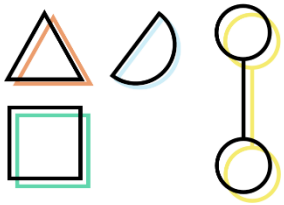
COVID Green/COVID Tracker & COVID Alert/COVID Shield

Contact Tracing Solutions
Threat Model and Audit Report



Prepared for the Open Source Technology Improvement Fund by Symbolic Software

Dr. Nadim Kobeissi, *Symbolic Software*
Sasha Lapiha, *Symbolic Software*

Contents

Executive Summary

COVID Green[1] and COVID Alert[2] are smartphone applications built around the Google Apple Exposure Notification (GAEN) API and targeting iOS and Android devices. COVID Green is sometimes labeled as "COVID Tracker" in public deployments. COVID Alert is sometimes labeled as "COVID Shield" in public deployments, with "COVID Alert" referring to the underlying framework used by COVID Shield.

The applications are intended to provide symptom reporting services, which may be confirmed by a licensed medical practitioner, allowing Irish and Canadian citizens to self-report and to therefore have any proximity contacts be notified of their potential exposure to the COVID-19 virus. COVID Green also provides useful news and information, such as the number of daily reported cases in the user's region in Ireland.

The applications are commissioned by the Irish and Canadian governments respectively. Symbolic Software was solicited by the Open Source Technology Improvement Fund (OSTIF)[3] in order to conduct an audit of the COVID Green and the COVID Alert client applications, in order to determine the applications' target security goals and to devise a threat model for the overall application stacks.

COVID Green and COVID Alert share great technical similarities: both clients are written using React Native[4] and loaded into an iOS and Android application wrapper. The server code relies on a loose collection of AWS and Hashicorp Terraform scripts in order to instantiate server-side operations, while the GAEN platform provides decentralized contact tracing based on the DP-3T contact tracing protocol.[5]

In auditing both applications, only minor issues were spotted, largely due to the application's simple design and reliance on the GAEN platform for much of the heavy lifting. However, the threat modeling of the COVID Green application led to the discovery of a number of serious potential issues, one of which may run against the spirit of the law in the context of the European General Data Protection Regulation (GDPR)[6]. The COVID Alert application also advertised some security and privacy goals that were found to be impossible to meet given the application's architecture, and this is also further documented in the contents of this report.

[1] https://github.com/covidgreen
[2] https://github.com/cds-snc/covid-alert-app
[3] https://ostif.org/
[4] https://reactnative.dev/
[5] https://github.com/DP-3T
[6] https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en

We have discussed our findings with both COVID Green and COVID Alert development teams, and many of the issues identified have already been addressed in response to our findings.

Scope and Coverage

In particular, code audit work focused on the following elements:

- **COVID Green Application Code:**
  - Exposure Notification Bindings
  - Calls to Server Infrastructure Components
- **COVID Green AWS and Server Interface:**
  - Diagnosis Confirmation and SMS Code Issuance API
- **COVID Alert Application Code:**
  - Exposure Notification Bindings
  - Calls to Server Infrastructure Components
- **COVID Alert "Rationale" Document**
  - Served as basis for Threat Model determination

The COVID Green and COVID Alert server codebases was specifically delineated as being outside the scope of this audit, but elements that interact with the server were still frequently tested and reviewed as part of the audit.

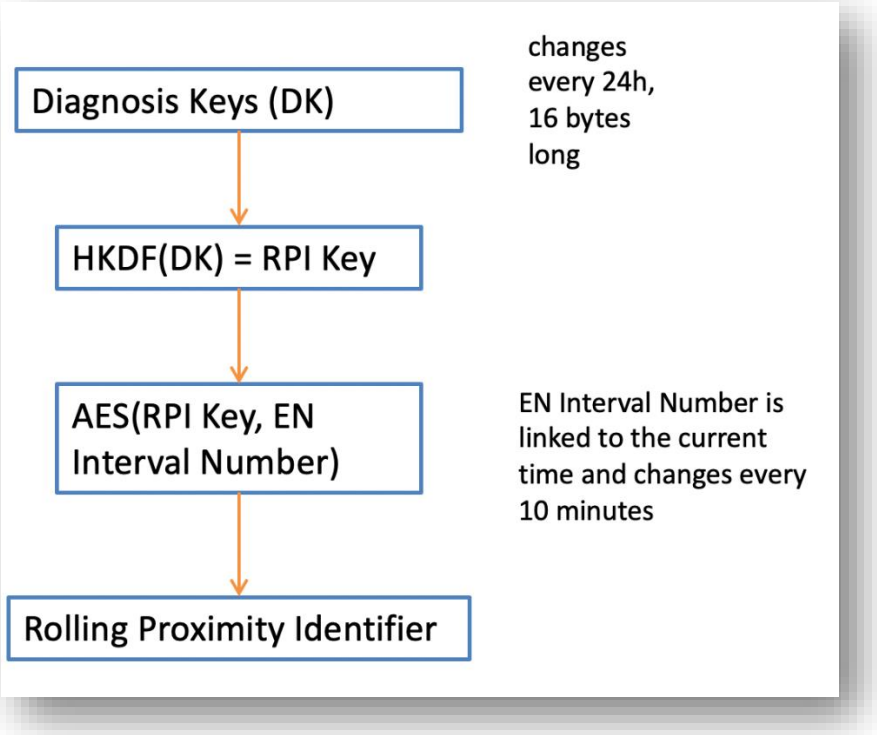


Figure 1: *GAEN framework high-level overview.*

Security Goals

In order to meet the general expectations of a pandemic tracing application that employs the Google Apple Exposure Notification framework, COVID Green and COVID Alert are expected to meet the following security goals:

User Privacy

1. User-identifying information (such as name, phone number, geographical location and IP address) should not be revealed to other users or third parties.
2. User-identifying information, positive test results, and the user symptoms log should not be linkable to one another.
3. User Diagnosis Keys (as generated and employed by the GAEN API) should be confidential from other users.
4. User Diagnosis Keys, as well as Proximity IDs (as generated and employed by the GAEN API) of any two different users should be indistinguishable from one another.

Authentication Between Client and Server

1. Uploading Diagnosis Keys to the Exposure Manager should only be possible by users who hold a positive test result and for whom the test result is pre-authenticated by a medical practitioner.
2. Diagnosis Keys must be unique for every user in order to avoid false positives.
3. Users must be able to authenticate the identity of the Exposure Service (via TLS, pinned certificates or similar) before sending their Diagnosis Keys.

Data Integrity

1. An attacker should not be able to tamper with the Diagnosis Keys delivered to the user or uploaded by the user.
2. An attacker should not be able to tamper with the statistic sent to the CSOStats service.

Data Management

1. A person diagnosed with COVID-19 should upload their Diagnosis Keys to the Exposure Registry every day until they are no longer exhibiting COVID-19 symptoms or are declared cured from COVID-19.
2. User data should be deleted from the Registration Service at the request of the user.
3. User data should not be provided to unauthorized third parties by the Registration Service or any other COVID Green or COVID Alert server component.
4. All data management undergone by the COVID Green or COVID Alert service must adhere to the General Data Protection Regulation (GDPR) as it applies to businesses and corporate entities.[7]

Service Availability

1. A user should be able upload his Diagnosis Keys only once.

[7] https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations_en

2. A user should be able to request only *x* SMS codes or phone calls in *y* hours, where *x* and *y* are pre-determined as sensible options, in order to avoid overloading the COVID Green or COVID Alert service.
3. Only a valid user should be able to request a phone call or SMS confirmation.
4. The Bluetooth connection should be scheduled properly, as most mobile devices may only connect to seven other devices at the same time.

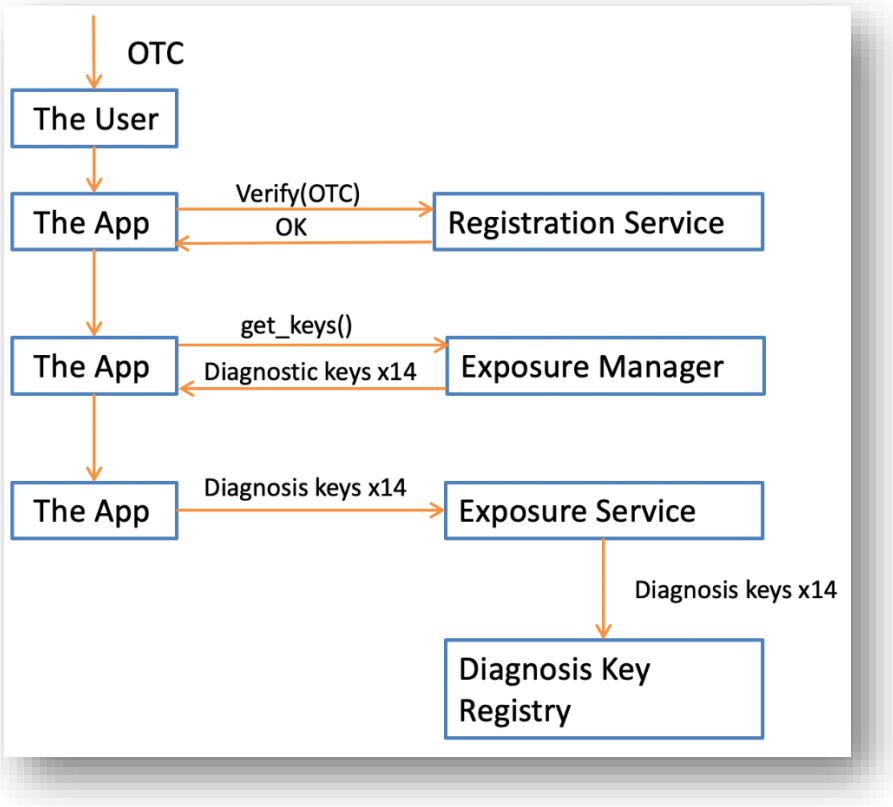


Figure 2: *Positive Test Result Network Flow (Excerpt).*

Threat Model

This section details potential adversaries, attacks and mitigations on the COVID Green and COVID Alert platform.

Partial Application Decomposition

The COVID Green and COVID Alert applications present to the user a set of common use cases, each correlated with a set of external entities. We illustrate these relationships below:

Common Use Cases	External Entities	Attacker Interaction
User Onboarding	Registration Service	Registers as User
Daily Check-In	Check In Service CSO Stats Service	Server Impersonation Client Impersonation Meddler-in-the-middle
Proximity ID Collection	Other Users	Proximity Bluetooth Sensors
Positive Test Result Processing	CCT 1 Call Handler CCT CCT Integration Notification Service Notification Worker **Twilio** Registration Service Exposure Service (Diagnosis Key Registry)	Steal User Device Server Compromise
Exposure Check	Exposure Service	Server Compromise
Exposure Event	Exposure Service Upload Service	Server Compromise
“Leaving the Community”	Registration Service	Registers as User
Sharing the Application	Social Networks (Facebook, Twitter...)	Phishing, Misinformation

From the client’s point of view, the **Trusted Parties** implicated here are:

1. **CCT1CallHandler:** Verifies phone numbers, checks which users are tested positive, symptoms onset date.
2. **Notification Service, Notification Worker.**
3. **Twilio:** Third-party SMS service situated outside of Europe.
4. **HSE:** Holds user identity data and test results.

Attacker Behavioral Summary

An attacker may exploit links between the user's identity information and the following metrics: location, social graph, symptom status, phone number, User Diagnosis Keys, Proximity IDs, and device fingerprinting information (eg. Panopticlick[8]).

1. The attacker is assumed to be **interested in Diagnosis Keys**: if the attacker collects data using Bluetooth proximity sensors (which can reach greatly expanded ranges compared to smartphone devices if paired with high-power antennas) and is able to learn Diagnosis Key data, the attacker may then be able to obtain some information about the location of these users.
2. The attacker may also be **interested in modifying the list of Diagnosis Keys** so that an Exposure Notification appears on the phone of a specific user.
3. The attacker can **aim to disrupt the network** by flooding the registry with fake Diagnosis Keys. If left unmitigated, this could cause the client applications to stop working completely across all client devices.
4. The attacker could **also aim to modify the Diagnosis Keys sent to users** such that exposed citizens are not notified that they could keep transmitting infections.
5. The attacker could furthermore be **interested in the information on *who* infected a specific user**.

Application Threats

An attacker registered as a user or as multiple users may:

1. Upload fake statistics to the CSO Stats Service.
2. Produce fake exposure notifications by copying a Diagnosis Key of an infected person and then deriving their proximity key from it. All other clients who have registered close contact with the attacker will receive an exposure notification.
3. If the user receives an exposure notification, they can then use the time of contact in the Exposure Detection Summary in order to narrow down the source of the infection by correlating reporting time periods.

An attacker eavesdropping on the communications between the user and external entities on the network may:

1. Deduce that the owner of the device is infected when the user uploads their Diagnosis Key.
2. Intercept the symptoms log, sent by the user every day.

An attacker that is impersonating the Exposure Service may:

1. Send fake Diagnosis Keys to users, such that nobody receives exposure notifications.

[8] https://panopticlick.eff.org/

2. Send many Diagnosis Identifiers such that user devices perform enormous amounts of useless calculations, impacting the user's device and potentially causing a distributed denial of service.

An attacker that is impersonating the CSO Stats Service may:

1. Transmit fake news about the pandemic to all users.
2. Collect statistical data.
3. Correlate a symptom log to a device's fingerprint.

An attacker with the ability to set up an array of Bluetooth proximity sensors with far-reaching antennas may:

1. Harass select users by locating a Bluetooth device at their workplace, record its proximity ID on their device, and then impersonating the Exposure Service and transmitting a corresponding Diagnosis Key as an infected person's.
2. Disrupt the contact tracing functionality in a specific area by broadcasting more Proximity IDs than other devices (which are limited to seven Bluetooth connections at a time).
3. Collect the Proximity IDs of a person that is showing symptoms and broadcasting them using an antenna with a broad range, thereby creating false positives.

Application Mitigations

Due to the nature of the Google Apple Exposure Notification service and to the restrictions of decentralized contact tracing, it is unclear whether much of the threats outlined above may be mitigated, and most are likely to be accepted as reasonable limitations under the chosen contact tracing application model.

Some high-level mitigations that are easy to apply include:

1. Diagnosis Key registries may be reasonably limited in terms of the accepted number of key submissions in order to prevent denial of service attacks such as those described above on user devices (see **CVG-01-001**).
2. When Diagnosis Keys are being resolved against local Proximity IDs, the contacts that occurred *after* the time period in which a specific Diagnosis Key was added to the registry should be ignored (see **CVG-01-002**).

Furthermore, providing uniquely the phone number of COVID-positive devices to third-party services, such as Twilio, should be avoided.

The mitigation strategies documented above will be discussed in further detail in the *Audit Findings* sections contained in this report.

Application: Audit Findings

The following sections list three vulnerabilities encountered during the audit. Each vulnerability is given a unique identifier (e.g. **CVG-01-001**) for the purpose of facilitating any future follow-up correspondence.

CVG-01-001 COVID Green: DoS through Diagnosis Key Flooding

Note: *This issue was reported to the client and was thereafter marked as resolved due to the client's assurance of server-side validations and checks.*

A malicious, compromised or misled COVID Green server infrastructure may send an inordinate number of Diagnosis Keys to users, causing a Denial of Service attack on all COVID Green client devices.

Similarly, it is unclear whether the COVID Green client effectively restricts users from uploading more than 14 Diagnosis Keys when submitting a COVID Diagnosis after receiving a Twilio confirmation code. A user could choose to upload an inordinate number of Diagnosis Keys to the server, facilitating a Denial of Service attack on other clients and, to a lesser degree, potentially on the server itself.

Affected Code:

```
Client has confirmed existence of server-side checks, for:
   1. Rejection of future keys (addresses CVG-01-002).
   2. Number of keys limited to 15.
   3. Key structure verified (keys must be 16 bytes).
   4. Deduplication of submitted Diagnosis Keys.
   5. Upload requires valid one-time code.
```

It is recommended to restrict the number of maximum Diagnosis Keys processed by a client device to, for example, thirty per minute, and to enforce that clients only upload the expected number of Diagnosis Keys as an accompaniment to a diagnosis that is confirmed by a medical practitioner.

CVG-01-002 COVID Green: Potential Diagnosis Keys Reuse

Note: *This issue was reported to the client and was thereafter remedied via server-side checks, as discussed in* ***CVG-01-001****.*

When Diagnosis Keys are being resolved against local Proximity IDs, the contacts that occurred *after* the time period in which a specific Diagnosis Key was added to the registry should be ignored. It was observed that the COVID Green application did not appear to contain checks meant to guard against a potential scenario where such Diagnosis Keys are not ignored, thereby leading to false proximity notifications.

Affected Code:
`C.f. CVG-01-001.`

It is recommended to implement specific checks tracking and enforcing the correct chronology with regards to the client-side management of Diagnosis Keys.

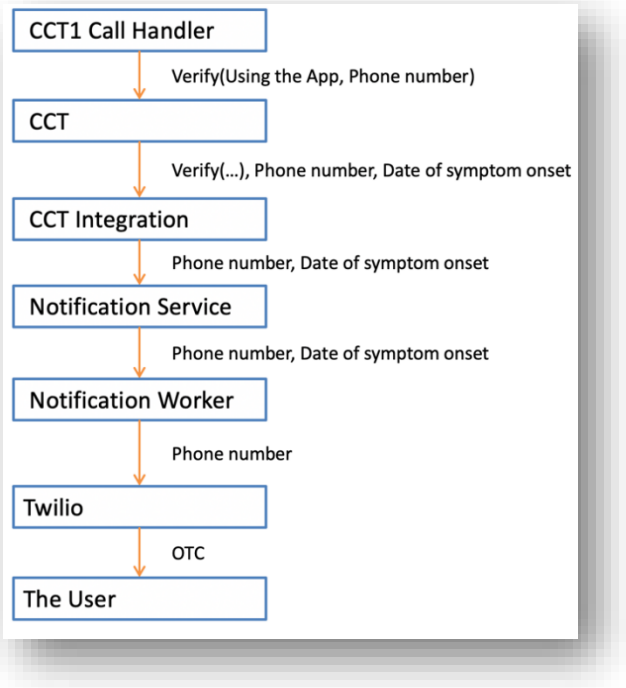


Figure 3: *Network call hierarchy for positive test results.*

CVG-01-003 COVID Alert: Misleading Security Guarantees

Upon initial installation and usage, the COVID Alert application presents a series of introductory screens demonstrating the application's functionality to users. The second of these screens informs the user that their *"privacy is protected"* due to COVID Alert not using GPS or otherwise tracking the user's location. COVID Alert claims that this makes it such that the COVID Alert service has no way of knowing the user's location, name or address.

However, it is not the case that the COVID Alert application can satisfy strong anonymity with regards to user location, name or address. Many functionalities in the COVID Alert application (and virtually all contact-tracing applications) require the client device to make some form of API request, frequently over HTTPS. Such requests (as exemplified in the *Affected Code* portion) immediately reveal to the COVID Alert service the user's IP address, thereby allowing for immediate identification of the user's location and subsequent potential identification of their name and home address via contact with the Internet Service Provider (ISP).

Affected Code (Example):

```
// BackendService/BackendService.ts, line 46
async retrieveDiagnosisKeys(period: number) {
    const periodStr = `${period > 0 ? period : LAST_14_DAYS_PERIOD}`;
    const message = `${MCC_CODE}:${periodStr}:${Math.floor(getMillisSinceUTCEpoch()
/ 1000 / 3600)}`;
    const hmac = hmac256(message, encHex.parse(this.hmacKey)).toString(encHex);
    const url = `${this.retrieveUrl}/retrieve/${MCC_CODE}/${periodStr}/${hmac}`;
    return downloadDiagnosisKeysFile(url);
}
```

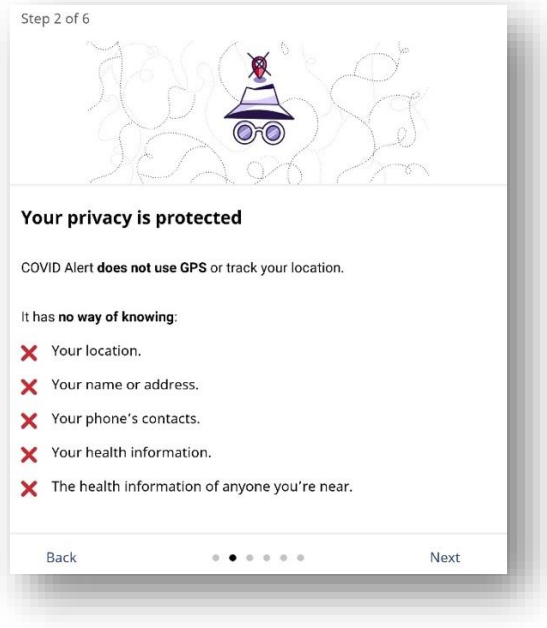


***Figure 4**: COVID Alert claims the above security guarantees when first installed on a user's smartphone device.*

Generally, this issue is recognized by the designers of the GAEN and the underlying DP-3T contact tracing protocol and is generally regarded as a future issue to resolve with the usage of mix-nets, VPNs, onion routers or other similar technology.[9]

It is recommended to rewrite this dialog in order to remove misleading security promises being displayed to COVID Alert users, namely that COVID Alert does not allow for COVID Alert servers to know the client's location, name or address.

[9] https://tracing-risks.com/

CVG-01-004 COVID Alert: Diagnosis Key Timestamps Set by Client

Note: *This issue was reported to the client and was thereafter termed as inconsequential due to COVID Alert servers not making full use of Diagnosis Key Timestamps as set by the client.*

It was found that Diagnosis Key reporting logic on the COVID Alert client allowed the client to set the timestamp for the Diagnosis Key reporting event. This could allow clients to maliciously provide incorrect timestamps for exposure key reporting events.

Affected Code:

```
// BackendService/BackendService.ts, line 136
async reportDiagnosisKeys(
    keyPair: SubmissionKeySet,
    _exposureKeys: TemporaryExposureKey[],
    contagiousDateInfo: ContagiousDateInfo,
  ) {
    […]
    const upload = covidshield.Upload.create({
      timestamp: {seconds: Math.floor(getMillisSinceUTCEpoch(),
      keys: exposureKeys.map(key =>
        covidshield.TemporaryExposureKey.create({
          keyData: Buffer.from(key.keyData, 'base64'),
          transmissionRiskLevel: TRANSMISSION_RISK_LEVEL,
          rollingStartIntervalNumber: key.rollingStartIntervalNumber,
          rollingPeriod: key.rollingPeriod,
        }),
      ),
 });
```

It is recommended to set and validate timestamps purely on the server side in order to avoid client-side malleability of exposure key reporting times.

High-Level Design Recommendations

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

CVG-01-005 COVID Green: SMS Provider Obtains Diagnosed Phone Numbers

Note: *This issue was reported to the client. The client's response indicates that Twilio usage was pre-approved by the Irish health authority and the Irish Data Protection Commissioner. Furthermore, COVID Green has since confirmed that Twilio is one of many supported SMS providers, and that environments employing COVID Green are free to set up their own provider.*

It has been observed that whenever a COVID Green user wishes to confirm a COVID-19 diagnosis, they are required to obtain a special one-time code from a medical practitioner, who is expected to use an API call similar to the following in order to trigger the sending of an SMS with the code to the user upon successful medical diagnosis:

Example API Call:

```
curl --location --request POST 'https://push-nfcs-dev.nf-covid-
services.com/notify/positive' \
--header 'Authorization: Bearer XXXXXXXXXXXXXXXXXXXXXXXX \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'symptomDate=2020-02-01' \
--data-urlencode 'mobile=+353875554333'
```

The issue lies with the fact that all patients with a confirmed medical diagnosis will have their phone numbers submitted to Twilio, a private corporation not based in Europe. Since Twilio is only receiving the phone numbers of positive COVID cases, all submitted presumably using the same Twilio API key, Twilio will then be trivially capable of collecting the phone numbers of every COVID-19-diagnosed individual in Ireland who reports using COVID Green. It is furthermore unclear if such a design would comply with several European legal directives, such as the General Data Protection Regulation (GDPR) and especially its provisions on data concerning health, or whether it would meet the intended spirit of the law.[10]

Furthermore, SMS broadcasts in and of themselves are problematic from a privacy standpoint: SMS traffic does not benefit from any privacy-preserving properties from anyone with access to nearby cellular antennas, from any node on the cellular network or from cellular network service providers. SMS traffic can be trivially intercepted and impersonated using commercial devices such as IMSI-catchers.[11]

[10] https://edps.europa.eu/data-protection/our-work/subjects/health_en
[11] https://en.wikipedia.org/wiki/IMSI-catcher

As noted in the Threat Model section of this report, users expect that user-identifying information (such as their phone number) will not be revealed to third parties and that user-identifying information will not be linkable to positive test results. These requirements are not met by this finding. In addition, there are alternative means that could be used that would not violate this requirement (e.g., by having the medical practitioner directly contact the individual with a positive result, or providing a secured diagnosis alert mechanism within the app itself). Therefore this issue is considered to have a significant impact because this approach fails to meet the security requirements listed above and it broadly reveals information about all users with a positive diagnosis, not just a few users.

It is strongly recommended to avoid using the Twilio approach for this, and to devise another approach, such as direct contact between the medical practitioner in order to communicate the submission code via phone call, online videoconferencing, or by email.

Unlike COVID Green, COVID Alert avoids SMS-based notifications and instead relies on in-app notifications and on direct communication with medical practitioners. This approach gives COVID Alert a clear advantage by enabling to side-step SMS entirely. Continued usage of SMS may make COVID Green run afoul of the United States HIPAA Act and similarly constrain deployment scenarios worldwide.

CVG-01-006 COVID Green: Statistical Data Figures Overly Precise

Note: *This issue was reported to the client and was confirmed to be out of scope.*

The COVID Green application provides a set of statistical data meant to inform the general population on the progression of the COVID-19 pandemic in their local region. The information includes the total number of confirmed cases, total deaths, the number hospitalized, the number that required ICU, and the exact number of individual cases per county in Ireland. The data appears to be updated at least once per day.

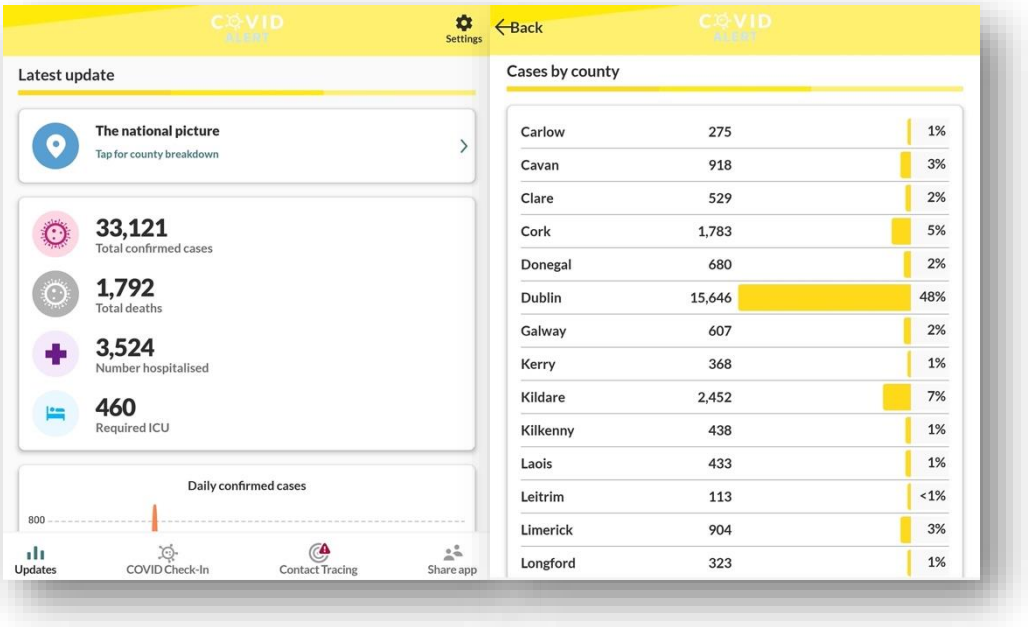


Figure 5*: COVID Green statistics feature, as shown on an Android smartphone.*

The issue lies in that such a precise series of metrics could facilitate the deanonymization or identification of those reporting to be suffering from COVID-19 symptoms, especially in counties with a smaller population.

It is therefore recommended to provide estimates (e.g. “110-120” instead of “113”) when listing figures. This should give those who choose to report their diagnosis greater comfort in the level of plausible deniability they may obtain vis-à-vis their peers. This said, it is unclear if the COVID Green service has any control over the reporting of these statistical figures, and it may very well be that the Irish government, for instance, may continue to provide too-precise statistical data on a per-county basis.

CVG-01-007 COVID Alert: Tests Fail to Generate Random Keys

It was found that the unit tests for the COVID Alert application failed to generate random keys, thereby forcing tests to always default to null keys, which could lead to reduced test coverage for a critical component of the application's functionality.

Affected Code:

```
// BackendService/BackendService.spec.ts, line 87
function generateRandomKeys(numberOfKeys: number) {
  const keys: TemporaryExposureKey[] = [];
  for (let i = 0; i < numberOfKeys; i++) {
    keys.push({
      keyData: '',
      rollingPeriod: i,
      rollingStartIntervalNumber: i,
      transmissionRiskLevel: 0,
    });
  }
  return keys;
}
```

It is recommended to revise the unit test code such that random keys are generated for the purposes of this particular unit test.

Conclusion

Symbolic Software has audited COVID Green and COVID Alert client stacks, a component of the server API calls and has devised a threat model and a set of security goals based on the COVID Green application stack and its employment of the Google Apple Exposure Notification (GAEN) framework.

Overall, issues found in COVID Green were largely restricted to the general reasonable assumptions that come with the threat model of any GAEN-based contact tracing application. **CVG-01-001** and **CVG-01-002** describe some issues that seem specific to COVID Green and that are likely easy to mitigate.

With COVID Alert, a number of minor code-level issues were spotted, of which the only serious issue seemed to deal with incomplete test coverage. However, COVID Alert was also documented as making inaccurate security guarantees and promises, as documented in **CVG-01-003**. Proper communication of effective security guarantees, especially in the novel context of contact tracing protocols and applications, is strongly recommended.

Both applications implemented front-ends to the GAEN framework without compromising the security goals of the underlying contact tracing protocol. As is unfortunately common with contact tracing applications, procedural decisions on top of GAEN can still end up negatively affecting user privacy. As such, **CVG-01-003** and **CVG-01-005** are deemed to be the most pertinent findings of this audit, despite their not being code-level issues.

As a result of this audit, many issues were addressed proactively with the COVID Green and COVID Alert developers, while others have led to discussions regarding the effective security and privacy guarantees of both applications.

Symbolic Software would like to thank Colm Harte, James M. Snell, John O'Brien and Derek Zimmer for their project coordination and general assistance and feedback regarding this audit.

About Symbolic Software

Since its founding in 2017, Symbolic Software has participated in over 300 cryptographic audits, working on projects as diverse as Mozilla Thunderbird, 1Password, NordVPN, MetaMask, NYM and more, auditing password managers, VPNs, cryptocurrencies, secure communications systems and other high-stakes cryptographic deployments. Symbolic Software has also published Verifpal, a cryptographic protocol analysis framework used by Zoom, and Noise Explorer, a cryptographic protocol analysis and implementation framework. Symbolic Software is directed by Dr. Nadim Kobeissi and employs talented researchers in France and around the world.