"
gmp_version: 108021
uploading_gmp_version: 244433
google_ad_id: "39415d30-611b-4a8f-9781-48cce8b388ab"
appInstanceId: "821d355b367f60282c98c9405424ea41"
dev_cert_hash: 6835378825754193913
daily_conversions_count: 2415
gmp_app_id: "1:719602967806:android:192860c5a2293b2d2431af"
previous_bundle_start_timestamp2: 1737379105137
service_upload: true
firebase_instance_id: "cgkOWpnhRqutbKYBo-un2t"
app_version_int: 1
config_version: 1737218313081256
M: 23181810
M: 23181696
dynamite_version: 130
consent_state: "G111"
google_signals: "google_signals"
target_os_version: 35
consent_diagnostics: "13333"
is_dma_region: true
core_platform_services: "-"
delivery_index: 739

```

Figure 4.5: Example of identifiers found in the response of a network connection to Firebase.

Each Firebase app has an App ID which is always sent in the protobuf message when a Google Analytics event is logged to Google servers. The Firebase App ID can be found on the Google Analytics dashboard under App stream details. In the network traffic, this ID can be found in the variable “gmp_app_id” as seen in Figure 4.5. A match between the ID on the Google Analytics dashboard and the ID in the network traffic confirms the connection originated from the sample Firebase application. To determine the connection was being logged to Google servers, network traffic was analysed for requests to the `app-measurement.com/a` endpoint, which Firebase uses to log user events. By combining these two criteria, matching app identifiers and detecting event logging requests, it was possible to systematically identify all relevant connections.

4.2 Google Analytics Settings

4.2.1 Consent Mode

Consent Mode allows app developers to set default consent states and update consent states based on user preferences. A consent state consists of four consent types. The consent

types are `ad_storage`, `ad_user_data`, `ad_personalization`, and `analytics_storage`. These consent types can be either granted or denied (i.e. set to “true” or “false”). In theory, whether they are granted or denied should influence the data being collected. However, it is unclear how these consent types affect the data transmitted by Google. To investigate this, network traffic was recorded under different consent configurations. Each consent type was individually enabled while keeping all others disabled, and these results were compared against a control case where all consent types were set to false. This approach allowed for a detailed examination of which identifiers were added or modified by each consent type. Specifically, the presence and persistence of the Firebase Instance ID, Google Ad ID and DSID cookie were examined. These three identifiers were chosen in particular because they play a critical role in tracking users, as discussed in Section 4.2.1, which outlines the results of these measurements. Identifiers are logged by Google Analytics to Google back-end servers. Thus, the examination of these identifiers addresses how Google can use them to track Android users. The third-party app that Google records the traffic from does not have access to these identifiers. Thus, the third-party apps data collection is not a concern for this particular examination.

4.2.2 Unspecified consent state

One of the data points included in the event protobuf message is the consent state. Essentially, the consent state signifies the current state of user consent [48]. The value of the consent state always begins with “G1” and is followed by two digits that are either 1 or 0 (granted or denied). The first of these two additional digits indicates whether the consent type `ad_storage` has been enabled or disabled. The second additional digit indicates whether `analytics_storage` has been enabled or disabled. Figure 4.6 illustrates what the consent state values signify.

CONSENT STATE	<code>Ad_storage</code>	<code>Analytics_storage</code>
“G100”	Denied	Denied
“G111”	Enabled	Enabled
“G101”	Denied	Enabled
“G1--”	Unspecified	Unspecified

Figure 4.6: The meaning of various consent state values visualised.

In their documentation, Google asserts “By default, no consent mode values are set” [49]. While this is technically true, a reader may assume that this means that consent mode values are denied. The reality is that if Consent Mode is not implemented by the app

developer, Google defaults the app to an “unspecified” consent state, where the consent state value “G1-” is logged. There is no documentation for this unspecified consent state. Subsequently, there is little understanding of the behaviour in this state. In particular, it is not clear what data Google allows to be logged to their servers when an app is in the “unspecified” consent state.

Consent Mode does not provide a consent banner to obtain the consent preferences from users [33]. Instead, it puts the responsibility on the app developer to implement a consent banner to collect users’ consent preferences. Then, the developer must update the consent states based on the user preferences to inform Google of the user’s decisions. This is a substantial responsibility to put on every app developer using Google Analytics. The unspecified consent state is important to understand because many developers using Google Firebase may neglect to implement consent mode. Thus, many apps will behave according to this “unspecified” state, for which there is no documentation.

Investigating the “unspecified” consent state involved creating another sample Firebase app using the steps outlined in Section 4.1.1. This new app made it possible to record the first instance the app was run. This ensured that no prior consent state had been set and that the app’s initial communication with Google servers could be observed. This new sample app was created in the same Google Firebase project as the original sample app. Thus, there was concern that the consent state would carry over from the last sample app, which had enabled Consent Mode. To confirm the consent state did not carry over, the prompt that asks the user to choose a favourite food was observed. The first time the app is run, the user is given this prompt. In every subsequent app event, the user’s choice of food is logged to Google Analytics. Similarly, the consent state of the app is logged to Google Firebase with every app event. Therefore, the presence of the prompt to choose a favourite food confirmed that no consent state was carried over from the other sample app.

4.2.3 Consent Settings

Google provides settings for Android users and app developers that are related to data collection. For Android users, there are privacy settings that are supposed to provide a way for them to have some control over the privacy of their data. They can configure these settings from the Settings app on their device. The Ad privacy section in Settings allows users to protect their identify and gives them more control over how advertisers choose which ads to shown them [50]. If logged into a Google account, users can also change how their data is collected via their Google account settings. In the “Data & privacy” section of Google account settings, users can take a Privacy Checkup which gives them options to choose which of their online activity is saved. This option allows them to personalize their Google experience. Similarly, app developers can configure how Google collects data from

their app. The configurations investigated involved Google Signals and Consent Mode. The results compare Google Signals being enabled versus disabled, as well as different combinations of the four consent types provided by Consent Mode. A diagram of data collection settings for Android users and app developers is shown in Figure 4.7. Combining these settings for both users and developers led to a significant sample of recordings to use to analyse data practices.

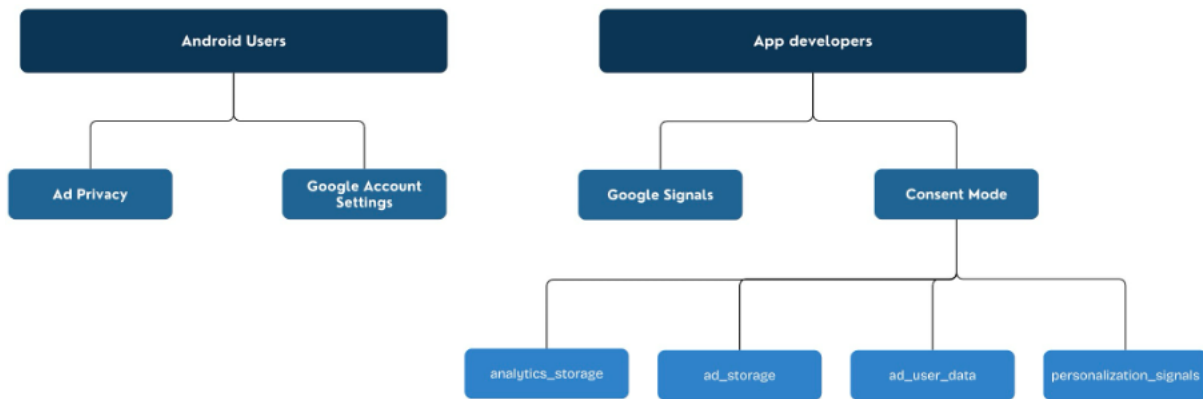


Figure 4.7: Summary of the consent settings available for Android users and app developers.

4.2.4 Women’s Health Apps

Investigating how Google Analytics operates in sensitive applications provides a real-world application of this project’s findings. Apps that handle health-related data present a higher risk to user privacy. This makes it critical to understand whether consent settings limit data collection in these apps. The measurement of sensitive apps will help evaluate the impact of Consent Mode in scenarios where improper data handling could have significant consequences.

Women’s health apps were the chosen category of sensitive apps. They were chosen because they contain highly sensitive data. Data such as fertility information, pregnancy status, and symptoms related to reproductive health are all commonly recorded. The data collected by these apps could potentially be used to detect an abortion. In countries where abortions are illegal, research suggests that such information could be incriminating [16]. If users can be uniquely identified, there is a risk that individuals could face legal consequences based on their private health data. Thus, these apps serve as a strong example of how poor consent management practices can pose serious risks to user privacy and safety.

Eleven apps were chosen as a sample from a list of the most downloaded women’s health apps of 2024 [51]. For each app in the sample, the techniques for network traffic analysis outlined in Section 4.1.2 were used to analyse network traffic. Recording traffic involved using the app until events that are logged to Google servers were generated. Firebase logs many events automatically. These include events when the user transitions to a different

screen on the app. Therefore, it was not difficult to generate events across different apps in the sample. While recording the network traffic, it was possible to identify when an Analytics event was logged because the `showhost` option within the `mitmdump` tool outputs the destination of the connection in real time. A connection to the endpoint `app-measurement.com/a` meant that an event had been logged to Google servers.

Some of the apps in the sample provided a consent banner, which allows the user to manage how their data is collected. For the apps that provided consent banners, two additional network traffic recordings were conducted.

1. Accepting all permissions in the consent banner.
2. Denying all permissions in the consent banner.

These additional recordings were conducted to discover the effectiveness of these consent banners on managing the data sent to Google servers. If the consent banners on these apps do not implement Consent mode correctly, Google may be collecting data from the user that users did not consent to being collected.

5 Experimental Results

5.1 Consent Mode

Figure 5.1 presents the four consent types and the identifiers that were modified or added when the corresponding consent type was enabled. These identifiers were analysed based on their persistence and their potential to uniquely identify a user.





CONSENT TAGS	IDENTIFIERS	UNIQUELY IDENTIFYING
analytics_storage	appInstanceID	
ad_storage	google_ad_id, cookie*	
ad_user_data	core_platform_services	
personalization_signals	No change to identifiers	

Figure 5.1: Table of consent mode tags.

The first consent type, `analytics_storage`, allows Google to collect the app instance ID. Firebase instance IDs identify each installation of an app [52]. They are unique to a particular app and device. While the instance ID is not a directly persistent or cross-app identifier, it still allows tracking within the scope of an app instance. App developers can delete instance IDs and can stop them from being automatically generated. Firebase documentation provides instructions for developers on how to implement these changes [52]. However, users cannot delete them or prevent them from being generated.

The consent type `ad_storage` enables Google to collect the `google_ad_id` identifier. The Google Advertising ID (GAID) uniquely identifies a mobile device, regardless of whether the user is logged into a Google Account or not [53]. The user can choose to reset or delete the ID at any point via the handset settings. If never reset, it is persistent across apps on the

same device. This enables advertisers to track user activity across different apps. If it is deleted, the value of this identifier becomes a sequence of zeros, which nullifies its use for tracking purposes. Additionally, this tag allows Google to collect the identifier cookie. The value of this identifier is a DSID cookie. Google provides minimal documentation about DSID cookies – what they are or how Google uses them.

Research methods explained in Section 4.1.3 were used to investigate DSID cookies. The function presented in Figure 5.2 shows the point at which the DSID cookie is saved. In this function, two strings named `str` and `str2` are saved to a `SharedPreferences` file. These represent the index and the value of the DSID cookie respectively. These two string values are passed as parameters to the function shown. Tracing the origin of these values reveals that the index of the DSID cookie is a variable called `account.name`. Further investigation using ADB revealed the file on the Client device that stores the DSID cookie. This file is shown in Figure 5.3, which reveals that the DSID cookie is indexed using the user’s Google account email address. This is concerning because it directly links the DSID cookie to an identifiable individual rather than just a device. This means it could be used to construct a detailed profile of the user’s online activity.

```
private final void g(String str, String str2, a aVar) {
    DoritosCookieStorage doritosCookieStorage = this.c;
    if (aVar == a.GAIALESS_DORITOS) {
        doritosCookieStorage.b.edit().putString(str, str2).apply();
    } else {
        doritosCookieStorage.sharedPreferences.edit().putString(str, str2).apply();
    }
    com.google.android.gms.ads.internal.util.client.h.d("Saved DSID.");
}
```

Figure 5.2: DSID cookie recorded in Google Play Services method.

```
1 <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2 <map>
3   <string
      name="markfinalyearproject@gmail
      .com">DSID=ABY2FK4mTuWutuIw9dsM9tRmSmUXPcmn2KA7i4sr9CmbTUii6vmVS4e4cvHZoXQ-s
      Ka-C1VXL8XvcOH6r-FOAd-kh6NRQeNwtP63QdfMxuYgzqbh4_ZRn20FBt0VwcRpZXt8R6vudbZtu
      PgQJFzqWhbuOHXDuDnOosq20UHJ_eKqbuJfXfPw0jjbnhTWbms9K9xKBS34ywLg0YP85FI3dtdPn
      Jlb9j6XTGQsjPYGHSPkceRlQq-kWd0zL5877Dw1gmhpZTtm7h0xb9n81PWgk6unyUQTds7vg;
      expires=Tue, 11-Mar-2025 00:00:00 GMT; path=/; domain=.doubleclick.net;
      Secure; HttpOnly; SameSite=none</string>
4 </map>
```

Figure 5.3: File on the client device where the DSID cookie is saved.

The tag `ad_user_data` controls the identifier `core_platform_services`. When the tag is disabled, the value of the identifier is an empty string. However, when the tag is enabled, the value is set to “ahmopsy”. This value seems to encode consent to Google Services. The same value appears in an article discussing event data using an IOS application [54]. In that

context, it is associated with a variable called `dma_cps`. According to Google's documentation, `dma_cps` is the HTTP parameter used to encode consent to Google services [55].

Finally, the consent mode tag `personalization_signals` was also tested. No additions or changes to identifiers were recorded when it was enabled. This suggests that it does not directly contribute to the collection of unique tracking identifiers in the same way that `ad_storage` and `ad_user_data` do.

5.2 Unspecified consent State

The identifiers logged to Google servers in the "unspecified" consent state are presented in Figure 5.4. When compared to Figure 5.5 – where all consent types are enabled – it becomes evident that the two states result in the same data collection. In the unspecified state, the `consent_state` variable is set to "G1—", indicating that Consent Mode has not been implemented by the app developer. All the same identifiers are logged in each scenario.

This network traffic suggests that when consent mode is not implemented, Google's data collection behaviour defaults to the most permissive settings. Thus, this allows the collection of the user's personal information as if they had fully consented. The lack of privacy in the unspecified state puts a greater responsibility on developers to implement Consent mode correctly on their app. If a developer doesn't implement consent mode, users may be subjected to a state that assumes they have fully consented. This raises questions about how often Consent mode is correctly implemented in real applications, which is addressed in Section 5.4.

The default behaviour in the absence of explicit consent is important to understand because it highlights a gap in Google's documentation that is potentially dangerous. If developers are unaware of the default behaviour when Consent mode is not implemented, they may assume that leaving consent unspecified will limit data collection. In reality, it does not.

```

upload_timestamp_millis: 1737635868540
event_timestamp: 1737635868067
bundle_end_timestamp: 1737635868071
previous_bundle_end_timestamp: 1737635866786
operating_system: "android"
operating_system_version: "13"
device_model: "Pixel 4a"
user_default_language: "en-us"
app_store: "manual_install"
package_name: "com.google.firebase.sample"
app_version: "1.0"
gmp_version: 108021
uploading_gmp_version: 244433
google_ad_id: "39415d30-611b-4a8f-9781-48cce8b388ab"
appInstanceId: "5d880cdc03484e75dd185156ecac34b"
dev_cert_hash: 6835378825754193913
daily_conversions_count: 48
gmp_app_id: "1:719602967806:android:c7d5c503d27425062431af"
previous_bundle_start_timestamp2: 1737635866785
service_upload: true
firebase_instance_id: "d-5D4yxcSmKKC-qdXnenbA"
app_version_int: 1
config_version: 1737558298820251
cookie: "DSID=ACZUy1zGqVnen25DFCxFBZtVdNASB1XeoTTHK3LmUr3hZJ65bXU8HyfnYpBnaGIpJf7Hew
MlpvDIB6SWhU7pyAIHv_uhjdWJlvY-UFAHsOwatIeuKOUlVkg28ZmXTn4yUAWgkezS62rJbSiYqC-XlXL8Qzh
BGBTXDRMwb5Y5gbbM7gotqFtX5Ei8w1LJX61BkM0c6h8Unt-RwsowQCry0KYepXeVX7TAnuRU18sliavA51jVP
F6tMM_5gjjw080bRrxViULE5x9--cGQjPxJrxVI-SGcEkQvhGg"
M: 23181810
M: 23181696
dynamite_version: 130
consent_state: "G1--"
target_os_version: 35
consent_diagnostics: "19911"
is_dma_region: true
core_platform_services: "ahmopsy"
delivery_index: 12

```

Figure 5.4: Identifiers logged to Google with unspecified consent state.

```

upload_timestamp_millis: 1737118867512
event_timestamp: 1737118866475
bundle_end_timestamp: 1737118866476
previous_bundle_end_timestamp: 1737118862182
operating_system: "android"
operating_system_version: "13"
device_model: "Pixel 4a"
user_default_language: "en-us"
app_store: "manual_install"
package_name: "com.google.firebase.quickstart.analytics"
app_version: "1.0"
gmp_version: 108021
uploading_gmp_version: 244433
google_ad_id: "39415d30-611b-4a8f-9781-48cce8b388ab"
appInstanceId: "821d355b367f60282c98c9405424ea41"
dev_cert_hash: 6835378825754193913
daily_conversions_count: 2078
gmp_app_id: "1:719602967806:android:192860c5a2293b2d2431af"
previous_bundle_start_timestamp2: 1737118862182
service_upload: true
firebase_instance_id: "cggk0WpnhRqutbKYBo-un2t"
app_version_int: 1
config_version: 1737046666811008
cookie: "DSID=ACZUy1zKH4rRqUvD_CV_xcxmmNCjp0Wo6h8-6wky4tjZIW9mfEjOpYhKPHCHX8IGKpfNI7S
hUxBJgOKDKHBJ1WxCaHn3ZBCeA4YbckyW4sEbw1By3541RR138VHzrAGtAadP-zdSzVBD-CPaalF6muCrKeXcsP
AIJjz0oVUt4D1bI5ppagyQfGhZu_EVZ1NswO4Tdm6f2HUgnBLD5f3PM1_S143e1PJmMPrx3_OifgUUQR0an2Ae
BBiZoGn1M8L84cUi_WED-Slw86gwZx_vIqcNH2XIYDTg"
M: 23181810
M: 23181696
dynamite_version: 130
consent_state: "G111"
target_os_version: 35
consent_diagnostics: "13333"
is_dma_region: true
core_platform_services: "ahmopsy"
delivery_index: 593

```

Figure 5.5: Identifiers logged to Google when all consent types are enabled.

5.3 Consent Settings

There are three sensitive identifiers discussed in Section 5.1. These are `app_instance_id`, `google_ad_id` and `cookie`. These identifiers enable Google to track users across sessions and devices. The results in Figure 5.6 indicate that user consent settings have no impact on the collection of these sensitive identifiers by Google. Regardless of the user’s Ad Privacy or Google Account settings, the same set of identifiers are logged. This raises concerns about the effectiveness of the settings available to users.

Figure 5.6 also shows the network traffic recorded under different configurations of Consent mode. The results show that enabling certain consent types impacts the presence of certain identifiers, regardless of the user’s preferences. This puts even greater responsibility on the developer to implement Consent mode because otherwise, users will have no control over their data being sent to Google.

Consent Types Enabled	analytics_storage, ad_storage, ad_user_data, ad_personalization_signals	N/A	analytics_storage, ad_storage	ad_user_data, ad_personalization_signals
All User Consent Given	app_instance_id, google_ad_id, cookie	-	app_instance_id, google_ad_id, cookie	-
User Ad Privacy Consent Given	app_instance_id, google_ad_id, cookie	-	app_instance_id, google_ad_id, cookie	
No User Consent Given	app_instance_id, google_ad_id, cookie	-	app_instance_id, google_ad_id, cookie	

Figure 5.6: Identifiers collected under different consent configurations with Google Signals enabled.

Further analysis reveals that Google Signals also plays a role in determining which identifiers are transmitted. When Google Signals is enabled, the `cookie` identifier is additionally logged. The `cookie` identifier contains a DSID cookie. When disabled, as in Figure 5.7, this cookie is absent. This confirms that the collection of the DSID cookie is controlled by Google Signals. As discussed in Section 5.1, the DSID cookie identifies a user’s Google Account. The only action required to collect this cookie is for the developer to enable Google Signals. The fact that the user has no control over whether this sensitive piece of data is collected is concerning. The lack of documentation surrounding Google Signals and DSID cookies means developers who enable Google Signals may not be aware of the sensitive data they are exposing to Google from their users. Google claims the DSID cookie “is used to identify a signed-in user on non-Google sites so that the user’s ads personalization setting is respected accordingly” [8]. However, this claim is misleading. The ad personalization settings available

to Android users through their Google account settings do not influence the transmission of the DSID cookie. As demonstrated in Figure 5.7, even when all user consent settings are disabled, the DSID cookie is still transmitted via the cookie identifier.

Consent Types Enabled	analytics_storage, ad_storage, ad_user_data, ad_personalization _signals	N/A	analytics_storage, ad_storage	ad_user_data, ad_personalization _signals
All User Consent Given	app_instance_id, google_ad_id	-	app_instance_id, google_ad_id	-
User Ad Privacy Consent Given	app_instance_id, google_ad_id	-	app_instance_id, google_ad_id	
No User Consent Given	app_instance_id, google_ad_id	-	app_instance_id, google_ad_id	

Figure 5.7: Identifiers collected under different consent configurations with Google Signals disabled.

5.4 Women’s Health Apps

Out of 11 apps tested, 9 of them used Google Analytics for Firebase. The apps from the sample that did not use Google Analytics were Clatch and Flo, the latter being the most popular women’s health app. The use by 9 out of 11 apps suggests that Google Analytics is a popular choice for app developers, as supported by reports [30].

For the 9 apps that implemented Google Analytics, Figure 5.8 provides an overview of how user consent and data collection are managed. Notably, none of these apps correctly implement Google’s Consent Mode. As a result, the consent status remains unspecified across all tested apps, meaning that Google Analytics does not receive any explicit signal about whether the user has consented to data collection.

4 out of 9 of these apps have Google Signals enabled. In theory, Google Signals should only allow the collection of DSID cookies if the user explicitly consents to ad personalization. However, because these apps fail to implement Consent Mode, Google defaults to the unspecified consent state. This consent state proceeds with data collection as if full consent has been granted. This means that any app enabling Google Signals is effectively permitting the collection of DSID cookies, which can be linked to a user’s Google account and used for cross-device tracking. In this case, the collection of these cookies does not depend on whether the user has actually agreed to the data collection.

For the 4 apps that do not have Google Signals enabled, data collection still occurs.

WOMEN'S HEALTH APP	CONSENT STATUS	GOOGLE SIGNALS	CONSENT BANNER
Period Tracker Period Calendar	Unspecified	No	Yes
MeetYou	Unspecified	Yes	Yes
My Period Calendar & Tracker	Unspecified	Yes	No
Clue Period & Cycle Tracker	Unspecified	No	No
Period Tracker Ovulation Cycle	Unspecified	Yes	Yes
Period & Cycle Tracker - Clover	Unspecified	No	No
Stardust: Period & Pregnancy	Unspecified	No	No
WoCute	Unspecified	Yes	No
Luna	Unspecified	Yes	No

Figure 5.8: Data collection of popular women's health apps.

Although DSID cookies are not collected, these apps still allow Google to collect other identifiers, such as the GAID. This identifier can be used to track a user across different apps on the same device and is persistent unless manually reset by the user. Since consent remains unspecified, users of these apps may unknowingly be subjected to extensive tracking by Google's analytics and advertising systems.

Consent mode requires that the app first collects user consent preference and then communicates it to Google. To collect user consent preferences, Google documentation instructs developers to implement a consent banner. 3 out of 9 apps in the sample implemented a consent banner. The rest of the apps have a system where users have to opt in to privacy terms before gaining access to the app. Out of the apps that implemented a consent banner, further recordings were conducted to demonstrate the effects of not implementing consent mode. The sensitive identifiers collected in these recordings are presented in Figure 5.9.

Figure 5.9 shows that a user's choice to grant or deny consent has no impact on the identifiers collected by Google. Regardless of the user's selection, the same unique identifiers (notably the GAID and Instance ID) are transmitted. In the case of the app Period Tracker

APPS	Consent Granted	Consent Denied
Period Tracker Period Calendar	GAID, Instance ID	GAID, Instance ID
MeetYou	GAID, Instance ID	GAID, Instance ID
Period Tracker Ovulation Cycle	GAID, Instance ID, DSID	GAID, Instance ID, DSID

Figure 5.9: Identifiers collected by Google on apps with a consent banner.

Ovulation Cycle, the DSID is also collected, even when consent is denied. This is because this app enables Google Signals. Therefore, despite the presence of a consent banner, the apps do not properly implement consent mode as intended by Google’s instructions. Instead of restricting data collection when consent is denied, these apps continue to transmit user-identifying information. Users may feel reassured when choosing to deny consent, believing their privacy preferences are being respected. In reality, their preferences are not strictly adhered to by Google Analytics as their sensitive information is collected regardless of their choice.

5.4.1 Discussion

The results of this research reveal a consent management system established by Google Analytics that significantly limits user control while placing substantial responsibility on app developers. The results indicate that users consent is not strictly being adhered to, with their personal data being collected without clear consent. The data collected from various consent modes illustrates that app developers have the authority to determine how user data is managed, yet they may not fully comprehend the privacy ramifications of their decisions. For example, the ability to collect identifiers like the DSID cookie and GAID without explicit user consent highlights a significant concern in users’ control over their data.

The results also show that when consent mode is not properly implemented, Google defaults to the most permissive data collection settings. This means that users may unknowingly be subjected to extensive tracking. The lack of clarity in Google’s documentation regarding the default behaviours of consent mode can lead to developers allowing data collection practices that infringe on user privacy. This situation is particularly alarming given the laws on data privacy. Many countries require explicit user consent for data collection and sharing. For example, Ireland, where the experiments were conducted, is subject to the General Data Protection Regulation (GDPR). GDPR states that consent must be freely given, specific, informed, and unambiguous [56]. While this report does not attempt to make a legal argument, it does highlight a potential discrepancy between Google’s consent management

practices and the laws that it is subject to.

As noted in Google's documentation [57], the onus is placed on developers to understand and comply with varying laws regarding user consent. This expectation may not be entirely reasonable, especially considering the complexity of privacy regulations that differ across regions and are subject to change. Developers may struggle to comply with these requirements. Consequently, this could lead to unintentional violations of user privacy rights.

Furthermore, this research highlights that even when consent banners are implemented, the actual impact on data collection is minimal. Users may feel reassured by the presence of a consent banner, believing their preferences are respected. However, the reality is that their choices often do not influence the identifiers collected by Google Analytics. The difference between user perception and actual data practices can damage trust in both app developers and the platforms they utilize. Research demonstrates that when users are made aware of the data practices of the apps they use, it can lead to increased frustration and distrust [58]. This ultimately undermines user confidence in the digital services they are using.

There are ethical considerations involved in this research. Essentially, the findings of this report represent vulnerabilities in Google's system. When exposing vulnerabilities, researchers are subject to responsible disclosure. This means there is a responsibility to inform Google about these issues. Upon informing Google, a month should proceed before the information is revealed to the public. These are live systems that effect many people around the world so it is crucial to reveal this information as soon as possible. However, it is also important to give Google enough time to respond.

6 Conclusion

This report investigated the data collection practices of Google Analytics. The investigation uncovered significant concerns for personal data privacy. By analysing network traffic, the research demonstrated that Google Analytics often collects user data beyond what is consented to. The findings of this report emphasize the need for greater transparency in Google's documentation. Given the increasing reliance on mobile devices, respecting user consent should be a priority for both developers and regulatory bodies. Ultimately, this research contributes to the broader discussion on digital privacy and the role of Google's advertising ecosystem.

6.0.1 Future Work

There is plenty of opportunity to expand on the research of this project. The most immediate opportunity seems to be to investigate how Google Analytics collects data on devices with other operating systems. For example, if data collection practices on iOS were found to be more privacy-conscious, it could strengthen the case for platform-level accountability.

This report does not attempt to make a legal argument. However, there is an opportunity for future work to formulate one. Google's advertising ecosystem is subject to increased scrutiny. As discussed, they are currently undergoing an anti-trust lawsuit in the US related to their ad-tech business [22]. Future work could add to these criticisms by exploring the legal implications of the data collection practices uncovered in this study. Particularly, exploring how these practices relate to user consent and transparency frameworks such as GDPR in the EU.

Lastly, while Google Firebase is very popular, it is not the only SDK available to app developers. Future work could investigate other SDKs that offer analytics services. If these services manage user consent better, it may encourage app developers to consider other SDKs when building their applications.

Bibliography

- [1] T. Economist, "The world's most valuable resource is no longer oil, but data," 2017. [Online]. Available: <https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data>
- [2] J. Kagan, "Data mining: What it is, how it works, and examples. investopedia," 2023. [Online]. Available: <https://www.investopedia.com/terms/d/datamining.asp>
- [3] Fibr., "10 tools to understand and analyze user behavior. fibr," n.d. [Online]. Available: <https://fibr.ai/conversion-rate-optimization/user-behavior-tools>
- [4] T. Magazine, "Top 10 companies in the world of big data," 2023. [Online]. Available: <https://technologymagazine.com/top10/top-10-companies-in-the-world-of-big-data>
- [5] *Google Analytics*, Google Firebase, 2025. [Online]. Available: <https://firebase.google.com/docs/analytics>
- [6] D. Ireland, "Consumer trends: Smartphone usage in ireland," 2023. [Online]. Available: <https://www.deloitte.com/ie/en/about/press-room/consumer-trends-smartphone-usage.html>
- [7] C. F. Hayes, D. and N. Le-Khac, "An effective approach to mobile device management: Security and privacy issues associated with mobile applications. digital business, 1(1), p.100001." vol. 1, 2020.
- [8] *How Google uses cookies*, Google Privacy Terms, 2025. [Online]. Available: <https://policies.google.com/technologies/cookies?hl=en-US>
- [9] D. P. Commission, "Rights of individuals under the general data protection regulation," 2025. [Online]. Available: <https://www.dataprotection.ie/en/individuals/rights-individuals-under-general-data-protection-regulation>
- [10] W. S. Journal, "European privacy regulators step up scrutiny of business data practices," 2023. [Online]. Available: <https://www.wsj.com/articles/european-privacy-regulators-step-up-scrutiny-of-business-data-practices-11674035444>

- [11] D. Leith, "Mobile handset privacy: Measuring the data ios and android send to apple and google," 2021.
- [12] —, "What data do the google dialer and messages apps on android send to google?" 2022.
- [13] W. Z.-R. Michael Grace, "Unsafe exposure analysis of mobile in-app advertisements," 2012.
- [14] M. e. a. Fan, "Giving without notifying: Assessing compliance of data transmission in android apps," 2024.
- [15] N. e. a. Alfawzan, "Privacy, data sharing, and data security policies of women's mhealth apps: Scoping review and content analysis," 2022.
- [16] J. e. a. Cao, "'i deleted it afer the overturn of roe v. wade': Understanding women's privacy concerns toward period-tracking apps," 2024.
- [17] AppBrain, "Free and paid android applications," 2025. [Online]. Available: <https://www.appbrain.com/stats/free-and-paid-android-applications>
- [18] A. Group, "How free apps make money: Become rich with free mobile apps," 2024. [Online]. Available: <https://attractgroup.com/blog/how-free-apps-make-money-become-rich-with-free-mobile-apps/>
- [19] B. of Apps, "Instagram revenue and usage statistics (2025)," 2025. [Online]. Available: <https://www.businessofapps.com/data/instagram-statistics/>
- [20] S. Levin, "Android phones send data to google even when location services are off," 2017. [Online]. Available: <https://www.theguardian.com/technology/2017/nov/28/android-apps-third-party-tracker-google-privacy-security-yale-university>
- [21] T. Franck, "How google makes money from ads, in one chart," 2021. [Online]. Available: <https://www.cnbc.com/2021/05/18/how-does-google-make-money-advertising-business-breakdown-.html>
- [22] B. Fung, "Doj antitrust case against google's ad tech heads to trial," 2024. [Online]. Available: <https://edition.cnn.com/2024/02/05/tech/doj-antitrust-case-google-ad-trial-september/index.html>
- [23] Lunio, "How google is taking more control of your ad spend (and what you can do about it)," 2022. [Online]. Available: <https://www.lunio.ai/blog/google-take-control-ad-spend>
- [24] *Concepts: Modes. mitmproxy documentation*, Mitmproxy, 2025. [Online]. Available: <https://docs.mitmproxy.org/stable/concepts-modes/>

- [25] *Overview*, Protocol Buffers, 2025. [Online]. Available: <https://protobuf.dev/overview/>
- [26] G. P. Services, "Guides: Overview of google play services," 2025. [Online]. Available: <https://developers.google.com/android/guides/overview>
- [27] M. Taul, "What is firebase? the complete story (abridged)," 2020. [Online]. Available: <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>
- [28] *How Google Analytics works*, Google Analytics Help, 2025. [Online]. Available: <https://support.google.com/analytics/answer/12159447?sjid=11446226387472104923-EU>
- [29] Google, "Firebase analytics gets new features and a familiar new name," 2019. [Online]. Available: <https://blog.google/products/marketingplatform/analytics/firebase-analytics-gets-new-features/>
- [30] E. Privacy, "Google analytics for firebase. exodus privacy reports," n.d. [Online]. Available: <https://reports.exodus-privacy.eu.org/en/trackers/49/>
- [31] P. PRO, "Is session-based analytics dead? no. here's why," 2023. [Online]. Available: <https://piwik.pro/blog/is-session-based-analytics-dead/>
- [32] *[GA4] Realtime and DebugView reports*, Google Support, 2025. [Online]. Available: <https://support.google.com/analytics/answer/9322688#zippy=%2Crealtime-report%2Cdebugview-report>
- [33] *About consent mode*, Google Support, 2025. [Online]. Available: <https://support.google.com/google-ads/answer/10000067?hl=en>
- [34] *Set up consent mode for apps*, Google Security and Privacy Hub, 2024. [Online]. Available: <https://developers.google.com/tag-platform/security/guides/app-consent?consentmode=advanced&platform=android>
- [35] *Samples*, Google Firebase Documentation, 2025. [Online]. Available: <https://firebase.google.com/docs/samples>
- [36] MeasureSchool, "Google signals explained (in-depth guide)," 2025. [Online]. Available: <https://measureschool.com/google-signals-explained/>
- [37] *Activate Google signals for Google Analytics properties*, Google Analytics Help, 2025. [Online]. Available: <https://support.google.com/analytics/answer/9445345?hl=en>
- [38] *Add Firebase to your Android project*, Google Firebase Documentation, 2025. [Online]. Available: <https://firebase.google.com/docs/android/setup?hl=en>
- [39] S. Stern, "Google analytics for firebase quickstart," 2017. [Online]. Available: <https://github.com/firebase/quickstart-android/blob/master/analytics/README.md>

- [40] *Enable debug mode*, Google Firebase Documentation, 2025. [Online]. Available: https://firebase.google.com/docs/analytics/debugview#enable_debug_mode
- [41] A. D. Blog, "Changes to trusted certificate authorities in android nougat," 2016. [Online]. Available: <https://android-developers.googleblog.com/2016/07/changes-to-trusted-certificate.html>
- [42] *Magisk Manager*, Magisk, 2025. [Online]. Available: <https://magiskmanager.com/>
- [43] N. Security, "Magisktrustusercerts," 2022. [Online]. Available: <https://github.com/NVISOsecurity/MagiskTrustUserCerts>
- [44] *Dependencies of Firebase Android SDKs on Google Play services*, Google Firebase Documentation, 2025. [Online]. Available: <https://firebase.google.com/docs/android/android-play-services>
- [45] Skylot, "Jadx – dex to java decompiler," 2025. [Online]. Available: <https://github.com/skylot/jadx>
- [46] *Android Debug Bridge (adb)*, Android Developers, 2025. [Online]. Available: <https://developer.android.com/tools/adb>
- [47] D. Leith, "Android protobuf decoding scripts," 2025. [Online]. Available: <https://github.com/doug-leith/android-protobuf-decoding>
- [48] OwnTag, "What is the gsc parameter?" 2023. [Online]. Available: <https://www.owntag.eu/blog/gsc-parameter/>
- [49] *Set up consent mode for apps*, Google, 2024. [Online]. Available: <https://developers.google.com/tag-platform/security/guides/app-consent?consentmode=advanced&platform=android>
- [50] *Manage your ad privacy settings on Android*, Google Android Help, 2025. [Online]. Available: <https://support.google.com/android/answer/13720755>
- [51] AppMagic, "Top apps – period tracker, worldwide, 2024," 2025. [Online]. Available: <https://appmagic.rocks/top-charts/apps?date=2024-01-01&tag=243520&aggregation=year>
- [52] *Manage Instance ID data*, Google Firebase Support, 2025. [Online]. Available: <https://firebase.google.com/support/privacy/manage-iids>
- [53] *Advertising ID*, Google Play Console Help, 2025. [Online]. Available: <https://support.google.com/googleplay/android-developer/answer/6048248?hl=en>

- [54] S. Ahava, "Send app data to server-side google tag manager, team simmer," 2024. [Online]. Available: <https://www.teamsimmer.com/2024/10/22/send-app-data-to-server-side-google-tag-manager/>
- [55] *Consent Mode HTTP Parameters*, Google Security and Privacy hub, 2024. [Online]. Available: https://developers.google.com/tag-platform/security/concepts/consent-mode#consent_mode_http_parameters
- [56] GDPR.eu, "Gdpr consent," n.d. [Online]. Available: <https://gdpr-info.eu/issues/consent/>
- [57] *Introduction to user consent management*, Google Analytics Help, 2025. [Online]. Available: <https://support.google.com/analytics/answer/12329599?hl=en>
- [58] A. e. a. Bowyer, "Human-gdpr interaction: practical experiences of accessing personal data," 2022.