

# Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12

Jun Tang  
University of Southern California  
juntang@usc.edu

Aleksandra Korolova  
University of Southern California  
korolova@usc.edu

Xiaolong Bai  
Tsinghua University  
bxl12@mails.tsinghua.edu.cn

Xueqiang Wang  
Indiana University  
xw48@indiana.edu

Xiaofeng Wang  
Indiana University  
xw7@indiana.edu

## ABSTRACT

In June 2016, Apple made a bold announcement that it will deploy local differential privacy for some of their user data collection in order to ensure privacy of user data, even from Apple [21, 23]. The details of Apple's approach remained sparse. Although several patents [17–19] have since appeared hinting at the algorithms that may be used to achieve differential privacy, they did not include a precise explanation of the approach taken to privacy parameter choice. Such choice and the overall approach to privacy budget use and management are key questions for understanding the privacy protections provided by any deployment of differential privacy.

In this work, through a combination of experiments, static and dynamic code analysis of macOS Sierra (Version 10.12) implementation, we shed light on the choices Apple made for privacy budget management. We discover and describe Apple's set-up for differentially private data processing, including the overall data pipeline, the parameters used for differentially private perturbation of each piece of data, and the frequency with which such data is sent to Apple's servers.

We find that although Apple's deployment ensures that the (differential) privacy loss per each datum submitted to its servers is 1 or 2, the overall privacy loss permitted by the system is significantly higher, as high as 16 per day for the four initially announced applications of Emojis, New words, Deeplinks and Lookup Hints [21]. Furthermore, Apple renews the privacy budget available every day, which leads to a possible privacy loss of 16 times the number of days since user opt-in to differentially private data collection for those four applications.

We applaud Apple's deployment of differential privacy for its bold demonstration of feasibility of innovation while guaranteeing rigorous privacy. However, we argue that in order to claim the full benefits of differentially private data collection, Apple must give full transparency of its implementation and privacy loss choices, enable user choice in areas related to privacy loss, and set meaningful defaults on the daily and device lifetime privacy loss permitted.

## ACM Reference Format:

Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. 2017. Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12.

## 1 INTRODUCTION

Differential privacy [7] has been widely recognized as the leading statistical data privacy definition by the academic community [6, 11]. Thus, as one of the first large-scale commercial deployments of differential privacy (preceded only by Google's RAPPOR [10]), Apple's deployment is of significant interest to privacy theoreticians and practitioners alike. Furthermore, since Apple may be perceived as competing on privacy with other consumer companies, understanding the actual privacy protections afforded by the deployment of differential privacy in its desktop and mobile OSes may be of interest to consumers and consumer advocate groups [16].

However, Apple's publicly-facing communications about its deployment of differential privacy have been extremely limited: neither its developer documents [1, 2, 21, 22, 24] nor interstitials prompting the users to opt-in to differentially private data collection (Figures 8 and 9) provide details of the technology, except to say what data types it may be applied to. Two aspects of the deployment are crucial to understanding its privacy merits: the algorithms or processes used to ensure differential privacy of the data being sent and the privacy parameters being used by those algorithms. Although one can speculate about the algorithms deployed based on the recent patents [17–19], the question of parameters used to govern permitted privacy loss remains open and is our primary focus.

Both EFF and academics have called for Apple to detail its privacy budget use [3, 8, 16, 20], to no avail<sup>1</sup>. As far as we are aware, we are the first to systematically study privacy budget use in Apple's deployment of differential privacy.

### 1.1 The (Differential) Privacy Budget

One of the core distinctions of differential privacy (DP) from colloquial notions of privacy is that the definition provides a way to quantify the privacy risk incurred whenever a differentially private algorithm is deployed. Typically called *privacy budget* or *privacy loss* and denoted by  $\epsilon$ , it quantitatively measures by how much the risk to an individual privacy may increase due to that individual's data inclusion in the inputs to the algorithm. The higher the value of  $\epsilon$ , the less privacy protection is provided by the algorithm; in particular, the increase in privacy risks is proportional to  $\exp(\epsilon)$ . Although the choice of  $\epsilon$  is typically treated as a social choice by

Preprint, September 10, 2017.  
2017. .

<sup>1</sup>Apple's only public comments on the privacy budget are "Restrict the number of submissions made during a period. No identifiers. Periodically delete donations from server" [21].

the theoretical computer scientists [6], it is of crucial importance in practical deployments, as the meaning of a privacy risk of  $\exp(1)$  vs  $\exp(50)$  is radically different.

In practice, an individual’s data contribution is rarely limited to one datum. Whenever multiple data are submitted with differential privacy, the overall differential privacy loss incurred by that individual is viewed as bounded by the sum of the privacy losses of each of the submissions, due to what is known as composition theorems [9, 15]. Hence, understanding the privacy implications of a deployed system such as Apple’s, requires not only understanding the privacy loss incurred per datum submitted, but also how many datums may be submitted per time period or over a lifetime of a user’s device. In fact, the need to understand the total privacy loss of differential privacy deployments has prompted Dwork and Mulligan to propose an “Epsilon Registry” [8].

## 1.2 Our Findings

We find that although the privacy loss per datum is strictly limited to privacy budgets typically used in the literature, the daily privacy loss permitted by the implementation exceeds values typically considered acceptable by the theoretical community [12], and the overall privacy loss per device may be unbounded (Section 4).

## 2 OVERVIEW

### 2.1 System Components

We start by listing the components of the DP system on Mac OS we have identified:

- The differential privacy framework, located at `/System/Library/PrivateFrameworks/DifferentialPrivacy.framework`. The framework contains code implementing differential privacy, which we will decompile with Hopper Disassembler. In particular, it contains code responsible for per-datum privatization and for periodic functions that manage the privacy budget, updates of the database for privatized data, and creation of report files to be submitted to Apple servers.
- The `com.apple.dprivacyd` daemon handling differential privacy, located at `/usr/libexec/dprivacyd`. We will study it using code tracing with LLDB.
- A database, located at `/private/var/db/DifferentialPrivacy`, which contains several tables of privatized records and a table related to available budget per record type. Anyone with `sudo` privileges can open the database using `sqlite3`. We will study its contents (Section 3.1) and the changes to them due to usage of features that are supposed to trigger differentially private data collection and over time.
- Configuration files (Figures 4, 5, 6 and 7) with extension `.plist`, located at `/System/Library/DifferentialPrivacy/Configuration/`. The four files, which can be inspected by anyone but are difficult to change, specify numerous parameters that configure the actions of the DP framework, such as the per datum privacy parameter, privacy budget increase rate, etc. We will study the effects of each of these parameters by changing them and observing their effects during code execution with LLDB and through the resulting report files produced.
- Report files (Figure 11) with extensions `.dpsub` and `.json.anon`, located at `/Library/Logs/DiagnosticReports/` and

`/private/var/db/DifferentialPrivacy/Reports/`. These files contain privatized data and are the ones transmitted to Apple’s servers. They can be opened with a text editor, and the `.dpsub` files can also be inspected through the MacOS Console under System Reports. We will study when they get created, their contents, and when they get deleted through observations and experiments.

- The MacOS Console (Figure 21), which contains messages mentioning differential privacy, either in the library or process name. The messages are timestamped and easily readable, and are thus useful in noting certain system actions.

### 2.2 System Organization and Data Pipeline

The `dprivacy` (`com.apple.dprivacyd`) daemon runs the system responsible for implementation of differential privacy. Once a user opts-in to differentially private data collection in the MacOS Security & Privacy Settings (Figure 8), the `dprivacy` daemon is enabled and the database that will be supporting relevant data storage and management is created in `/var/db/DifferentialPrivacy`. Furthermore, there’s a message visible on Console: “`dprivacyd: accepting work now`”.

Per Apple’s original announcement [1, 21, 23], the use of DP is focused on four applications: new words, emojis, deeplinks, and lookup hints in Notes, with iCloud data added as an additional application in early 2017 [2], and further types of data collection such as health data introduced in mid-2017 [24]. We observed how to reliably trigger DP-related activity when entering new words and emojis<sup>2</sup>; thus, our conclusions will be based on experiments with those applications.

Whenever a user enters an emoji or a previously unseen new word in Notes, the relevant datum is perturbed using a differentially private algorithm and its privatized version and some metadata are added to a corresponding database table.

A `ReportGenerator` task (Figure 10) is run periodically, at which point some records from the database are selected and written to report files (Figure 11), which are then transmitted to Apple’s servers. The table rows corresponding to the selected records are “marked as submitted” and eventually deleted from the database by a task.

There are several other periodic maintenance tasks, whose effects are: to delete records from the database (even those that weren’t submitted) and to delete report files from disk. These periodic tasks are accompanied by messages observable on the Console (Figure 21).

### 2.3 Study Questions

In order to understand the privacy loss in Apple’s implementation of differential privacy, we need to understand the following aspects of the system:

- (1) What are the privacy parameters used in order to achieve privatization before the privatized datum gets entered into the database? This will let us understand per datum privacy.

<sup>2</sup>To reliably trigger DP application to emojis, the user needs to call out the emoji keyboard by pressing “`ctrl-cmd space`”, then click an emoji (or select an emoji with the arrow key and press “`Enter`”). For new words, the user can type an incorrectly spelled word in the Notes app and then ignore the spelling suggestion by pressing “`esc`”.

- (2) How frequently are records selected for inclusion in a report? How many records can be included in one report? How frequently are the reports created and submitted? This will let us understand the rate of privacy loss.
- (3) Is the total privacy loss that a particular user can incur limited?
- (4) How easy is it to alter the performance of the system, e.g., change parameters responsible for each datum's privatization, change the number of records selected for inclusion into a report or the frequency of report generation?

We discovered that the answers to questions (1) – (3) (see Section 4) depend on the parameters specified in the configuration files and their use by the framework to establish the available privacy budget. We describe our findings and observations regarding the database tables, configuration files, and the functionality related to report generation and privacy budget maintenance next. We will discuss (4) in Section 5.2.

### 3 SYSTEM'S DETAILS

#### 3.1 The Database

The ZOBHRECORD (See Figure 1) and ZCMSRECORD tables in the database store the perturbed data, with the former dedicated to the privatized emoji records, and the latter – to the privatized new words records. Every emoji typed by a user gets privatized and stored in the ZOBHRECORD table. In contrast, only words that haven't been previously typed are privatized and stored in the ZCMSRECORD table.

A notable table is ZPRIVACYBUDGETRECORD, whose schema and example contents are shown in Figure 2. The table contains 7 entries, one for each of the applications (NewWords, Emoji, AppDeepLink, Search, and health) and two for helper functions (default and testBudget). The ZBALANCE column contains the integer value of the currently available privacy budget for each application.

```
CREATE TABLE ZOBHRECORD ( Z_PK INTEGER PRIMARY KEY, Z_ENT INTEGER, Z_OPT INTEGER,
ZREPORTVERSION INTEGER, ZSUBMITTED INTEGER, ZSEQUENCEBITPOSITION INTEGER,
ZSEQUENCEBITVALUE INTEGER, ZFRAGMENTBITPOSITION INTEGER, ZFRAGMENTBITVALUE INTEGER,
ZFRAGMENTPOSITION INTEGER, ZFRAGMENTWIDTH INTEGER, ZCREATIONDATE TIMESTAMP, ZKEY
VARCHAR, ZPLAINSEQUENCE VARCHAR, ZPLAINFRAGMENT VARCHAR );

select * from zobhrecord;
2|7|1|11|0|4919|1|||518421187.24894|com.apple.keyboard.Emoji.en_US.EmojiKeyboard|
3|7|1|11|0|2432|0|||518421190.866433|com.apple.keyboard.Emoji.en_US.EmojiKeyboard|
4|7|1|11|0|2051|0|||518421193.667271|com.apple.keyboard.Emoji.en_US.EmojiKeyboard|
5|7|1|11|0|1269|0|||518421196.684048|com.apple.keyboard.Emoji.en_US.EmojiKeyboard|
6|7|1|11|0|1184|0|||518421199.336346|com.apple.keyboard.Emoji.en_US.EmojiKeyboard|
```

Figure 1: Schema and entries in the ZOBHRECORD table.

```
CREATE TABLE ZPRIVACYBUDGETRECORD ( Z_PK INTEGER PRIMARY KEY, Z_ENT INTEGER, Z_OPT
INTEGER, ZBALANCE INTEGER, ZCREATIONDATE TIMESTAMP, ZLASTUPDATE TIMESTAMP, ZKEY VARCHAR );

sqlite> select * from zprivacybudgetrecord;
1|9|35|115|516767088.54269|523679088.54269|com.apple.keyboard.NewWords
2|9|1|1|516767088.544767|516767088.544767|com.apple.DifferentialPrivacy.default
3|9|1|2|516767088.545904|516767088.545904|com.apple.health
4|9|35|324|516767088.546588|523679088.546588|com.apple.differentialprivacy.testBudget
5|9|31|36|516767088.547229|523333488.547229|com.apple.keyboard.Emoji
6|9|1|10|516767088.547831|516767088.547831|com.apple.parsec.AppDeepLink
7|9|35|73|516767088.54843|523679088.54843|com.apple.parsec.Search
sqlite>
```

Figure 2: Schema of and privacy budget items from the ZPRIVACYBUDGETRECORD table.

#### 3.2 Configuration Files

There are four configuration files for the DP daemon: `com.apple.dprivacyd.{keynames, keyproperties, algorithmparameters, budgetproperties}.plist`. See Figures 4, 5, 6 and 7 for snippets of the configuration files, Figure 3 for a schematic relationship between keys in them, and Tables 1 and 2 for a snippet of their values.

**3.2.1 KeyName  $\rightarrow$  PropertiesName.** `keynames.plist` contains a mapping of KeyNames to PropertiesNames. KeyNames describe the possible data types, e.g., `com.apple.keyboard.NewWords.en_US` – a new word in English using the US keyboard<sup>3</sup>, `com.apple.keyboard.NewWords.en_GB` – a new word in English using the Great Britain keyboard, `com.apple.keyboard.NewWords.ru_RU` – a new word in Russian, `com.apple.keyboard.Emoji.fr_FR.EmojiKeyboard` – an emoji in French, `com.apple.parsec.AppDeepLink` – a deeplink. In MacOS 10.12.3 `keynames.plist` contains 160 distinct KeyNames.

Each KeyName is assigned one of 13 possible PropertiesName. For example, KeyName `com.apple.keyboard.NewWords.en_US` has a PropertiesName `NewWords`, as do `com.apple.keyboard.NewWords.en_GB` and `com.apple.keyboard.NewWords.ru_RU`; KeyName `com.apple.parsec.AppDeepLink` has a PropertiesName `DeepLinks`; KeyName `com.apple.keyboard.Emoji.fr_FR.EmojiKeyboard` has a PropertiesName `TermFrequency`, as do `com.apple.keyboard.Emoji.ru_RU.EmojiKeyboard` and `com.apple.keyboard.Emoji.en_US.EmojiKeyboard`.

**3.2.2 Determining the Per-Datum Privacy Loss: KeyName  $\rightarrow$  PropertiesName  $\rightarrow$  PrivatizationAlgorithm, PrivacyParameter.** For each of the 13 possible PropertiesName values, the `keyproperties.plist` file specifies a PrivatizationAlgorithm and PrivacyParameter. For example, for PropertiesName=HealthDataTypes: PrivatizationAlgorithm=OneBitHistogram and PrivacyParameter=1; for PropertiesName=NewWords: PrivatizationAlgorithm=CountMedianSketch and PrivacyParameter=2; for PropertiesName=TermFrequency: PrivatizationAlgorithm=OneBitHistogram and PrivacyParameter=1.

`algorithmparameters.plist` specifies additional parameters of the privatization algorithm.

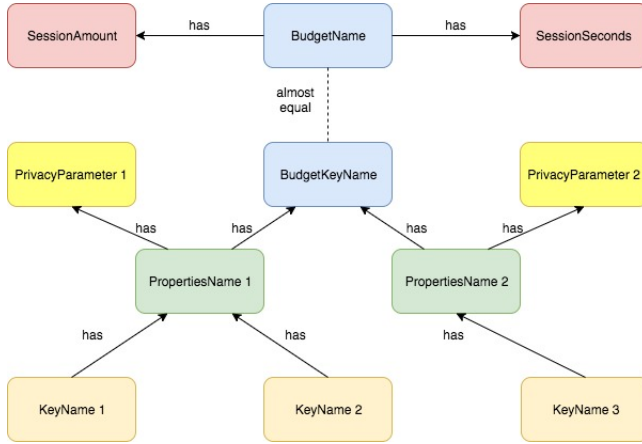
**3.2.3 Determining a Budget for Particular Data Types: KeyName  $\rightarrow$  PropertiesName  $\rightarrow$  BudgetKeyName.** Furthermore, for each of the 13 possible PropertiesName values, the `keyproperties.plist` file specifies a BudgetKeyName. For example, for PropertiesName=LocalWords: BudgetKeyName=`com.apple.keyboard.NewWords`; for PropertiesName=NewWords: BudgetKeyName=`com.apple.keyboard.NewWords`; for PropertiesName=DeepLinks: BudgetKeyName=`com.apple.parsec.AppDeepLink`; for PropertiesName=TermFrequency: BudgetKeyName=`com.apple.keyboard.Emoji`. In particular, this implies that all new words, regardless of language, have the same BudgetKeyName=`com.apple.keyboard.NewWords`, all emojis – the same BudgetKeyName=`com.apple.keyboard.Emoji`, etc. There are a total of 7 distinct BudgetKeyNames, which is consistent with what we saw in the ZPRIVACYBUDGETRECORD table in the database (Figure 2).

<sup>3</sup>We are not certain whether the second identifier encodes a keyboard preference or a region preference, or both.

KeyName	PropertiesName	Privatization-Algorithm	Privacy-Parameter	BudgetName	Session-Amount
com.apple.keyboard.NewWords.it_IT	NewWords	CountMedianSketch	2	com.apple.keyboard.NewWords	2
com.apple.keyboard.NewWords.ru_RU	NewWords	CountMedianSketch	2	com.apple.keyboard.NewWords	2
com.apple.keyboard.NewWords.zh_Hans	NewWordsChinese	CountMedianSketch	2	com.apple.keyboard.NewWords	2
com.apple.keyboard.LocalWords.en_US	LocalWords	CountMedianSketch	2	com.apple.keyboard.NewWords	2
com.apple.keyboard.Emoji.fr_FR.Emoji	TermFrequency	OneBitHistogram	1	com.apple.keyboard.Emoji	1
com.apple.parsec.AppDeepLink	DeepLinks	CountMedianSketch	1	com.apple.parsec.AppDeepLink	10
com.apple.health.datatypes	HealthDataTypes	OneBitHistogram	1	com.apple.health	2
com.apple.lookup.QueryMatch. query-Only.notHighlighted	Search	OneBitHistogram	1	com.apple.parsec.Search	1
com.apple.lookup.DomainMatch	Search	OneBitHistogram	1	com.apple.parsec.Search	1

**Table 1: Values for particular KeyNames: PropertiesName, PrivatizationAlgorithm, PrivacyParameter and BudgetName**

BudgetKeyName	Session-Seconds	Session-Amount
com.apple.keyboard.Emoji	86400	1
com.apple.parsec.Search	86400	1
com.apple.keyboard.NewWords	86400	2
com.apple.parsec.AppDeepLink	86400	10
com.apple.health	604800	2
com.apple.differentialprivacy.testBudget	86400	4
com.apple.DifferentialPrivacy.default	86400	1

**Table 2: Budget Properties as specified in *budgetproperties.plist* configuration file on MacOS 10.12.3****Figure 3: Relation between KeyName, PropertiesName, BudgetKeyName, BudgetName, etc.**

**3.2.4 Budget Properties.** The *budgetproperties.plist* file specifies two quantities for each BudgetKeyName: SessionSeconds and SessionAmount (see Figure 7 for a snippet and Table 2 for the values).

SessionSeconds = 86400 for all BudgetKeyNames except com.apple.health, for whom SessionSeconds = 604800. These correspond to number of seconds in a day and in a week, respectively.

SessionAmount values range from 1 to 10, depending on BudgetKeyName (see Table 2).

```
<dict>
  <key>KeyName</key>
  <string>com.apple.keyboard.NewWords.en_US</string>
  <key>PropertiesName</key>
  <string>NewWords</string>
</dict>
```

**Figure 4: Example of keynames.plist**

```
<dict>
  <key>PropertiesName</key>
  <string>NewWords</string>
  <key>DataAlgorithm</key>
  <string>SequenceDiscoveryWithPuzzlePiece</string>
  <key>FragmentWidth</key>
  <integer>2</integer>
  <key>FragmentCount</key>
  <integer>5</integer>
  <key>PrivatizationAlgorithm</key>
  <string>CountMedianSketch</string>
  <key>BudgetKeyName</key>
  <string>com.apple.keyboard.NewWords</string>
  <key>PrivacyParameter</key>
  <integer>2</integer>
  <key>Transport</key>
  <string>crashcopier</string>
  <key>ServerAlgorithmString</key>
  <string>SequenceFragmentPuzzle+CountMedianSketch</string>
  <key>KeyPatternsAllowed</key>
  <false/>
  <key>SubmissionPriority</key>
  <integer>1</integer>
</dict>
```

**Figure 5: Example of keyproperties.plist**

## 4 PRIVACY LOSS FINDINGS

We now answer the questions posed in Section 2.3.

### 4.1 Each Datum's Privatization

**PrivacyParameter of the KeyName** specifies the privacy parameter used for privatization of the datum of that KeyName prior to its addition to the database.

```

<dict>
  <key>ServerAlgorithmString</key>
  <string>SequenceFragmentPuzzle+CountMedianSketch</string>
  <key>AlgorithmParameters</key>
  <dict>
    <key>epsilon</key>
    <integer>4</integer>
    <key>sequenceFragmentBias</key>
    <real>0.5</real>
    <key>puzzleCount</key>
    <integer>256</integer>
    <key>sequenceM</key>
    <integer>32768</integer>
    <key>sequenceK</key>
    <integer>285</integer>
    <key>fragmentM</key>
    <integer>32768</integer>
    <key>fragmentK</key>
    <integer>285</integer>
  </dict>
</dict>

```

Figure 6: Example of algorithmparameters.plist

```

<dict>
  <key>BudgetName</key>
  <string>com.apple.keyboard.NewWords</string>
  <key>SessionSeconds</key>
  <real>86400</real>
  <key>SessionAmount</key>
  <integer>2</integer>
</dict>

```

Figure 7: Example of budgetproperties.plist

For example, for an emoji in French, Russian, or English, the privatization algorithm that will be run is OneBitHistogram with PrivacyParameter 1. For a new word in English using the US keyboard or a new word in Russian, the privatization algorithm that will be run is CountMedianSketch with PrivacyParameter 2, etc. See Table 1 for the PrivacyParameter values used for the various datum types in Mac OS 10.12.3.

It is difficult to understand and verify correctness of the privatization algorithms when one only has access to the binary code. We expect that the algorithms implement the ideas described in the patents [17–19]. What we do verify using LLDB and by changing the PrivacyParameter in the configuration file and observing the effects, is that the privacy parameter epsilon used for emoji and new words privatization is what one would expect based on the values in the configuration file (see Section A.1.1).

## 4.2 Report Generation and Privacy Budget Management over Time

SessionAmount specifies how many records belonging to a particular KeyName can be included in a report file. SessionAmount also specifies the increase to the available budget balance for each BudgetKeyName that happens every SessionSeconds.

**4.2.1 Number of Records per Report.** Specifically, Apple keeps track of the privacy budget balance (ZBALANCE) available for each of the 7 BudgetKeyNames in the ZPRIVACYBUDGETRECORD database table. The available budget balance together with the SessionAmount is used to decide how many records to include in each report file. Specifically, every 18 hours, the daemon runs a ReportGenerator task. It selects records from the database tables to be included in the report according to the following:

- At most  $\min(\text{SessionAmount}, 40)$ <sup>4</sup> records per KeyName may be selected.
- The total number of records belonging to the same BudgetKeyName selected may not exceed the privacy budget balance for that BudgetKeyName currently available as per the ZPRIVACYBUDGETRECORD table in the database. When the number of records of a particular BudgetKeyName available in the database tables exceeds the available budget balance, then the subset of records whose total number does not exceed the available budget balance are chosen at random while taking into account SubmissionPriority.

Each record selected is “marked as submitted” in the corresponding table in the database, and for each record submitted the corresponding privacy budget balance is decreased by one.

The report files created contain the creation time (or creation time adjusted forward by 7 hours) in the file name, and are placed in the folder /Library/Logs/DiagnosticReports/ or /private/var/db/DifferentialPrivacy/Reports/. Records with KeyNames that have TermFrequency, NewWords or LocalWords as their PropertiesName are included in reports in the first folder; records with KeyNames that have Search as PropertiesName – in the second folder.

**4.2.2 Budget Increase.** A periodic task PrivacyBudgetMaintenance increases the ZBALANCE value in the ZPRIVACYBUDGETRECORD table for each BudgetKeyName by its corresponding SessionAmount every SessionSeconds (see Section A.1.2 for experimental and code evidence supporting this claim). Thus, for all BudgetKeyNames except health and default, the available privacy budget balance is increased by SessionAmount every 24 hours<sup>5</sup>. Due to OS sleep, which is common on MacOS, in practice the daemon increases the privacy budget balance by the SessionAmount multiplied by the number of days that have elapsed since the last budget update (a time which is kept track of in the database table).

When a user opts-in to differentially private data collection, the budgets for each BudgetKeyName are initialized with their corresponding SessionAmount.

**4.2.3 Total Privacy Loss Permitted.** Consider an example: a user opts in to DP, then types one or several emoji every day for  $t$  days. Since for emoji PrivacyParameter=1, every emoji will be put into database after being privatized with privacy loss of 1. Since for emoji the SessionAmount=1, the privacy budget balance will be increased by one every day, and so every day one privatized emoji will be included in a report sent to Apple’s servers. After  $t$  days, by composition theorems [9, 15], the privacy loss incurred will be  $1 \cdot 1 \cdot t = t$ .

<sup>4</sup>The number 40 is hard-coded in the binary code (Figure 13).

<sup>5</sup>The default and AppDeepLink BudgetKeyNames are the exceptions to this; the former likely due to its role as a default value for abuse scenarios (Section 5.2) and the latter due to DeepLink functionality not being present in MacOS 10.12.

Consider another example: a user opts in to DP, but doesn't use any emoji for 20 days. The available privacy budget balance for emoji will be 20 after that time. Then the user types two emoji each in 10 different languages supported by differential privacy in one day. Each of the 20 emojis will be put into the database after being privatized with privacy loss of 1. In the first day, 10 different emoji, one from each language (since  $\text{SessionAmount} = 1$  for each emoji  $\text{KeyName}$  and available privacy budget balance is 20), will be included in the report, for a privacy loss of 10. The next day, the remaining 10 different emoji will be included in the report, for an additional privacy loss of 10.

In other words, the privacy loss for a particular application permitted by the implementation is  $\text{PrivacyParameter} \cdot \text{SessionAmount}$  every  $\text{SessionSeconds}$ . The privacy loss that is not realized during particular time period if that application is not used remains available for future use. Thus, for the applications of NewWords, AppDeepLink, Search, and Emoji, whose respective  $\text{PrivacyParameters}$  are: 2, 1, 1, 1, and  $\text{SessionAmounts}$  are: 2, 10, 1, 1, and  $\text{SessionSeconds}$  is 86,400 in MacOS 10.12.3, the overall daily privacy loss permitted is 16. Moreover, since unused privacy budget balance rolls over for subsequent use, the overall privacy loss of a device for the four initially announced applications by Apple may reach 16 times the number of days since the user of the device has opted in to DP. A caveat to these findings is that DeepLink functionality does not appear to be implemented on MacOS yet, so the actual privacy loss on Mac OS 10.12.3 is currently as large as 6 per day and on iOS 10.1.1 – as large as 14 per day (Section 5.4) for the four initially announced applications.

## 5 DISCUSSION

### 5.1 Report File and Database Maintenance

Besides the periodic tasks of ReportGenerator and PrivacyBudgetMaintenance, whose actions have already been described in Section 4.2, the following 3 periodic tasks are responsible for database and report file management (Figure 10):

- **StorageCulling** (every 24 hours): deletes records that have been submitted and records with the mismatched version number from the database.
- **StorageMaintenance** (every 12 hours): deletes records to limit database size and deletes records added to the database more than two weeks before the current date.
- **ReportFilesMaintenance** (every 24 hours): removes report files older than a month<sup>6</sup> from disk.

### 5.2 Ease of Altering System's Performance

We have observed several precautions that are implemented by Apple in order to make it difficult to abuse the implementation:

- The configuration files are difficult to change, as such a change on Mac OS requires turning off Apple's System Integrity Protection, which is not trivial. We have not found a way to change the configuration files on iOS.
- Even if one succeeds in changing the configuration files, whenever a  $\text{PrivacyParameter}$  in a configuration file is set

to a value higher than  $\text{epsilonMax}$ , a constant value equal to 2 which is hard-coded in the code of the framework (Figure 14), the  $\text{PrivacyParameter}$  used is defaulted to 1 at runtime and the corresponding record's Submission Priority is set to 99999, effectively ensuring it does not get included in report files. Furthermore,  $\text{SessionAmount}$  is defaulted to at most 40 at runtime (Section 4.2).

- Time measures used by the daemon, such as the number of seconds in 18 hours, in a day, in 7 days, are hardcoded in the code (Section A.1.3). That may be the reason why we have not succeeded in accelerating report file generation or privacy budget increase by changing the  $\text{SessionSeconds}$  in the configuration file or changing the computer's clock.

On the other hand, anyone with root permissions can alter the privacy budget balance in the database, thereby artificially increasing the privacy loss.

### 5.3 Configuration Differences between MacOS versions

We observed that Apple made changes to configuration files from MacOS 10.12.1 to 10.12.3 (see Table 3). The main distinctions are that in MacOS 10.12.3:

- $\text{SessionAmount}$  for `com.apple.keyboard.NewWords` increases from 1 to 2, resulting in a higher daily privacy loss.
- $\text{BudgetName}$  `com.apple.health` and  $\text{PropertiesName}$  `LocalWords` are introduced, signaling new applications for DP.
- $\text{SubmissionPriority}$  is adopted, signaling new protections against abuse.
- Health-related  $\text{PropertiesName}$  and  $\text{Budget}$  are introduced, signaling that health-related data will also be included in DP data collection.

	MacOS 10.12.1	MacOS 10.12.3
SessionAmount for NewWords	1	2
SessionAmount for testBudget	1	4
BudgetName com.apple.health	no	yes
PropertiesName LocalWords	no	yes
apple.photos.search.miss.unnormalized.en_US	yes	no
apple.photos.search.miss.normalized.en_US	yes	no
PropertiesName HealthDataTypes	no	yes
PropertiesName LocalWords	no	yes
SubmissionPriority	no	yes

**Table 3: Configuration file difference between MacOS 10.12.1 and 10.12.3**

### 5.4 A Note on iOS

Studying the iOS DP implementation is significantly more difficult. From our observations of the Console messages when the iPhone is connected to a computer running MacOS and of the reports (observable under Settings → Privacy → Analytics → Analytics Data), the iOS implementation follows the same principles as the MacOS one. The configuration files for iOS 10.1.1 we obtained from a jailbroken phone were identical to those of MacOS 10.12.1. The distinctions we found relate to iOS reports containing more metrics

<sup>6</sup>Concluded based on observation and dynamic code analysis, as we could not find this in the framework code.

(in particular, we were not able to trigger DeepLink functionality on MacOS while such records abound on iOS), and to faster report file deletion from the phone than from the computer (7 vs 30 days).

## 6 CONCLUSIONS AND FUTURE WORK

We applaud Apple for its deployment of differential privacy in the local privacy model and for the many safeguards put in place to make it difficult to abuse. However, we believe the deployment has several significant shortcomings.

- (1) The privacy loss permitted by the system is not explained anywhere and takes significant effort to reverse-engineer. This is contrary to one of the main conceptual advantages of differential privacy – that a user can make an informed choice whether to opt-in to differentially private data collection based on the quantifiable knowledge of risk announced by the data collector.
- (2) Furthermore, the lack of transparency on privacy loss opens the door for intentional or un-intentional abuse by Apple itself, e.g., by unilaterally changing either the per-datum privacy loss or the rate of privacy loss in a time period or by introducing additional BudgetKeyName(s), Apple may significantly weaken the privacy guarantees provided without anyone's knowledge or consent. In fact, this may already be happening – by inspecting iOS 11 beta report files (Figure 12), we have observed that the daily privacy loss permitted increased by at least 29 compared to that of iOS 10.1.1 and MacOS 10.12.3:
  - the PrivacyParameter used for Emoji is 2 (instead of 1) and as many as 10 records per Emoji KeyName (instead of 1 as in iOS 10.1.1 and MacOS 10.12.3) are included in one report file;
  - additional KeyNames, such as com.apple.safari.DomainVisited and com.apple.safari.DomainCausingEnergyDrain, are introduced, each using a PrivacyParameter of 1 and permitting as many as 10 records for their BudgetKeyName per report file.
- (3) The privacy loss of 16 per day permitted by the system is significantly higher than what is commonly considered reasonable in academic literature. Furthermore, since the permitted privacy loss balance is replenished every day, over a course of time the total privacy loss per device becomes larger by orders of magnitude.
- (4) Due to the way the database and report files are structured, the implementation leaks what features of MacOS a user is using and in what language and, possibly, with what geographic and keyboard preference, both to Apple and to anyone who has access to the database. Furthermore, because only new words are privatized and added to the relevant database tables, one can potentially test whether a particular non-dictionary word has been ever used by the owner of the device by observing whether typing it triggers changes to the DP database.

We call for Apple to make its implementation of privacy-preserving algorithms public and to make the rate of privacy loss fully transparent and tunable by the user.

**6.0.1 Future Work.** Apple does not transmit any user or device identifiers along with reports [21]. It would be worthwhile to investigate the effect that such (or other ways of) decoupling of data source from data aggregator can play in mitigating the implications of the (theoretically) infinitely increasing privacy loss [14].

It has been observed that properly implementing algorithms claiming to preserve differential privacy is non-trivial in practice [4, 5, 13]. It would be worthwhile to develop further techniques for verifying correctness of claimed DP implementations.

Finally, the question of how to intuitively interpret and convey the privacy guarantees and limitations of differential privacy at various privacy loss levels to the public, remains open.

## REFERENCES

- [1] Apple. 2016. Apple previews iOS 10, the biggest iOS release ever. (June 2016). <https://www.apple.com/newsroom/2016/06/apple-previews-ios-10-biggest-ios-release-ever.html>
- [2] Apple. 2017. macOS Sierra: Share analytics information with Apple. (March 2017). [https://support.apple.com/kb/PH25654?locale=en\\_US&viewlocale=en\\_US](https://support.apple.com/kb/PH25654?locale=en_US&viewlocale=en_US)
- [3] Greg Barbosa. Sep 26, 2016. Comment: Differential privacy and data collection is still not clearly defined as opt-in on iOS 10. In *9to5Mac*. <https://9to5mac.com/2016/09/26/comment-differential-privacy-and-data-collection-is-still-not-clearly-defined-opt-in-on-ios-10/>
- [4] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, César Kunz, and Pierre-Yves Strub. 2014. Proving Differential Privacy in Hoare Logic. In *Proceedings of the 2014 IEEE 27th Computer Security Foundations Symposium (CSF)*. 411–424.
- [5] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2016. Proving Differential Privacy via Probabilistic Couplings. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 749–758.
- [6] Cynthia Dwork. 2011. A firm foundation for private data analysis. *Commun. ACM* 54, 1 (2011), 86–95.
- [7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*. 265–284.
- [8] C. Dwork and G. J. Pappas. 2017. Privacy in Information-Rich Intelligent Infrastructure. *ArXiv e-prints* (June 2017). arXiv:1706.01985
- [9] 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.
- [10] Ulfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1054–1067.
- [11] European Association for Theoretical Computer Science. 2017. 2017 Gödel Prize. <https://eatscs.org/index.php/component/content/article/1-news/2450-2017-godel-prize>.
- [12] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C Pierce, and Aaron Roth. 2014. Differential privacy: An economic method for choosing epsilon. In *27th IEEE Computer Security Foundations Symposium (CSF)*. 398–410.
- [13] Ilya Mironov. 2012. On Significance of the Least Significant Bits for Differential Privacy. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS)*. 650–661.
- [14] Ilya Mironov. 2017. Private Communication. (2017).
- [15] Kobbi Nissim, Thomas Steinke, Alexandra Wood, Micah Altman, Aaron Benbenek, Mark Bun, Marco Gaboardi, David O'Brien, and Salil Vadhan. 2017. Differential Privacy: A Primer for a Non-technical Audience (Preliminary Version). (2017).
- [16] Erica Portnoy, Gennie Gebhart, and Starchy Grant. Sep 27, 2016. In *EFF DeepLinks Blog*. [www.eff.org/deeplinks/2016/09/facial-recognition-differential-privacy-and-trade-offs-apples-latest-os-releases](http://www.eff.org/deeplinks/2016/09/facial-recognition-differential-privacy-and-trade-offs-apples-latest-os-releases).
- [17] A.G. Thakurta, A.H. Vyrros, U.S. Vaishampayan, G. Kapoor, J. Freudiger, V.R. Sridhar, and D. Davidson. 2017. Learning new words. (March 14 2017). <https://www.google.com/patents/US9594741> US Patent 9,594,741.
- [18] A.G. Thakurta, A.H. Vyrros, U.S. Vaishampayan, G. Kapoor, J. Freudiger, V.R. Sridhar, and D. Davidson. 2017. Learning new words. (May 9 2017). <https://www.google.com/patents/US9645998> US Patent 9,645,998.
- [19] A.G. Thakurta, A.H. Vyrros, U.S. Vaishampayan, G. Kapoor, J. Freudinger, V.V. Prakash, A. Legendre, and S. Duplinsky. 2017. Emoji frequency detection and deep link frequency. (July 11 2017). <https://www.google.com/patents/US9705908>

US Patent 9,705,908.

- [20] Abhradeep Guha Thakurta. 2017. Differential Privacy: From Theory to Deployment. USENIX Association, Vancouver, BC.
- [21] WWDC 2016. June, 2016. *Engineering Privacy for Your Users*. <https://developer.apple.com/videos/play/wwdc2016/709/>.
- [22] WWDC 2016. June, 2016. *Platforms State of the Union*. <https://developer.apple.com/videos/play/wwdc2016/102/>.
- [23] WWDC 2016. June, 2016. *WWDC 2016 Keynote*. <https://www.apple.com/apple-events/june-2016/>.
- [24] WWDC 2017. June, 2017. *Privacy and Your Apps*. <https://developer.apple.com/videos/play/wwdc2017/702/>.

## A APPENDIX

### A.1 Code Support for Findings

The code convention for Objective-C, which Hopper disassembles the framework code to, is as follows. In an Objective-C function name, the leading `-` means this is an instance method that can only be accessed by an instance of the class, while `+` indicates the method is a class method and can be accessed anytime by simply referencing the class. The following `[]` includes both the function name and argument name, and arguments are separated by `:` (an example can be found in Figure 22).

*A.1.1 Checking PrivacyParameter corresponds to the privacy parameter used in a datum’s privatization.* For new words, we observed that the function `_DPCMSSample initWith` uses `PrivacyParameter` to create the `_DPBiasedCoin` (see Figure 16 for Hopper code and Figure 17 for our interpretation of it), and using runtime LLDB we observed that the value is  $1.0/(\exp(\text{PrivacyParameter}) + 1.0)$  (Figure 20). Analogously, we analyzed the emoji randomization code (Figure 22).

*A.1.2 Checking that SessionAmount controls the daily budget increase value.* In the `[_DPPPrivacyBudgetProperties initWithDictionary:]` function (Figure 18), the `SessionAmount` is assigned to `intervalBudgetValue`. In the `_DPPPrivacyBudget updateAllBudgetsIn` function (Figure 19), we can see the value used to multiply with the number of days since `ZLASTUPDATE` is `r14`, and `r14 = (r15 intervalBudgetValue) integerValue`. So we can conclude that `SessionAmount` is used as the daily budget increase value.

*A.1.3 Hard-coded values.* In addition to `epsilonMAX` and `_kSecondsInOneDay` (Figure 14), the following constants are also hardcoded in the framework code: `_kSecondsIn3Day`, `_kSecondsIn7Day`, `_kSecondsIn14Day`, `_kSecondsIn12Hours`, `_kSecondsIn18Hours`, `_kSecondsIn24Hours` (Figure 15).

### A.2 Aspects of the System that we don’t Understand

Several details of the implementation’s behavior were puzzling:

- The privacy budget balance sometimes changes dramatically. We observed this phenomenon twice over the course of 6 months. In both cases, the privacy budget increased and was set to  $\frac{ZLASTUPDATE - ZCREATIONDATE}{86400}$ . We don’t know the reason for this; one possibility is that the Apple server triggered this change remotely.

- The daemon occasionally automatically stops data collection and un-checks the boxes under Settings → Security & Privacy (Figure 8) indicating opt-in to Analytics Sharing (and hence, differential privacy). Over the course of our experiments in the last 6 months, we observed this effect several times. We were not able to reliably reproduce it. In particular, our experiments of setting the privacy budget balance to zero or a negative number in the database or entering thousands of emoji within a short period of time did not trigger opt-out.
- When `PrivacyParameter > epsilonMAX` (Section 5.2), the `NewWords` record will be inserted into the `ZOBHRECORD` table, even though it is typically inserted into the `ZCMSRECORD` table.
- We don’t understand the role of `testBudget`, one of the 7 `BudgetKeyNames`.

### A.3 Figures

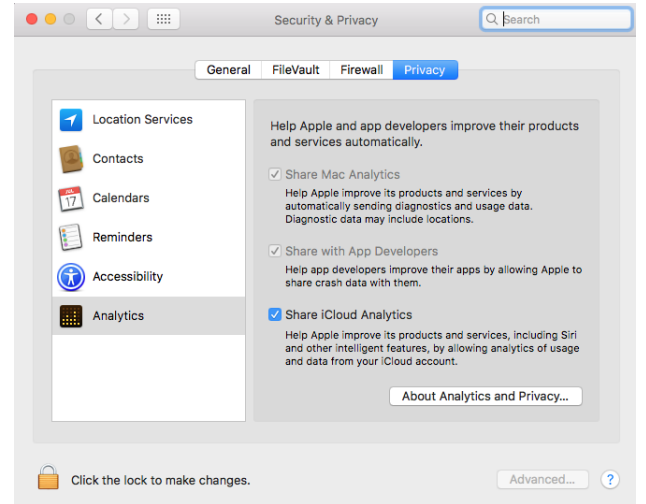


Figure 8: Screenshot of the opt-in interface.

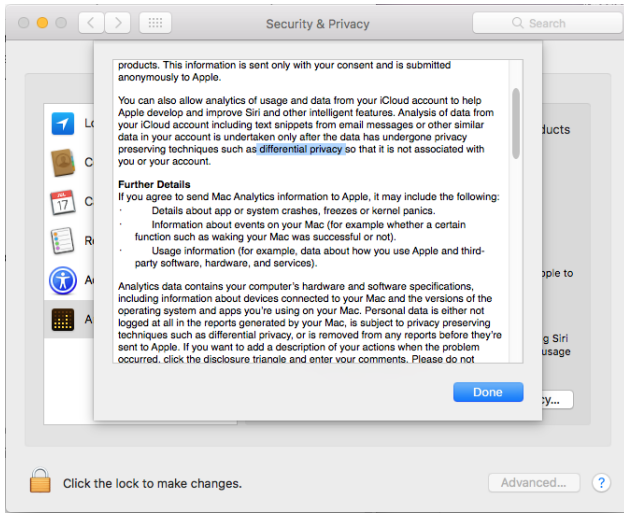


Figure 9: Screenshot of Apple mentioning DP in “About Analytics and Privacy”.

Message
Registered XPC Activity: com.apple.DifferentialPrivacy.StorageMaintenance
Registered XPC Activity: com.apple.DifferentialPrivacy.PrivacyBudgetMaintenance
Registered XPC Activity: com.apple.DifferentialPrivacy.ReportFilesMaintenance
Registered XPC Activity: com.apple.DifferentialPrivacy.ReportGenerator
Registered XPC Activity: com.apple.DifferentialPrivacy.StorageCulling

Figure 10: Five periodic maintenance tasks.

```
{
  "version": 11,
  "segments": [
    {
      "algorithm": "OneBitHistogram",
      "key": "com.apple.keyboard.Emoji.en_US.EmojiKeyboard",
      "parameters": {"epsilon": 1, "p": 5000},
      "records": [
        "49477827,+1"
      ]
    },
    {
      "algorithm": "SequenceFragmentPuzzle+CountMedianSketch",
      "key": "com.apple.keyboard.NewWords.en_US",
      "parameters": {"fragmentM": 32768, "fragmentCount": 5,
        "fragmentWidth": 2, "fragmentK": 285, "epsilon": 4,
        "sequenceM": 32768, "sequenceK": 285,
        "sequenceFragmentBias": 0.5, "puzzleCount": 256},
      "records": [
        "131,
        020020C00020000000E0001214000002401001042808601000080C0001:
        0001005000024890030000080001001019000120100700288C0080860201
```

Figure 11: An example report file (located in /Library/Logs/DiagnosticReports/), which includes an Emoji record and a (partial) NewWords record.

```
{
  "algorithm": "HadamardCountMedianSketch",
  "key": "com.apple.safari.DomainCausingEnergyDrain",
  "parameters": {"epsilon": 1, "k": 1024, "m": 32768},
  "records": [
    "818,165,+1",
    "301,25170,+1"
  ]
},
{
  "algorithm": "HadamardCountMedianSketch",
  "key": "com.apple.safari.DomainVisited",
  "parameters": {"epsilon": 1, "k": 1024, "m": 32768},
  "records": [
    "860,2389,+1",
    "689,12234,+1",
    "513,16288,+1",
    "754,13919,-1",
    "458,21448,-1",
    "0,3996,-1",
    "188,30563,+1",
    "127,30993,+1"
  ]
},
{
  "algorithm": "CountMedianSketch",
  "key": "com.apple.keyboard.Emoji.ru_RU.EmojiKeyboard",
  "parameters": {"epsilon": 2, "k": 32768, "m": 256},
  "records": [
    "19802,0000C0800000228206A04008110000000C080021211000046080040801040820",
    "17857,0020014080000400000900800D224020402A080040221100400000A300910101",
    "20312,800284858880000000200104000200001400800110424840004010000100800C",
    "11033,0020040010400001020000107000080820000400050402400008008010008062",
    "25035,84402200008020000A000401A94041000000040B01080000002031010800240",
    "4455,0200802028002000002000020000000200040100002000A301A1064280210800",
    "12670,11000008040C0510000000000801800080200220212C0016000000081000004",
    "3009,48411102802000000128000000040000300800006920700880000040240000",
    "15,0001000000010000C1700608920402580800000840050300009002021010090",
    "25960,000003840822004900810010000CC0000300009000100200100200020010500"
  ]
},
}
```

Figure 12: An example report file from iOS11.

```
void * +[_DPReportGenerator queryRecordsForKey:storage:]
| (void * self, void * _cmd, void * arg2, void * arg3) {
...
    r15 = [[_DPPrivacyBudgetProperties
        budgetPropertiesForKey:rdx] retain];
    rbx = [[r15 intervalBudgetValue] retain];
    r12 = [rbx integerValue];
    [rbx release];
    r13 = 0x28;
    if (r12 < 0x28) {
        r13 = r12;
    }
...
    [var_98 fetchRecordsFor:r15
        predicate:var_C0
        fetchLimit:r13
        fetchOffset:0x0
        withCompletion:&var_90];
...}
```

Figure 13: Code snippet for fetch records to submit (0x28 = 40).

```

00000000000041c30      db  0x00 ; '.'
00000000000041c31      db  0x00 ; '.'
00000000000041c32      db  0x00 ; '.'
00000000000041c33      db  0x00 ; '.'
00000000000041c34      db  0x00 ; '.'
00000000000041c35      db  0x00 ; '.'
00000000000041c36      db  0x00 ; '.'
00000000000041c37      db  0x40 ; '@'

_kSecondsInOneDay:
00000000000041c38      db  0x80 ; 'P'
00000000000041c39      db  0x51 ; 'Q'
00000000000041c3a      db  0x01 ; '.'
00000000000041c3b      db  0x00 ; '.'
00000000000041c3c      db  0x00 ; '.'
00000000000041c3d      db  0x00 ; '.'
00000000000041c3e      db  0x00 ; '.'
00000000000041c3f      db  0x00 ; '.'

```

**Figure 14:** epsilonMAX (equals to 2 if interpreted as a double type number) and number of seconds in one day are hard-coded in the code.

```

_kSecondsIn7Day:
00000000000041c48      db  0x80 ; '.'
00000000000041c49      db  0x3a ; ':'
00000000000041c4a      db  0x09 ; '.'
00000000000041c4b      db  0x00 ; '.'
00000000000041c4c      db  0x00 ; '.'
00000000000041c4d      db  0x00 ; '.'
00000000000041c4e      db  0x00 ; '.'
00000000000041c4f      db  0x00 ; '.'

_kSecondsIn14Day:
00000000000041c50      db  0x00 ; '.'
00000000000041c51      db  0x75 ; 'u'
00000000000041c52      db  0x12 ; '.'
00000000000041c53      db  0x00 ; '.'
00000000000041c54      db  0x00 ; '.'
00000000000041c55      db  0x00 ; '.'
00000000000041c56      db  0x00 ; '.'
00000000000041c57      db  0x00 ; '.'

_kSecondsIn12Hours:
00000000000041c58      db  0xc0 ; '.'
00000000000041c59      db  0xa8 ; '.'
00000000000041c5a      db  0x00 ; '.'
00000000000041c5b      db  0x00 ; '.'
00000000000041c5c      db  0x00 ; '.'
00000000000041c5d      db  0x00 ; '.'
00000000000041c5e      db  0x00 ; '.'
00000000000041c5f      db  0x00 ; '.'

```

**Figure 15: Example of other hardcoded variables.**

```
void *+[DPCMS$Sample dataFor:hashAtIndex:privacyParameter:bitCount:]
void * self, void * _cmd, void * arg2, unsigned long long arg3,
double arg4, unsigned long long arg5) {

    rcx = arg3;
    r14 = arg5;
    var_C8 = intrinsic_movsd(var_C8, arg4, arg2, rcx, arg5);
    rbx = arg2;
    asm { bswap     ecx };
    var_54 = rcx;
    r12 = [rbx retain];
    CC_SHA256_Init(&var_C0);
    CC_SHA256_Update(&var_C0, &var_54, 0x4);
    rbx = [objc_retainAutorelease(rbx) UTF8String];
    [r12 release];
    CC_SHA256_Update(&var_C0, rbx, strlen(rbx));
    CC_SHA256_Final(&var_50, &var_C0);
    var_D0 = (var_50 & 0xffff0000 | var_50 & 0xffffffff) % r14;
    exp(intrinsic_movsd(xmm0, var_C8));
    xmm1 = intrinsic_movsd(xmm1, *0x27ce8);
    xmm0 = intrinsic_addsd(xmm0, xmm1);
    intrinsic_movapd(xmm0, intrinsic_divsd(xmm1, xmm0));
    r13 = [_objc_msgSend(@class(_DPBiasedCoin),
        @selector(coinWithBias:)) retain];
    r14 = r14 >> 0x3;
    r15 = malloc(r14);
    if (r14 != 0x0) {
        var_C8 = r14;
        r12 = r15;
        do {
            *(int8_t *)r12 = [r13 generateByte];
            r12 = r12 + 0x1;
            r14 = r14 - 0x1;
        } while (r14 != 0x0);
    }
    else {
        var_C8 = r14;
    }
    *(int8_t *) (r15 + (var_D0 >> 0x3)) = *(int8_t *) (r15 +
        (var_D0 >> 0x3)) & 0xff ^ 0x1 << (var_D0 & 0x7);
    rbx = [NSData dataWithBytesNoCopy:r15 length:var_C8 retain];
    [r13 release];
    if (*__stack_chk_guard == *__stack_chk_guard) {
        rax = [rbx autorelease];
    }
    else {
        rax = __stack_chk_fail();
    }
    return rax;
}
```

**Figure 16: Code snippet from Hopper for new word randomization (partial).**

```

NSData *dataFor_hashAtIndex_privacyParameter_bitCount_(
    NSString * string, unsigned long hashAtIndex,
    double epsilon, int bc) {
    _byteswap_ulong ( hashAtIndex );
    CC_SHA1_CTX sha1_ctx;
    CC_SHA1_Init(&sha1_ctx);
    CC_SHA256_Update ( &sha1_ctx, &hashAtIndex, 4 );
    char *stringChars = [string UTF8String];
    int stringLen = strlen ( stringChars );
    CC_SHA256_Update ( &sha1_ctx, stringChars,
        stringLen );
    char md[32];
    CC_SHA256_Final(md, &sha1_ctx);
    DPBiasedCoin *coin = DPBiasedCoin::coinWithBias_(
        1.0 / ( exp( epsilon ) + 1.0 ) );
    int dataLen = bc >> 3;
    byte *dataBuffer = malloc( dataLen );
    for (int i = 0; i < dataLen; i++) {
        dataBuffer[i] = coin.generateByte();
    }
    dataBuffer[0>>3] ^= 1 << (0 & 7);
    return [NSData dataWithBytesNoCopy:
        dataBuffer length: dataLen];
}

```

Figure 17: Our interpretation of the code snippet for new word randomization (partial).

```

rbx = [var_48
    objectForKeyedSubscript:@"SessionAmount"];
r14 = [rbx retain];
r13 = r13 +
    *_OBJC_IVAR_$_DPPPrivacyBudgetProperties.
    _intervalBudgetValue;
objc_storeStrong(r13, rbx);

```

Figure 18: Code evidence that connects SessionAmount to intervalBudgetValue.

```

if (xmm0 <= 0x0) {
    var_1D0 = @selector(budgetRecord);
    var_250 = @selector(budgetProperties);
    var_258 = @selector(intervalBudgetValue);
    var_1E8 = @selector(integerValue);
    rcx = _objc_msgSend(var_1D0, var_1E8);
    xmm2 = intrinsic_movsd(xmm2, var_1B8);
    xmm0 = intrinsic_movapd(xmm0, xmm2);
    xmm1 = intrinsic_movsd(xmm1, *0x27c90);
    rdx = intrinsic_cvttsd2si(rdx,
        intrinsic_subsd(xmm0, xmm1));
    rax = intrinsic_cvttsd2si(0x8000000000000000,
        xmm2);
    xmm2 = intrinsic_ucomisd(xmm2, xmm1);
    asm { cmovae rax, rdx };
    r13 = rax / rcx;
    xmm0 = intrinsic_punpckldq(zero_extend_64(
        _objc_msgSend(var_1D0, var_1E8) * r13),
        *(int128_t *)0x27cb0);
    xmm0 = intrinsic_subpd(xmm0,
        *(int128_t *)0x27cc0);
    xmm0 = intrinsic_haddpd(xmm0, xmm0);
    var_240 = intrinsic_movapd(var_240, xmm0);
    xmm0 = intrinsic_ucomisd(xmm0, var_1B8);
    asm { cmova r13, rax };
    r15 = [_objc_msgSend(r12, var_250) retain];
    rbx = [_objc_msgSend(r15, var_258) retain];
    r14 = _objc_msgSend(rbx, var_1E8);
    [rbx release];
    [r15 release];
    rbx = [_objc_msgSend(r12, var_1D0) retain];
    [rbx release];
    if (rbx != 0x0) {
        rbx = [[r12 budgetRecord] retain];
        [rbx lastUpdate];
        xmm1 = intrinsic_movsd(xmm1, var_1B8);
        xmm1 = intrinsic_minsd(xmm1, var_240);
        xmm1 = intrinsic_addsd(xmm1, xmm0);
        xmm0 = intrinsic_movapd(xmm0, xmm1);
        [rbx setLastUpdate:rdx];
        [rbx release];
        rbx = [[r12 budgetRecord] retain];
        [rbx setBalance:[rbx balance] + r14 * r13];
        [rbx release];
        rbx = [[r12 budgetRecord] retain];
        [var_280 addObject:rbx];
        [rbx release];
    }
}

```

Figure 19: Code evidence that connects intervalBudgetValue to daily budget increase value.

```
(lldb)
Process 12047 stopped
* thread #2, queue = 'com.apple.NSXPCConnection.user.538', stop reason = instruction step over
    frame #0: 0x00000001080c4dd8 DifferentialPrivacy`+[DPCMSample dataFor:hashAtIndex:privacyParameter:bitCount:] + 251
DifferentialPrivacy`+[DPCMSample dataFor:hashAtIndex:privacyParameter:bitCount:]
-> 0x1080c4dd8 <+251>: movq    %rax, %rdi
    0x1080c4ddb <+254>: callq  0x1080c98b8          ; symbol stub for: objc_retainAutoreleasedReturnValue
    0x1080c4de0 <+259>: movq    %rax, %r13
    0x1080c4de3 <+262>: shrq    $0x3, %r14
(lldb) po $rax
_DPBiasCoin: { bias=0.1192029220221175 }
```

Figure 20: \_DPBiasedCon value from LLDB.

All Messages		Errors and Faults			
Devices	Date & Time	Type	Message	Process	Library
labmac2's MacB...	2017-08-08 08:22:04.782596		Initiating XPC Activity: com.apple.DifferentialPrivacy.StorageMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:04.782984		Initiating XPC Activity: com.apple.DifferentialPrivacy.ReportFilesMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:04.783689		Initiating XPC Activity: com.apple.DifferentialPrivacy.ReportGenerator	UserEventAgent	com.apple.cts
	2017-08-08 08:22:04.783937		Initiating XPC Activity: com.apple.DifferentialPrivacy.StorageCulling	UserEventAgent	com.apple.cts
	2017-08-08 08:22:04.785283		Initiating XPC Activity: com.apple.DifferentialPrivacy.PrivacyBudgetMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:05.663393		Completed XPC Activity: com.apple.DifferentialPrivacy.PrivacyBudgetMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:05.664057		Rescheduling XPC Activity: com.apple.DifferentialPrivacy.PrivacyBudgetMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:05.716204		Initiating XPC Activity: com.apple.DifferentialPrivacy.PrivacyBudgetMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:05.717239		Completed XPC Activity: com.apple.DifferentialPrivacy.PrivacyBudgetMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:05.717653		Rescheduling XPC Activity: com.apple.DifferentialPrivacy.PrivacyBudgetMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:05.727184		Completed XPC Activity: com.apple.DifferentialPrivacy.ReportFilesMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:05.727788		Rescheduling XPC Activity: com.apple.DifferentialPrivacy.ReportFilesMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:15.009100		Deleted 0 records with report version mismatch	dprivacyd	DifferentialPrivacy
	2017-08-08 08:22:15.009448		Deleted 0 records to enforce total size limit on the database	dprivacyd	DifferentialPrivacy
	2017-08-08 08:22:20.964109		Deleted 3 submitted records	dprivacyd	DifferentialPrivacy
	2017-08-08 08:22:20.964546		Completed XPC Activity: com.apple.DifferentialPrivacy.StorageCulling	UserEventAgent	com.apple.cts
	2017-08-08 08:22:20.965013		Deleted 112 old records	dprivacyd	DifferentialPrivacy
	2017-08-08 08:22:20.965066		Rescheduling XPC Activity: com.apple.DifferentialPrivacy.StorageCulling	UserEventAgent	com.apple.cts
	2017-08-08 08:22:20.966240		Completed XPC Activity: com.apple.DifferentialPrivacy.StorageMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:20.966652		Rescheduling XPC Activity: com.apple.DifferentialPrivacy.StorageMaintenance	UserEventAgent	com.apple.cts
	2017-08-08 08:22:30.906565		Completed XPC Activity: com.apple.DifferentialPrivacy.ReportGenerator	UserEventAgent	com.apple.cts
	2017-08-08 08:22:30.907818		Rescheduling XPC Activity: com.apple.DifferentialPrivacy.ReportGenerator	UserEventAgent	com.apple.cts

Figure 21: A screenshot of Console.app, which includes output from 5 DP periodic tasks.

```
void * -[DPOBHRandomizer randomizeStrings:forKey:](void * self, void * _cmd, void * arg2, void * arg3) {
    var_130 = self;
    r14 = [arg2 retain];
    var_128 = [arg3 retain];
    xmm0 = intrinsic_xorpd(xmm0, xmm0);
    rbx = [_objc_msgSend(@class(NSDate), @selector(dateWithTimeIntervalSinceNow:)) retain];
    [rbx timeIntervalSinceReferenceDate];
    var_138 = intrinsic_movsd(var_138, xmm0);
    [rbx release];
    var_150 = [x _NSArray0__ mutableCopy];
    xmm0 = intrinsic_xorpd(xmm0, xmm0);
    intrinsic_movapd(var_C0, xmm0);
    intrinsic_movapd(var_D0, xmm0);
    var_E0 = intrinsic_movapd(var_E0, xmm0);
    var_F0 = intrinsic_movapd(var_F0, xmm0);
    rax = [r14 retain];
    var_158 = rax;
    rax = [rax countByEnumeratingWithState:&var_F0 objects:&var_B0 count:0x10];
    var_100 = rax;
    if (rax != 0x0) {
        var_140 = *var_E0;
        do {
            r15 = 0x0;
            do {
                if (*var_E0 != var_140) {
                    objc_enumerationMutation(var_158);
                }
                r13 = [[var_130 randomizedBitForString:*(var_E8 + r15 * 0x8)] retain];
                xmm0 = intrinsic_movsd(xmm0, var_138, var_128, *(var_E8 + r15 * 0x8), [r13 index], ([r13 value] == 0x1 ? 0x1 : 0x0) & 0xff, 0x0, 0x0);
                rbx = [_objc_msgSend(@class(_DPOBHRSequenceRecord), @selector(recordWithKey:sequence:bitPosition:bitValue:creationDate:submitted:objectId:)) retain];
                if (rbx != 0x0) {
                    [var_150 addObject:rbx];
                }
                [rbx release];
                [r13 release];
                r15 = r15 + 0x1;
            } while (r15 < var_100);
            rax = [var_158 countByEnumeratingWithState:&var_F0 objects:&var_B0 count:0x10];
            var_100 = rax;
        } while (rax != 0x0);
    }
    [var_158 release];
    r14 = objc_retainAutoreleasedReturnValue(var_150);
    [var_150 release];
    [var_128 release];
    [var_158 release];
    if (*__stack_chk_guard == *__stack_chk_guard) {
        rax = r14;
    }
    else {
        rax = __stack_chk_fail();
    }
    return rax;
}
```

Figure 22: The emoji randomization code (partial) from Hopper Disassembler.