Differentially-Private Software Analytics for Mobile Apps: Opportunities and Challenges

Hailong Zhang
Ohio State University
Columbus, OH, USA
zhang.4858@osu.edu

Sufian Latif
Ohio State University
Columbus, OH, USA
latif.28@osu.edu

Raef Bassily
Ohio State University
Columbus, OH, USA
bassily.1@osu.edu

Atanas Rountev
Ohio State University
Columbus, OH, USA
rountev@cse.ohio-state.edu

ABSTRACT

Software analytics libraries are widely used in mobile applications, which raises many questions about trade-offs between privacy, utility, and practicality. A promising approach to address these questions is *differential privacy*. This algorithmic framework has emerged in the last decade as the foundation for numerous algorithms with strong privacy guarantees, and has recently been adopted by several projects in industry and government. This paper discusses the benefits and challenges of employing differential privacy in software analytics used in mobile apps. We aim to outline an initial research agenda that serves as the starting point for further discussions in the software engineering research community.

CCS CONCEPTS

• **Security and privacy** → *Domain-specific security and privacy architectures*; • **Software and its engineering** → *Dynamic analysis*;

KEYWORDS

software analytics, differential privacy, mobile apps

ACM Reference Format:
Hailong Zhang, Sufian Latif, Raef Bassily, and Atanas Rountev. 2018. Differentially-Private Software Analytics for Mobile Apps: Opportunities and Challenges. In *Proceedings of the 4th ACM SIGSOFT International Workshop on Software Analytics (SWAN '18), November 5, 2018, Lake Buena Vista, FL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3278142.3278148

1 INTRODUCTION

Software analytics libraries have strong presence in the mobile app markets and are used by app developers for behavioral analytics, app improvements, targeted advertising, and location tracking [12, 27]. With such data gathering, user privacy becomes a concern. This concern is amplified in the current environment, in which both software users and legislative bodies are becoming increasingly proactive in demanding privacy protections. In this context, it is essential to understand and enforce meaningful trade-offs between the benefits of data gathering and the privacy of software users.

Such trade-offs can be studied through the lens of privacy-preserving data analysis. In particular, we consider *differential privacy* [8], an approach that quantifies these trade-offs and provides a framework for disciplined algorithm design. Due to its attractive properties, differential privacy has reigned as the gold standard of statistical data privacy. This approach has recently found several successful adoptions in industry—for example, Google's use of RAPPOR in the Chrome browser [11] and Apple's use of differential privacy in iOS-10 [1, 26].

The goal of this short position paper is to highlight the benefits and challenges of applying differential privacy in the context of software analytics used in mobile apps. We aim to outline an initial research agenda that serves as the starting point for further discussions in the software engineering research community.

The rest of the paper is organized as follows. Section 2 provides background on software analytics for mobile apps, with focus on analytics for Android apps. Section 3 discusses differential privacy, and in particular *local* differential privacy (LDP) which is especially well suited for analyzing the behavior of mobile apps. Section 4 describes several dimensions in designing general and practical LDP mechanisms for app analytics, and defines a series of research questions to be addressed by future work. Section 5 summarizes our conclusions and suggestions.

2 ANALYTICS FOR ANDROID APPS

There are many providers of analytics libraries for mobile apps. Prominent examples include Google Analytics (GA) [16], its successor Firebase [15], Facebook Analytics [13], and Yahoo's Flurry [23]. Recent examination [12] of a collection of Android apps identified the following percentages of apps that use these analytics libraries:

Google Firebase	43%
Google Analytics	38%
Facebook Analytics	24%
Flurry (by Yahoo)	18%

Even though the policies of these analytics infrastructures require that developers do not gather user-identifiable information, there is

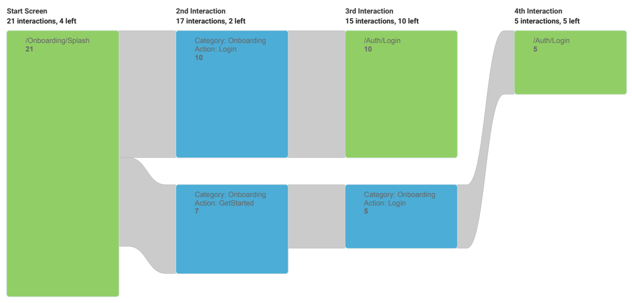

Figure 1: GA behavior flow of screen views and events.

no enforcement of such guidelines; as a result, users are vulnerable to abuse and data leaks by app developers and library providers.

Analytics libraries allow a wealth of data to be collected and analyzed. As one basic capability, an app developer can define interesting *events* to be recorded when a specific action occurs: e.g., visiting a screen, making a purchase, or clicking a button. An event could also have data parameters such as the price of a purchase. The run-time occurrence of such an event is reported by the app via calls to analytics library APIs. These calls send the data to the analytics server (e.g., Google's server). An app developer can query the server to obtain, for example, a histogram of event frequencies.

Example. Google Analytics (GA) can be used to gather detailed information about user behavior. For example, every time the user views some screen in the app's GUI, this event can be recorded. As another example, data from e-commerce transactions could be collected. The app uses library `com.google.android.gms.analytics` to record the data. For example, the execution of `tracker.send(new HitBuilders.EventBuilder().setCategory("Action").setAction("Share").build())` will send to Google's servers an event that records a user action related to content sharing. GA gathers data from multiple users—that is, from different instances of the same app running on thousands of devices—and presents various summary statistics to the developer of this app. (It is also possible to drill down this information to the level of a single app user.) From this data, GA generates various reports for event frequencies. For example, Figure 1 illustrates observed frequencies for different app screens and the events that trigger flow between screens, accumulated over a collection of app users.

Google provides other services to facilitate the analysis and presentation of GA and Firebase data. Data Studio enables developers to generate more readable and complex reports. Another related service is BigQuery, which is used to query large datasets. All raw GA data related to events and users could be exported to BigQuery. Developers can then use SQL-like syntax to query the data.

Analytics frameworks for mobile apps provide weak privacy protections. For example, a developer-defined configuration parameter in GA can eliminate the last component of the IP address. As another example, the best-effort privacy in GA and Firebase is to prohibit developers from including "personally identifiable information such as names, social security numbers, or email addresses, even in hashed form".[1] While such restrictions are useful, there is no indication that they are being systematically checked. Even more troubling, it is well known that anonymization provides no protection against more sophisticated privacy attacks that cross-reference data from several sources. In this day and age, both in society and (increasingly) in law, privacy is becoming a major concern. Privacy-preserving software analytics for mobile apps is an important and challenging instance of this problem, which can be addressed in a foundational manner using differential privacy.

[1] https://support.google.com/firebase/answer/6317519 and https://support.google.com/analytics/answer/2795983

3 DIFFERENTIAL PRIVACY

In the last decade, differential privacy [8, 10, 21] has emerged as a powerful protection mechanism against privacy attacks. The scenario under consideration is the following. After a planned public release of some data, an adversary attempts to learn private information from that data—for example, to achieve person re-identification, linking of records from different sources, or differencing attacks. The data is released to government, businesses, or researchers for legitimate reasons. The adversary can also access the data and, in addition, may know what kinds of data sanitization and aggregation were performed before the release. Here "adversary" should be considered in the broadest possible sense: e.g., data released to a company may later be obtained and misused by another company (e.g., as part of a corporate takeover), or may be subpoenaed by law enforcement. Anonymizing or removing personally-identifiable information does not provide strong protections, as demonstrated by many researchers [6, 18, 19, 25]. For example, Narayanan and Shmatikov [18] analyzed the anonymized data in the Netflix Prize dataset and identified users by cross-linking with IMDB data.

Differential privacy (DP) provides theoretical guarantees against such attacks. Intuitively, when an adversary observes the results of a DP analysis, she will reach essentially the same conclusion about an individual's private information, regardless of whether that individual's data is part of the analysis input [21]. Many DP solutions have been proposed for various analysis problems. As one example, the output of a non-DP analysis could be perturbed using random noise with Laplacian or Gaussian distribution.

Descriptions of this vibrant research area are available elsewhere [10, 21]. DP solutions have been deployed by companies such as Google, Apple, and Uber. As another example, last year it was announced that the U.S. Census Bureau will use differential privacy in the 2020 census. Given the rapid emergence of large-scale data analytics and machine learning, and their detrimental effects on privacy, the importance and urgency of privacy solutions (including differential privacy) will continue to increase.

Local differential privacy (LDP) [17] is a variant of DP in which each user performs local data perturbation. The modified data is sent to an untrusted data curator, where data analysis is performed and the analysis results are released to clients. There exist several practical realizations of LDP algorithms. Google's RAPPOR identifies URLs in the Chrome browser [11, 14]. Apple gathers analytics data for emoji and quick type suggestions, search hints, and health-related usage [1, 26]. Samsung's Harmony collects data from smart devices [20]. Microsoft uses LDP to collect telemetry data [5].

The LDP model is particularly well suited for app analytics for mobile devices, as it provides privacy guarantees to the app user regardless of the actions of the app developer or the analytics infrastructure company (e.g., Google). For the widely-used analytics frameworks discussed in Section 2, there is no prior work on achieving LDP analytics. The next section outlines several challenges for achieving the rigorous privacy/utility trade-offs provided by LDP.

4 CHALLENGES FOR DIFFERENTIALLY PRIVATE MOBILE APP ANALYTICS

We illustrate LDP using a fundamental problem in data analytics: *event frequencies*. This problem is closely related to many analytics problems, including counting queries, histograms, heavy hitters, distribution estimation, regression, clustering, and classification.

4.1 Frequency Estimates

Consider n app users and let each user is identified by an id $i \in \{1, \dots, n\}$. Each user has a data item v_i. The data items are from some data dictionary $\mathcal{D}$ that is pre-defined by the app developer based on her analytics needs. For any $v \in \mathcal{D}$, its frequency $f(v)$ is the number of users that hold this v. The app developer's goal is to obtain all $f(v)$—or, at least, accurate estimates of these frequencies. The analytics frameworks for mobile apps described earlier gather such frequencies and present them to app developers.

An LDP solution to this problem will apply a local randomizer $R : \mathcal{D} \rightarrow \mathcal{Z}$ to each user's item v_i. The resulting $z_i = R(x_i)$ is sent to the server. The server uses all z_i to determine estimates $\hat{f}(v)$ for all actual frequencies $f(v)$. There are various ways to define R. For example, the popular RAPPOR approach [11] represents the data of user i as a bit vector of length $|\mathcal{D}|$, where only bit v_i is set to 1. Then each bit could be randomly inverted using a biased coin. The server processes the resulting bit vectors and accounts for the randomization when computing the estimates $\hat{f}(v)$.

R is parameterized by a parameter ϵ. Two key properties of the ϵ-LDP protocol based on R are *privacy* and *accuracy*. The privacy is due to the design of R: for any $v, v' \in \mathcal{D}$ and $z \in \mathcal{Z}$, we have

$$Pr[R(v) = z] \leq e^{\epsilon} Pr[R(v') = z]$$

The randomization achieves privacy trough plausible deniability: if a user has item v and as a result $z = R(v)$ is reported to the server, the observation of that z (by the server or by a third party) does not reveal "too much" information about which item was held by that user, since the probability that the item was some other v' is close to the probability that the item was the actual v. The accuracy is measured based on the ℓ_∞ norm of the difference between the vector of actual frequencies and the vector of estimates—that is, the largest value of $|\hat{f}(v) - f(v)|$ over all v. Larger values of parameter ϵ result in less privacy but higher accuracy.

4.2 Problem Dimensions

The simplified problem defined above provides a useful baseline for considering the theoretical properties of LDP data analytics. However, practical application of these ideas to real-world mobile apps demands that various more complex problem dimensions be considered. Some of these issues have been addressed by existing work in differential privacy, but most have not. This presents an exciting opportunity for software engineering researchers to define and pursue a new research agenda for LDP software analytics.

4.2.1 *Simple vs Rich Data/Analyses.*

The problem from Section 4.1 considers a single data item per user and a basic frequency analysis of these items. Typically, the data collected from a user's execution of an app is much more complex. In general, one could model this data as a temporal sequence of data tuples drawn from some database schema. For illustration, if we consider one specific code example from the documentation of Google Analytics, we could model the data sent to the server as a tuple `("P12345","Warhol T-Shirt","Apparel/T-Shirts","Google", "Black")` from a conceptual database table with columns "product id", "name", "category", "brand", and "color". Clearly, the definitions of privacy loss and analysis accuracy may have to consider the structure of this information. The type of analysis will also affect the considerations of privacy and accuracy. Rather than simple counts of items, much more powerful analyses could be performed—for example, how many records are returned from some complex database query. The DP literature already considers a range of such analyses (e.g., heavy hitters [2, 4], distribution estimation [7], clustering [22], learning [17], and convex optimization [24]) but the use of these theoretical approaches to achieve practical analyses of real app data remains to be seen.

4.2.2 *One-Shot vs Continuous Data Collection.*

Given the events from multiple instances of an app running on many users' devices, the simplified problem defined in Section 4.1 considers "one-shot" data collection. Each user runs the app for a while; during this period, information is recorded locally. At the end of the data collection period, the data is sent to the server. The server aggregates the information from all n users and reports the results to the app developer. This completes the entire data collection process.

In reality, the observation is continuous and data is sent to the server many times. More precisely, consider an (infinite) sequence of moments in time $t_1, t_2, \dots$ Here t_1 can be thought of as the time the app is published in the app market. Let T_j be the time period from t_i to t_{i+1}, U_j be the set of users that actively use the app during T_j, and $\mathcal{U}$ be the union of all U_j. Let $n_j = |U_j|$ and $n = |\mathcal{U}|$. As before, each $u \in \mathcal{U}$ is identified by an id $i \in \{1, \dots, n\}$.

For the frequency estimation problem discussed earlier, assume a user $i \in U_j$ generates a finite sequence of values $v_{i,1}, v_{i,2}, \dots$ during time period T_j. A user $i \notin U_j$ does not generate any events during T_j. For any $v \in \mathcal{D}$, its frequency during T_j is $f_j(v) = |\{(i,k) \in \{1, \dots, n\} \times \mathbb{N} : v_{i,k} = v\}|$. The histogram defined by $f_j(v)$ for all $v \in \mathcal{D}$ is computed by the server and made accessible to the app developer at the end of each period T_j. Note that from one time period to the next, some new users may be added (due to new app installs) and some existing users may be dropped (due to app uninstalls). Further, a user may be completely inactive during one time period, but active in previous/following time periods. It is necessary to develop research solutions that consider these generalizations of LDP analytics for mobile apps.

The continuous collection of data also raises issues of privacy loss (as exemplified by [9]). As a fundamental theoretical property, each subsequent observation of new public data derived from private data leads to further reduction of privacy guarantees. An important question—both for an app developer and for an analytics infrastructure provider such as Google or Facebook—is how to "forget" previously-observed data. Although this question has been considered for time-series data [3], broader investigations should be part of the research agenda for developing LDP app analytics.

4.2.3 Black-Box vs White-Box Analysis. Several deployment scenarios can be considered for LDP analytics of mobile apps.

Scenario 1: Black-box analytics infrastructure. The analytics frameworks described in Section 2 are not designed with DP capabilities. Suppose an app developer wants to use some existing infrastructure and APIs to collect information from mobile apps in a LDP manner. In this scenario, the app must use calls to the standard analytics APIs but needs to introduces it own implementation layer on top of these APIs (e.g., in order to add randomization). The server is completely unaware of the fact that there is any differential privacy aspect to the data collection and analysis. On the user's device, there are no changes to the API implementations—i.e., there are no changes to the analytics library that implements the APIs and connects to the server. After the server reports its results, they have to be post-processed by the app developer to obtain the actual estimates of the desired analytics data. Such a "black-box" solution is easy to deploy today. However, it requires substantial new algorithmic developments, since all existing work on differential privacy assumes that both the user's local environment and the server's operation are under the control of the protocol designer.

Scenario 2: Local changes, black-box server. Another scenario is when the app implementation layer is combined with a modified version of the analytics libraries running locally on the user's device. The analytics server is still unmodified and LDP-unaware. The modified libraries, created by a trusted party (e.g., privacy researchers), can live side-by-side with the original analytics libraries and can be made available to LDP-aware apps. Clearly, this solution is more intrusive and harder to deploy than the black-box one described earlier. It is interesting to consider what additional capabilities can be achieved in this case, compared to Scenario 1.

Scenario 3: White-box analytics infrastructure. A fully general deployment scenario is where both the analytics libraries on mobile devices and the analytics server are modified. In this case there is great flexibility in achieving and fine-tuning LDP. This scenario requires substantial infrastructure changes and does not allow for an immediate deployment of a LDP solution. Further, such changes require the buy-in of infrastructure companies such as Google and Facebook. However, this scenario is still worth considering in exploring the trade-offs between theoretically-superior solutions and real-world deployment constraints.

4.2.4 Practical Considerations. The performance of DP analyses depends on various parameters: for example, the privacy loss parameter ϵ, the size of the data dictionary $\mathcal{D}$, the number of users n, the number of hash functions used to map the data into compressed representations [2, 11], etc. Accuracy, privacy, and time/memory/communication cost all depend on such parameters. Demonstrating useful trade-offs for these factors under realistic usage scenarios (e.g., using data streams obtained from app executions on real devices) is an important open problem. Creating benchmarks and prototype analysis implementations, and then sharing them with the research community, are also of great importance.

5 SUMMARY

In an environment where data analytics and machine learning are becoming increasingly prevalent, privacy should be a major concern for the designers and users of analytics solutions. Many apps for mobile devices already collect and analyze data with the help of analytics frameworks, but with little or no privacy considerations. Differentially-private mobile app analytics is an attractive but unexplored solution that combines rigorous theoretical reasoning, algorithm design for a wide range of analytics problems, and challenging issues of practical usability and deployment. The software engineering research community can find many important and exciting problems in this emerging research area.

REFERENCES

[1] Apple. 2017. Learning with privacy at scale. https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html.

[2] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. 2017. Practical locally private heavy hitters. In *NIPS*. 2285–2293.

[3] Jean Bolot, Nadia Fawaz, S Muthukrishnan, Aleksandar Nikolov, and Nina Taft. 2013. Private decayed predicate sums on streams. In *ICDT*. 284–295.

[4] Mark Bun, Jelani Nelson, and Uri Stemmer. 2018. Heavy hitters and the structure of local privacy. In *PODS*. 435–447.

[5] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. In *NIPS*. 3571–3580.

[6] Irit Dinur and Kobbi Nissim. 2003. Revealing information while preserving privacy. In *PODS*. 202–210.

[7] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In *FOCS*. 429–438.

[8] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*. 265–284.

[9] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. 2010. Differential privacy under continual observation. In *STOC*. 715–724.

[10] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[11] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*. 1054–1067.

[12] Exodus Privacy. 2018. Most frequent app trackers for Android. https://reports.exodus-privacy.eu.org/reports/stats/.

[13] Facebook. 2018. Facebook Analytics. https://analytics.facebook.com.

[14] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *PoPETs* 2016, 3 (2016), 41–61.

[15] Google. 2018. Firebase. https://firebase.google.com.

[16] Google. 2018. Google Analytics. https://analytics.google.com.

[17] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[18] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *S&P*. 111–125.

[19] Arvind Narayanan and Vitaly Shmatikov. 2009. De-anonymizing social networks. In *S&P*. 173–187.

[20] Thông T Nguyên, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. 2016. Collecting and analyzing data from smart device users with local differential privacy. *arXiv:1606.05053* (2016).

[21] Kobbi Nissim, Thomas Steinke, Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, David O'Brien, and Salil Vadhan. 2018. Differential privacy: A primer for a non-technical audience (preliminary version). https://privacytools.seas.harvard.edu/publications. *Vanderbilt Journal of Entertainment and Technology Law* (2018).

[22] Kobbi Nissim and Uri Stemmer. 2017. Clustering algorithms for the centralized and local models. *arXiv:1707.04766* (2017).

[23] Oath. 2018. Flurry. http://flurry.com.

[24] Adam Smith, Abhradeep Thakurta, and Jalaj Upadhyay. 2017. Is interaction necessary for distributed private learning?. In *S&P*. 58–77.

[25] Latanya Sweeney. 1997. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics* 25, 2-3 (1997), 98–110.

[26] Abhradeep Guha Thakurta, Andrew H. Vyrros, Umesh S. Vaishampayan, Gaurav Kapoor, Julien Freudiger, Vivek Rangarajan Sridhar, and Doug Davidson. 2017. Learning new words. In *Granted US Patents 9594741 and 9645998*.

[27] Yale Privacy Lab. 2017. App trackers for Android. https://privacylab.yale.edu/trackers.html.