Analysis of Diagnosis Key distribution mechanism in contact tracing applications based on Google-Apple Exposure Notification (GAEN) framework

Guillaume Kessibi, Mathieu Cunche, Antoine Boutet, Claude Castelluccia, Cédric Lauradoux, Daniel Le Métayer, Vincent Roca

Analysis of Diagnosis Key distribution mechanism in contact tracing applications based on Google-Apple Exposure Notification (GAEN) framework

Guillaume Kessibi,
Mathieu Cunche[†] *
Antoine Boutet[†]
Claude Castelluccia
Cedric Lauradoux
Daniel Le Métayer
Vincent Roca

PRIVATICS Team, Inria, France
[†]INSA-Lyon, CITI Lab, France
desire-contact@inria.fr

July 07, 2020

1 Introduction

On April 10th 2020, Apple and Google jointly released a framework to facilitate the implementation of contact tracing applications based on BLE. The Google-Apple-Exposure-Notification (GAEN) framework is based on a *decentralized* model, in which temporary identifiers of infected users are regularly distributed to all active applications via a central server. This large scale distribution implies that, in practice, the temporary identifiers of infected users are public.

*Mathieu Cunche is the contact author. The other co-authors are listed in alphabetical order.

GAEN and other decentralized contact tracing systems [14] do not transmit identifiers per se, but rather transmit keys, called Temporary Exposure Keys (TEKs), that are used to derive identifiers; each TEKs is used to derive the set of temporary identifiers of the day, using a documented algorithm. Thus, the knowledge of the TEKs implies the knowledge of the identifiers.

The exposure of the TEKs of infected users, also called *Diagnosis Keys*, is at the source of several privacy threats affecting *decentralized* systems [15, 3]. In particular, all infected users will be exposed to: identification of their infection status, linkability of their identifiers, and thus location tracking. The tracking issue has been further demonstrated through a simulation [13], showing that mobility of infected users could be easily recorded by a network of Bluetooth receivers.

The impact of the exposure of this data is undeniable. Nevertheless it remains to be seen whether it is feasible to obtain this data. As this data needs to be distributed to millions of apps, access control appears to be challenging; especially since registration and user identification needs to be avoided for privacy reasons.

In their recent analysis [16], Vaudenay and Vuagnoux pointed that, in the GAEN-based SwissCovidapplication, the TEKs of infected users could be easily retrieved by downloading files from a webserver.

In an initiative for transparency [6], Farell and Leith have created a webpage [4] reporting the statistics on TEKspublished by three GAEN-based systems (Immuni, SwissCovidand Corona-Warn-App). They also published the scripts [5] used to download the TEKs from the servers of health authorities.

In this document we investigate how the TEKs of infected users, a.k.a. Diagnosis Keys, can be collected by an attacker. To this aim, we analyze four contact tracing systems built on the GAEN framework and currently active: the Italian application Immuni, the Swiss application SwissCovid, the German application Corona-Warn-App and the Latvian application Apturi Covid.

Based on a dynamic analysis of the Android app, we analyze how the keys are retrieved from the server. We found that the keys are stored in files (KeyFiles) hosted on a web server. It appears that the four systems are built on the sample implementation provided by Google [11, 9]; in particular the formatting of the KeyFiles is the one specified by Apple and Google. During our investigation, we have not identified any measure that could restrict the access to the KeyFiles. Finally, we found that Immuni and SwissCovid servers are deployed on third party infrastructures, respectively *Akamai* and *CloudFront*.

The findings presented in this document shows that, for those four applications, KeyFiles can be downloaded by anybody and that TEKs of infected users can be trivially extracted from those KeyFiles and used to derive the corresponding Bluetooth identifiers, exposing infected users to a range of privacy issues.

2 The GAEN framework

The *Google Apple Exposure Notification* (GAEN) framework has been introduced by Apple and Google on April 10th 2020. The purpose of GAEN is to "enable the use of Bluetooth technology to help governments and health agencies reduce the spread of the virus, with user privacy and security central to the design"[1].

The core of the GAEN is a software element controlled by Apple or Google and running on the device. The GAEN framework handles Bluetooth operations as well as the management of temporary identifiers used for contact tracing. The features of the GAEN framework are exposed to applications through a dedicated API. The access to this API is only granted to applications that are vetted by Apple and Google [8].

The GAEN framework is based on a *decentralized* contact tracing model, in which the exposure computations are performed on the device, based on data collected by the device over Bluetooth and data fetched from the server. In particular, the GAEN framework follows a model inspired from the first version of DP3T [14].

In addition to the framework running on the device, Google has released reference implementations for both the Android application [11] and the server [9] of the *decentralized* system.

2.1 Technical overview of GAEN

As other contact tracing systems, GAEN relies on the exchange between nearby devices of temporary identifiers over Bluetooth. In GAEN, those temporary identifiers are 16 bytes long and are called `Rolling Proximity Identifiers` (RPI). They are derived from a `Temporary Exposure Key` (TEK) using cryptographic operations [2]. `TEKs` are themselves rotated at a period defined as `EKRollingPeriod` which is set as 24 hours in current version of GAEN [2].

When the GAEN framework is active, it broadcasts its RPI and records the RPI of nearby devices along with some metatada. If a user is diagnosed positive, the app should transmit to the server the `TEKs` it has used over the last 14 days, i.e 14 `TEKs`. This set of `TEKs`, called `DiagnosisKeys` in GAEN, will then be added to a list of `TEKs` corresponding to infected users.

To verify the exposure status, each application fetches `TEKs` of infected users periodically on the server and submits them to the GAEN framework which compares the corresponding RPI to the locally stored RPI.

2.2 Google reference implementation

In addition to the document describing the GAEN API, Google has released a reference implementation for the application [11] and the server [9]. It includes documentation and code to build a fully functional contact tracing system based on

GAEN. In particular this reference implementation includes the code in charge of the distribution of the TEKs from the server to the application.

2.3 Exposure Key export file format

Google API documentation describes how TEKs must be submitted to the GAEN framework and specifies the file format that needs to be used [7][1]. The Exposure Key file format is defined as a zip archive containing two entries:

- export.bin: a binary containing temporary TEKs along with metadata.
- export.sig: a signature to verify export.bin.

Both files include a JSON-like data structure and are serialized with protobuff[2] (Google solution for data structure serialization). The export.bin file includes a list of TEKs, called Exposure Keys, and metadata (start and end dates, batch identifiers, and signature information). The exposure file includes the following elements [7]:

- start_timestamp: start of the time window of keys included in this file (based on arrival to server)
- end_timestamp: end of the time window of keys included in this file (based on arrival to server)
- region: identifier of the region of origin of the keys
- batch_num: index of the file in the batch
- batch_size: number of files composing the batch
- signature_infos: information about associated signatures
- keys: an array of TEKs:
  - key_data: the Temporary Exposure Key
  - rolling_start_interval_number: the interval number since epoch for which a key starts (by increments of 10 minutes since UTC)
  - rolling_period: increments of 10 minutes describing how long a key is valid
  - transmission_risk_level: risk associated with a key depending on diagnosis method

Thus the export.bin entry of a KeyFile looks like this example borrowed from Google documentation[3]:

Listing 1: Example of an export.bin file in json format

```
"EK Export v1    " // the special 16 byte header
TemporaryExposureKeyExport {
  start_timestamp: 1589068800
  end_timestamp: 1589155200
  region: "US"
```

[1] https://developers.google.com/android/exposure-notifications/exposure-key-file-format
[2] https://developers.google.com/protocol-buffers
[3] https://developers.google.com/android/exposure-notifications/exposure-key-file-format

```
  batch_num: 1
  batch_size: 1
  signature_infos: [
    SignatureInfo {
      verification_key_id: "gov.health.foo"
      verification_key_version: "v1"
      signature_algorithm: "1.2.840.10045.4.3.2"
    },
  ]
  keys = [...]
}
```

2.4 Transmission risk level

In a KeyFile, each key is associated with a `transmission_risk_level`, which provides metadata used by the GAEN framework to compute the risk score [10]. The `transmission_risk_level`, can take 8 different values listed below:

- 0: Unused
- 1: Confirmed test - Low transmission risk level
- 2: Confirmed test - Standard transmission risk level
- 3: Confirmed test - High transmission risk level
- 4: Confirmed clinical diagnosis
- 5: Self report
- 6: Negative case
- 7: Recursive case
- 8: Unused/custom

Value 0 is specified as unused. Values from 1 to 4 appear to correspond to people who have been identified as infected (through a test or a clinical diagnosis). The value 5 is reserved for self reported cases (less reliable information than 1-4). It is not clear what the purpose of values 6 (Negative case) and 7 (Recursive case) are. Finally, value 8 is reserved for custom uses (like testing).

2.5 KeyFiles & Index

KeyFiles are published on the server on a regular basis, each new file including the latest `TEKs` that have been reported to the server. Furthermore, to avoid large KeyFiles, it is possible to split a KeyFile in a batch of several files[4].

Thus, at a given time there will be several relevant KeyFiles on the server and the remote applications need to know which KeyFiles are to be fetched. Google

[4]It is possible to split a set of keys in a batch leveraging the entries `batch_size` and `batch_num`.

suggests using an *index file* to indicate the current relevant KeyFiles[5] need to be fetched.

3 Retrieving TEKs in deployed applications

We analysed four GAEN-based system, namely `Immuni` (Italy), `SwissCovid` (Switzerland), `Corona-Warn-App` (Germany) and `Apturi Covid` (Latvia), and we found that the `TEKs` can be retrieved by downloading files hosted on an open webserver.

Our analysis is based on a dynamic analysis of the corresponding Android applications. Using FRIDA[6], a tool for dynamic analysis of applications, we monitored calls that were associated with network operations. We were then able to identify HTTP requests made by the app to download index and KeyFiles.

Our analysis revealed that the four apps are based on the reference implementation provided by Google (see Section 2.2). Thus, those apps are sharing several elements that we present in Section 3.1. Nevertheless, the four apps differ by a number of elements such as the URI of file, format of the index file, and management of KeyFiles. Those specifics are detailed for each app in Sections 3.2, 3.3, 3.4 and 3.5.

3.1 Common elements

Being based on GAEN, the apps are using the format specified by Apple & Google for the KeyFiles (see Section 2.3). More specifically, the keyFiles are zip archives containing a binary file `export.bin` and a signature file `export.sig`. Both these files use the same formats as described in Section 2.3.

3.2 Immuni app – Italy

The `Immuni` app is the official contact tracing system deployed in Italy by the Ministry of Health. The code of the mobile application as well as the server are publicly available[7].

Hosting We found that for the `Immuni` system, the index and the Keyfiles are hosted on server behind the domain `get.immuni.gov.it`. This domain is a subdomain belonging to the Italian government (*Presidenza del Consiglio dei Ministri*).

[5] "We recommend that a consistent index file is used so that a client would download that index file to discover any new, unprocessed batches." `https://github.com/google/exposure-notifications-server/blob/master/docs/server_functional_requirements.md`

[6] `https://frida.re/`

[7] `https://github.com/immuni-app`

The server appears to be hosted on an *Akamai* infrastructure[8].

Both files are made available through TLS only and the certificate is issued by `Actalis S.p.A./03358520967` to `Sogei S.p.A. / Server Sicuri`.

Index file The index file `https://get.immuni.gov.it/v1/keys/index` is a JSON file with two integers, `oldest` and `newest`, designating a range of identifiers corresponding to key files.

Listing 2: Index file of the `Immuni` system on 25/06/2020 at `https://get.immuni.gov.it/v1/keys/index`

```
{
  "oldest": 5,
  "newest": 13
}
```

Keyfiles Keyfiles are identified by an integer and their URL is following the format `https://get.immuni.gov.it/v1/keys/<id>`, where `<id>` is the integer identifying the KeyFiles.

Listing 3: Content extracted of the `export.bin` of KeyFile 13 with the `TEKs` redacted `https://get.immuni.gov.it/v1/keys/13`.

```
start_timestamp: 1593010800
end_timestamp: 1593025200
region: "222"
batch_num: 1
batch_size: 1
signature_infos {
  verification_key_version: "v2"
  verification_key_id: "222"
  signature_algorithm: "1.2.840.10045.4.3.2"
  1: "it.ministerodellasalute.immuni"
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 8
  rolling_start_interval_number: 2654208
  rolling_period: 144
}
```

[8] `nslookup www.pt.bfs.admin.ch` returns `Non-authoritative answer: get.immuni.gov.it canonical name = immuni.gov.it.edgekey.net. immuni.gov.it.edgekey.net canonical name = e33050.e3.akamaiedge.net.`

```
keys {
  key_data: "REMOVED"
  transmission_risk_level: 8
  rolling_start_interval_number: 2653776
  rolling_period: 144
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 8
  rolling_start_interval_number: 2653920
  rolling_period: 144
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 8
  rolling_start_interval_number: 2654064
  rolling_period: 144
}
[...]
```

According to the timestamps contained in the KeyFiles, each KeyFile covers a period of 4 hours.

3.3 SwissCovid – Switzerland

The `SwissCovid` app is the official contact-tracing system deployed in Switzerland[9]. Its code is also public[10].

Hosting In the `SwissCovid` system, the index and the Keyfiles are hosted on a server behind the domain `www.pt.bfs.admin.ch` which is a subdomain belonging to the Swiss Federal Office for Telecommunicaitons (*Bundesamt für Informatik und Telekommunikation/Office fédéral de la communication*).

The server appears to be hosted on a *CloudFront* infrastructure[11] and the files hosted on this server can only be retrieved via a connection protected by TLS. Furthermore, the corresponding certificate is issued by *QuoVadis Limited* to *Bundesamt für Informatik und Telekommunikation BIT*.

[9] `https://www.bag.admin.ch/bag/fr/home/krankheiten/ausbrueche-epidemien-pandemien/aktuelle-ausbrueche-epidemien/novel-cov/swisscovid-app-und-contact-tracing.html`

[10] `https://github.com/DP-3T`

[11] `nslookup www.pt.bfs.admin.ch` returns `Non-authoritative answer: www.pt.bfs.admin.ch canonical name = da6aw9bvreynb.cloudfront.net.`

Indexing The SwissCovid system does not use an index file. Rather, it relies on the current date to identify the relevant KeyFiles that are identified by a timestamp. At the time of our study, the first batch would correspond to June 20 (1592611200) which is not yet 14 days apart from the current date.

KeyFiles The KeyFiles are named after a UNIX timestamp and are located at URLs following the format https://www.pt.bfs.admin.ch/v1/gaen/exposed/<id>. Here, <id> is the timestamp padded with zeros.

Listing 4: Content extracted of the export.bin of KeyFile with the TEKs redacted

```
start_timestamp: 1593561600
end_timestamp: 1593568800
region: "ch"
batch_num: 1
batch_size: 1
signature_infos {
  verification_key_version: "v1"
  verification_key_id: "228"
  signature_algorithm: "1.2.840.10045.4.3.2"
  1: "ch.admin.bag.dp3t"
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 0
  rolling_start_interval_number: 2655936
  rolling_period: 144
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 0
  rolling_start_interval_number: 2655936
  rolling_period: 144
}
...
```

3.4 CoronaWarnApp – Germany

The Corona-Warn-App is the official contact-tracing system deployed in Germany, its code is public[12].

[12] https://github.com/corona-warn-app/cwa-app-android/

Hosting In the Corona-Warn-App system, the index and KeyFiles are hosted on a server behind the domain svc90.main.px.t-online.de which is a subdomain belonging to *T-Online*, a division of *DeutschTelecom*. As opposed to Immuni and SwissCovid, the server appears to be hosted on *T-Online*'s own infrastructure[13].

The files hosted on this server can only be retrieved via TLS, and the corresponding certificate is issued by *DigiCert Inc* to *Deutsche Telekom AG*.

Index file The index file is located at https://svc90.main.px.t-online.de/version/v1/diagnosis-keys/country/DE/date, and it contains an array of strings corresponding to dates under the format "yyyy-mm-dd". Each element of this array identifies a KeyFile.

Listing 5: index on 28/06/2020 at /version/v1/diagnosis-keys/country/DE/date

```
["2020-06-23","2020-06-24","2020-06-25","2020-06-26","2020-06-27"]
```

KeyFiles KeyFiles are identified by a string representing a date and their URL follows the format /version/v1/diagnosis-keys/country/DE/date/<id>[14], where <id> is the date under the format "yyyy-mm-dd".

Listing 6: Content extracted of the export.bin of KeyFile with the TEKs redacted.

```
start_timestamp: 1593496800
end_timestamp: 1593547200
region: "DE"
batch_num: 1
batch_size: 1
signature_infos {
  verification_key_version: "v1"
  verification_key_id: "262"
  signature_algorithm: "1.2.840.10045.4.3.2"
  1: "de.rki.coronawarnapp"
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 8
  rolling_start_interval_number: 2655360
  rolling_period: 144
}
```

[13] nslookup svc90.main.px.t-online.de returns Non-authoritative answer: Name: svc90.main.px.t-online.de

[14] https://svc90.main.px.t-online.de/version/v1/diagnosis-keys/country/DE/date/<id>

```
keys {
  key_data: "REMOVED"
  transmission_risk_level: 8
  rolling_start_interval_number: 2655216
  rolling_period: 144
}
```

As the naming convention suggests, each file contains data for a given day. The timestamp included in the file indicates that each file covers a period of 12 hours ranging from 7am to 7 pm. Note that the file under the path `yyyy-mm-dd` contains data corresponding to the previous day, i.e. `yyyy-mm-(dd-1)` (e.g. the KeyFile named `2020-06-27` contains the data of the 26th).

3.5 `Apturi Covid` – Latvia

The `Apturi Covid` application is the official contact-tracing system deployed in Latvia. Its code is public[15].

Hosting In the `Apturi Covid` system, the index and KeyFiles are hosted on a server behind the domain `apturicovid-files.spkc.gov.lv` which is a subdomain belonging to the Latvian government. The server appears to be hosted on the government's own infrastructure[16].

The index hosted on this server can only be retrieved via TLS, while the KeyFiles can be obtained with or without TLS. The certificate used for TLS connections is issued by *Let's Encrypt* for the domain *apturicovid-files.spkc.gov.lv*[17].

Index file `Apturi Covid` also uses an index file but instead of using a JSON format, the paths to KeyFiles are given as a simple list of text. It follows the format `/dkfs/v1/index.txt` with files as:

Listing 7: Example of `Apturi Covid`'s index file, with paths redacted.

```
https://server.com/dkfs/v1/ID1.zip
https://server.com/dkfs/v1/ID2.zip
```

KeyFiles `Apturi Covid`'s KeyFiles are identified by an integer and are following the standard format of the GAEN framework.

[15] `https://github.com/ApturiCOVID/`

[16] `nslookup apturicovid-files.spkc.gov.lv` returns `Non-authoritative answer: Name: apturicovid-files.spkc.gov.lv Address: 212.70.163.116`

[17] The certificate does not include other subject fields.

Listing 8: Content extracted of the `export.bin` of KeyFile with the `TEKs` redacted

```
start_timestamp: 1594226904
end_timestamp: 1594226904
region: "247"
batch_num: 1
batch_size: 1
signature_infos {
  verification_key_version: "v1"
  verification_key_id: "247"
  signature_algorithm: "1.2.840.10045.4.3.2"
  1: "lv.spkc.gov.apturicovid"
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 1
  rolling_start_interval_number: 2656656
  rolling_period: 144
}
keys {
  key_data: "REMOVED"
  transmission_risk_level: 1
  rolling_start_interval_number: 2656512
  rolling_period: 144
}
```

4 Analysis of retrieved data

This section presents observations on the `TEKs` that have been retrieved from the studied systems.

A first general observation that applies to all systems, is that there is no overlapp between KeyFiles; i.e. there is no TEK that appears in more than one KeyFile.

4.1 `Immuni` – Italy

In `Immuni` all of the `TEKs` have a `transmission_risk_level` of 8 which, according to Google's documentation, means an unused or custom risk level.

Evolution of the number of `TEKs` in each KeyFile is presented in Figure 1. As each KeyFile corresponds to a period of 24 hours, this distribution provides an indication on the number of `TEKs` injected in the system per day. The number of `TEKs` vary from one day to the other, from a maximum of 28 `TEKs` in file 14 to a single TEK in files 10 and 18

It is not clear whether those TEKs are dummy or if they correspond to actual cases. They are not associated with a risk level value matching real cases (1-4), but with the value 8, that is dedicated to unused or custom uses. On the other hand, the high variation of the number of TEKs suggests that their injection is not automated and could thus be the results of real cases declarations.

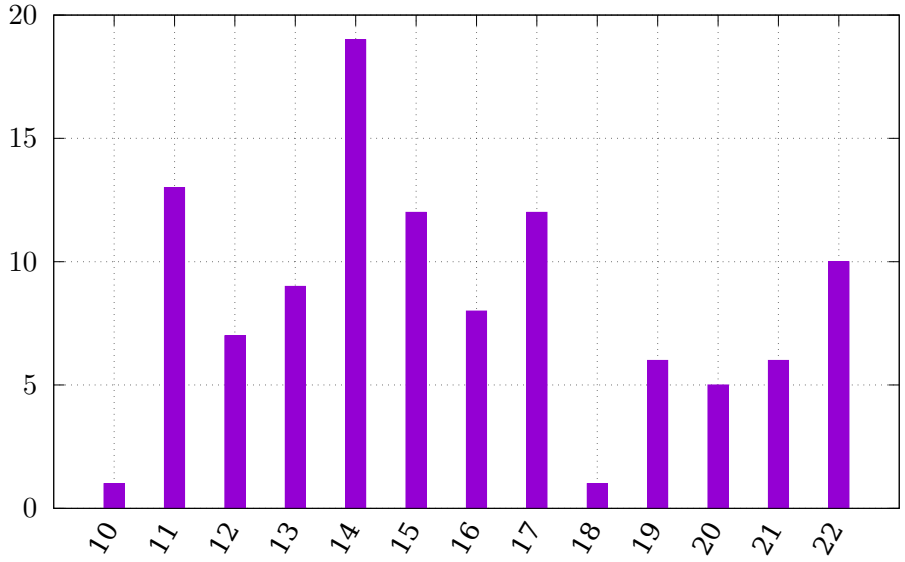

Figure 1: Distribution of the number of TEKs found in the keyfiles of Immuni.

4.2 SwissCovid – Switzerland

In the SwissCovidsystem, we found that a new KeyFile was released daily and that each KeyFile contains a number of TEKs that never ventures below 10 as seen on Figure 2. As of today (09/07/20), all the TEKs are associated with the risk level of 0, which means a dummy/unused risk, suggesting that no actual case has been reported to the SwissCovidsystem.

Therefore it seems that the dummy keys are regularly injected in the system, either for testing or privacy protection purposes as noted by Farell and Leith [6].

4.3 Corona-Warn-App – Germany

The number of TEKs published daily in Corona-Warn-Appis presented on Figure 3. The number of published TEKs greatly vary : 1450 TEKs were published on 30/06 while only 180 TEKs were published on the 05/07. Note that according to the github of Corona-Warn-App[18], random TEKs are injected as a mean to improve infected

[18] https://github.com/corona-warn-app/cwa-server/pull/609

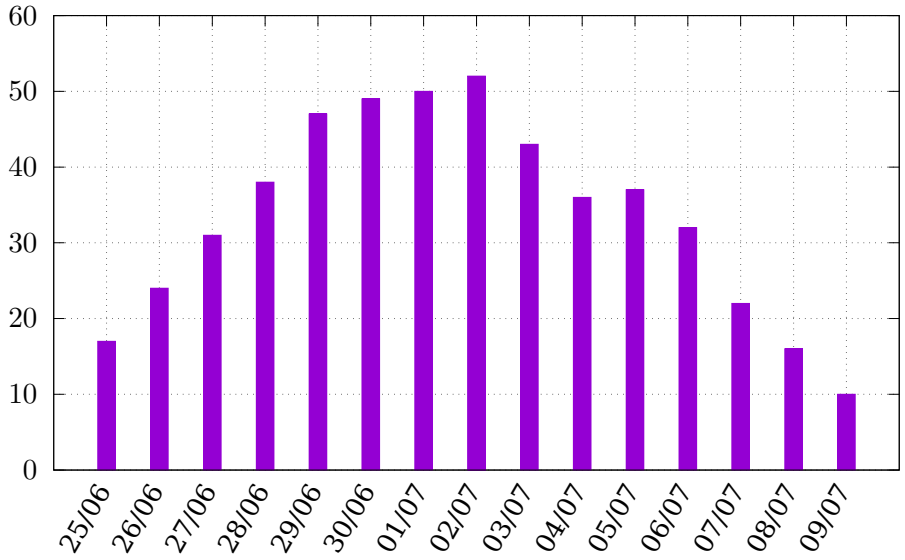

Figure 2: Number of TEKs found in the keyfiles of `SwissCovid`

users' privacy. The number of random `TEKs` is proportional to the number of real `TEKs` .

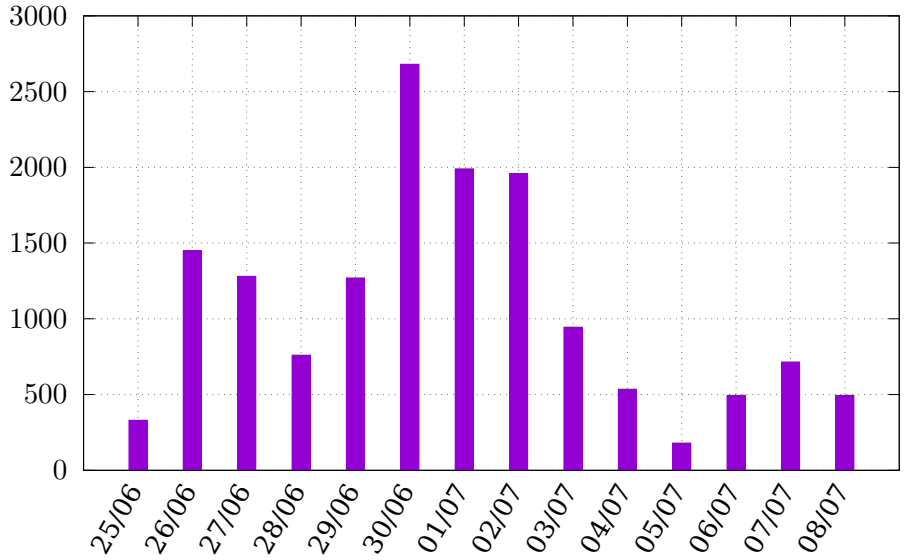

Figure 3: Amount of TEKs retrieved per day from `Corona-Warn-App`'s System

The `TEKs` published in `Corona-Warn-App` are associated to various risk levels as seen in Table 1. In addition to Confirmed test values (1-3), it also includes values

Table 1: Distribution of risk levels in Corona-Warn-App(09/07/2020)

transmission_risk_level	nb TEKs	percentage
0 (unused)	0	0%
1 (low risk)	6480	43%
2 (standard risk)	30	0.2%
3 (high risk)	1285	8.6%
4 (clinical diagnosis)	0	0%
5 (self report)	1340	8.9%
6 (negative case)	1540	10.3%
7 (recursive case)	70	0.5%
8 (unused/custom)	4170	28%

for self report, negative case, recursive case and unused/custom.

Focusing on the risk level 1-3, corresponding to Low, Standard and High transmission risk level, Figure 4 shows the evolution of TEKs associated to each of those risk values. A majority of the TEKs are associated with Low risk level (43%), leaving only a small portion of TEKs with Standard (0.2%) and High (8.6%) risk levels accounts.

The variation of the number of published TEKs along with the risk level associated to an actual transmission risk suggests that the TEKs published in Corona-Warn-App correspond to actual cases (not counting the random TEKs).

4.4 Apturi Covid – Latvia

As of today (09/07/20), there are only 4 TEKs available in Apturi Covid (see Figure 5). However, we can note that all of the transmission_risk_level values are of 1 (confirmed test, low transmission risk level). This risk level suggests that those TEKs correspond to real infection cases.

The relatively small number of published TEKs in Apturi Covidcan be explained by the small number of cases in the country[19]

5 Conclusion

In this document, we analyzed four applications based on *Google Apple Exposure Notification* (GAEN) and more particularly the key distribution mechanism use to communicate TEKs (a.k.a. *diagnosis keys*) of infected users to applications. We found that TEKs are packed in KeyFiles that are delivered through open webservers from which the TEKs can be easily extracted. As a consequence, in the four studied

[19] 15 cases per week https://eng.lsm.lv/article/society/health/health-ministry-to-review-covid-19-restrictions-in-latvia.a366610/

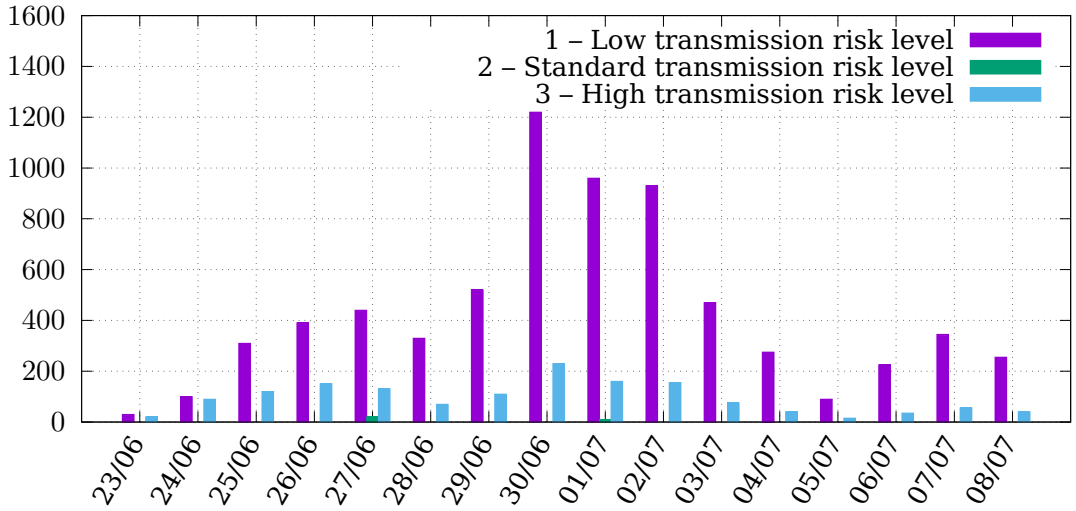

Figure 4: Distribution of *Confirmed test* transmission_risk_level (values 1-3) of `TEKs` published in `Corona-Warn-App`no confirmed clinical diagnosis were reported through TEKs yet.

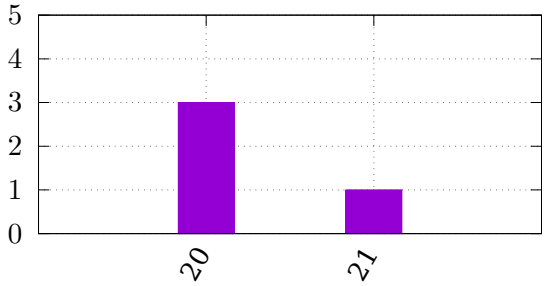

Figure 5: Amount of `TEKs` retrieved per day from `Apturi Covid`'s System

systems, the `TEKs` of infected users can be easily obtained by anybody. Actually, this collection of keys have already started, with researchers retrieving them to publish statistics [4].

The exposure of those `TEKs` significantly increases the feasibility of a number of privacy attacks, exposing infected users to serious privacy threats [15, 3].

We note that this problem of `TEKs` exposure is inherent to decentralized systems and in particular to solutions based on the GAEN framework. This exposure of information is accepted as being one of the drawback of decentralized systems [12, section 3.3] and it looks that there is no solution to prevent this exposure.

References

[1] Apple and Google. Privacy-Preserving Contact Tracing. https://www.apple.com/covid19/contacttracing.

[2] Apple and Google. Exposure Notification - Cryptography Specification v1.2. https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-CryptographySpecificationv1.2.pdf, April 2020.

[3] Antoine Boutet, Nataliia Bielova, Claude Castelluccia, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. Proximity Tracing Approaches - Comparative Impact Analysis. Research Report, INRIA Grenoble - Rhone-Alpes, April 2020.

[4] Stephen Farrell and Douglas J Leith. Testing Apps for COVID-19 Tracing (TACT) - TEK Survey. https://down.dsg.cs.tcd.ie/tact/tek-counts/.

[5] Stephen Farrell and Douglas J Leith. Collecting TEKs. https://github.com/sftcd/tek_transparency, July 2020.

[6] Stephen Farrell and Douglas J Leith. Transparency in the Deployment of Coronavirus Contact Tracing Apps. June 2020.

[7] Google. Exposure Key export file format and verification. https://developers.google.com/android/exposure-notifications/exposure-key-file-format.

[8] Google. Exposure Notifications: Helping fight COVID-19. https://www.google.com/intl/en_us/covid19/exposurenotifications/.

[9] Google. Exposure Notification Reference Key Server. https://github.com/google/exposure-notifications-server, June 2020. original-date: 2020-04-29T23:03:36Z.

[10] Google. Exposure Notifications Android API Documentation. https://static.googleusercontent.com/media/www.google.com/en//covid19/exposurenotifications/pdfs/Android-Exposure-Notification-API-documentation-v1.3.2.pdf, May 2020.

[11] Google. Exposure Notifications API: Android Reference Design. https://github.com/google/exposure-notifications-android, June 2020. original-date: 2020-05-04T15:30:21Z.

[12] The DP-3T Project. Privacy and Security Attacks on Digital Proximity Tracing Systems, April 2020.

[13] Otto Seiskari. BLE contact tracing sniffer PoC. https://github.com/oseiskar/corona-sniffer.

[14] Carmela Troncoso, Mathias Payer, Marcel Salathé, James Larus, Dr Wouter Lueks, Theresa Stadler, Dr Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Capkun, David Basin, Dennis Jackson, KU Leuven, Bart Preneel, Nigel Smart, Dr Dave Singelee, Dr Aysajan Abidin, TU Delft, Seda Guerses, and Cas Cremers. Decentralized Privacy-Preserving Proximity Tracing, April 2020.

[15] Serge Vaudenay. Analysis of DP3T. https://eprint.iacr.org/2020/399, 2020.

[16] Serge Vaudenay and Martin Vuagnoux. Analysis of SwissCovid. `https://chaosticino.ch/docs/20200605--vaudenay%2Bvuagnoux--analysis-of-swisscovid.pdf`, May 2020.