On the Effectiveness of Time Travel to Inject COVID-19 Alerts[⋆]

Vincenzo Iovino[1], Serge Vaudenay[2], and Martin Vuagnoux[3]

[1] University of Salerno, Italy
[2] EPFL, Lausanne, Switzerland
[3] base23, Geneva, Switzerland

Abstract. Digital contact tracing apps allow to alert people who have been in contact with people who may be contagious. The Apple/Google Exposure Notification (EN) system is based on Bluetooth proximity estimation. It has been adopted by many countries around the world. However, many possible attacks are known. The goal of some of them is to inject a false alert on someone else's phone. This way, an adversary can eliminate a competitor in a sport event or a business in general. Political parties can also prevent people from voting.
In this report, we review several methods to inject false alerts. One of them requires to corrupt the clock of the smartphone of the victim. For that, we build a time-traveling machine to be able to remotely set up the clock on a smartphone and experiment our attack. We show how easy this can be done. We successfully tested several smartphones with either the Swiss or the Italian app (SwissCovid or Immuni). We confirms is also works on other EN-based apps: NHS COVID-19 (in England and Wales), Corona-Warn-App (in Germany), and Coronalert (Belgium).

1 Introduction

Apple and Google deployed together the Exposure Notification (EN or GAEN) system as a tool to fight the pandemic [1]. The goal of an EN-based app is to alert people who have been in close proximity for long enough with someone who was positively tested with COVID-19 and who volunteered to report. How a user responds to such alert is up to the user, but one would expect that such user would contact authorities and be put in quarantine for a few days. In Switzerland, the alerted user is eligible to have a free COVID-19 test but the result of the test would not change his quarantine status.

EN is provided by default in all recent Android or iOS smartphones which are equipped with Bluetooth (except Chinese ones due to US regulation). It is installed without the consent of the user. However, it remains inactive until the user activates it (and possibly install an app which depends on the region).

Once activated, EN works silently. A user who is tested positive with COVID-19 is expected to contribute by reporting through EN. This may have the consequence of triggering an alert on the phones of the EN users whom the COVID-positive user met.

Assuming an alerted user is likely to self-quarantine, and possibly make a test and wait for the result, this alerted user may interrupt his activities for a few days. A malicious adversary could take advantage of making some phones raise an alert. In a sport competition (or any other competition), an alerted competitor would stay away for some time. Malicious false alert injections could be done at scale to disrupt the activities of a company or an organization. This could be done to deter people from voting.

False injection attacks have been well identified for long [13,14]. It was sometimes called the *lazy student attack* where a lazy student was trying to escape from an exam by putting people in quarantine. Nevertheless, EN was deployed without addressing those attacks.

[⋆] Videos are available on `https://vimeo.com/477605525` (teaser) and `https://vimeo.com/476901083`.

In most of cases, those attacks require to exploit a backdoor in the system, or to corrupt the health authority infrastructure, or to corrupt a diagnosed user. Our goal is to show how easily and inexpensively we can make an attack which requires no such corruption.

Our contribution. In this paper, we analyze possible false alert injection attacks. We focus on one which requires to corrupt the clock of the victim and to literally make it travel through time. By doing so, we can replay Bluetooth identifiers which have just been publicly reported but that the victim did not see yet. We replay them by making the victim go to the time corresponding to the replayed identifier then coming back to present time. We show several ways to make a smartphone travel through time and to make it receive an alert when it comes back to present time.

In the easiest setting, we assume that the victim and the adversary are connected to the same Wi-Fi network Essentially, the network tells the current time to the phone. We report on our successful experiments.

In Section 6.1 we describe the equipment we used in the experiments: a Raspberry Pi Zero W (Fig. 1) and a home-assembled device endowed with an ESP32 chipset (Fig. 2, both available on the market for about 10$. It takes less than a minute to run the attack. In favorable cases (specifically, with the variant using a rogue base station), the attack duration can be reduced to one second. The attacks possibly works on all EN-based systems. We mostly tested it on the Swiss and Italian systems (SwissCovid and Immuni). We also verified on other apps. We conclude that such attacks are serious threats to society.

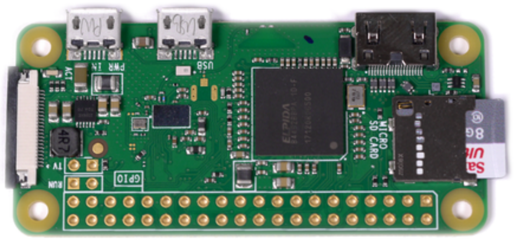


Fig. 1. Raspberry Pi Zero W

Our attacks experimentally confirm the evidence that EN offers no protection even against (traditional) replay attacks.

Our technique can be also used to debug the notification mechanism of EN without directly involving infected individuals; this is a step forward in disclosing the EN's internals since EN is closed source and not even debuggable. (Precisely, to experiment with the EN system, you need a special authorization.)

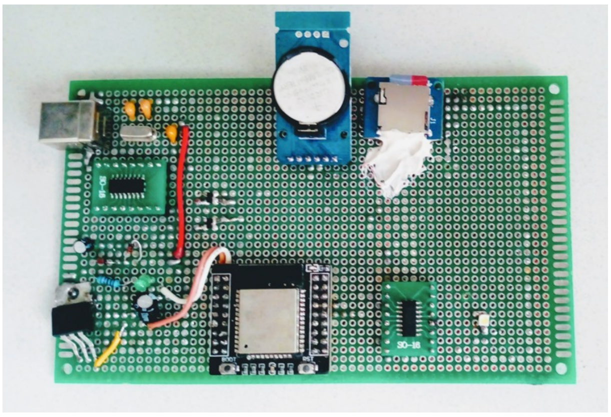


Fig. 2. Our ESP32-based device

Disclaimer. We did a responsible disclosure. We first reported and discussed the attack with the Italian Team of Immuni on September 24, 2020.[4] Few days after we received an answer from an account administrated by the team stating:

> *"thank you for reporting this replay attack. Unfortunately we believe that this is an attack against GAEN rather than Immuni and so it should be resolved by a protocol implementation update. Should you have suggestions for our own code base to prevent or mitigate the attack, please let us know and we will evaluate them."*

We reported the attack in Switzerland on October 5, 2020.[5] We received an acknowledgement on October 10 stating:

> *"The NCSC considers the risk in this case as acceptable. The risk assessment must also take into account whether there is a benefit and a ROI for someone who takes advantage of it. Especially since an attacker typically must be on site."*

We also reported a detailed attack scenario to Google on October 8.[6] We received the following response:

> *"At first glance, this might not be severe enough to qualify for a reward, though the panel will take a look at the next meeting and we'll update you once we've got more information."*

The attack was also mentionned in the Swiss press and in an official document by Italian authorities. In *24 Heures* on October 8[7], the representative of EPFL declared that the described attack is technically possible but would require too much resources and efforts.

The Italian "Garante della Privacy" (the national data protection officer)[8] commented that replay attacks with the purpose of generating fake notifications do not represent a serious

[4] `https://github.com/immuni-app/immuni-app-android/issues/278`.
[5] Registered incident INR 8418 by the National Cyber Security Center (NCSC) `https://www.melani.admin.ch/dam/melani/de/dokumente/2020/SwissCovid_Public_Security_Test_Current_Findings.pdf.download.pdf/SwissCovid_Public_Security_Test_Current_Findings.pdf`
[6] Reference 170394116 for component 310426.
[7] `https://www.24heures.ch/les-quatre-failles-qui-continuent-de-miner-swisscovid-348144831017`
[8] `https://www.garanteprivacy.it/home/docweb/-/docweb-display/docweb/9468919`.

vulnerability since they require the attacker to take possession of the victim's phone. Since both traditional replay attacks and the variants of replay attacks we show in this paper can be performed without taking possession of the victim's phone, we contacted the aforementioned Italian authorities to provide clarifications about replay attacks and to ask whether they are aware of the fact that replay attacks can be performed without taking possession of the victim's phone but at time of writing we did not receive any answer.

2 How EN Works

In short, EN selects every day a random key called TEK (as for *Temporary Exposure Key*). Given the daily TEK, it deterministically derives some ephemeral keys called RPI (as for *Rolling Proximity Identifier*). Each RPI is emitted over Bluetooth several times per second during several minutes. Additionally, EN scans Bluetooth signals every 3–5 minutes and stores all received RPI coming from other phones. If the user is diagnosed, the local health authorities provide an access code (which is called a *covidcode* in Switzerland). This is a one-time access code which is valid for 24 hours which can be used to *report*. If EN is instructed to report, it releases every TEK which were used in the last few days which the user allows to publish. At this point, a TEK is called a *diagnosis key*. The report and access code are sent to a server which publishes the diagnosis keys. Once a while, EN is also provided with the published diagnosis keys on the server. EN re-derives the RPI from those diagnosis keys and compares with the stored RPI of encounters. Depending on how many are in common, an alert is raised.

On Fig. 3, we have three users with their smartphones: Alice, Bob, and Charly. Bob meets the two others but Alice and Charly do not meet each other. They exchange their RPI. After a while, Alice gets positive and receives a covidcode. She publishes her TEK using her covidcode. Other participants see the diagnosis key from Alice. They compare the derived RPI with what they have received. Only Bob finds a match and raises an alert.

More precisely, we set

$$\mathsf{RPI} = f(\mathsf{TEK}, t)$$

where t is the time when RPI is used for the first time and f is a cryptographic function based on AES [1]. In EN, time is encoded with a 10-minute precision. Actually, the value of t is just incremented from one RPI to the next one, starting from the time when TEK is used for the first time. There is also an Associated Encrypted Metadata (AEM) which is derived by

$$\mathsf{AEM} = g(\mathsf{TEK}, \mathsf{RPI}) \oplus \mathsf{metadata}$$

where $\oplus$ denotes the bitwise exclusive OR operation, $\mathsf{metadata}$ encodes the power π used by the sender to emit the Bluetooth signal, and g is a similar cryptographic function.

We list below a few important details.

- What is sent over Bluetooth is the pair $(\mathsf{RPI}, \mathsf{AEM})$.
- Received $(\mathsf{RPI}, \mathsf{AEM})$ pairs are stored with the time of reception t and the power p of reception.
- What is published on the server are pairs (TEK, τ) where τ is the time when TEK was used for the first time. (See Fig. 4.)
- New (TEK, τ) pairs are posted on the server with a date of release (not shown on the picture). Since they are posted when the user report, the posting date can be quite different from τ. This posting date is used to retreive only newly uploaded pairs. Hence, the downloaded τ do not come in order.

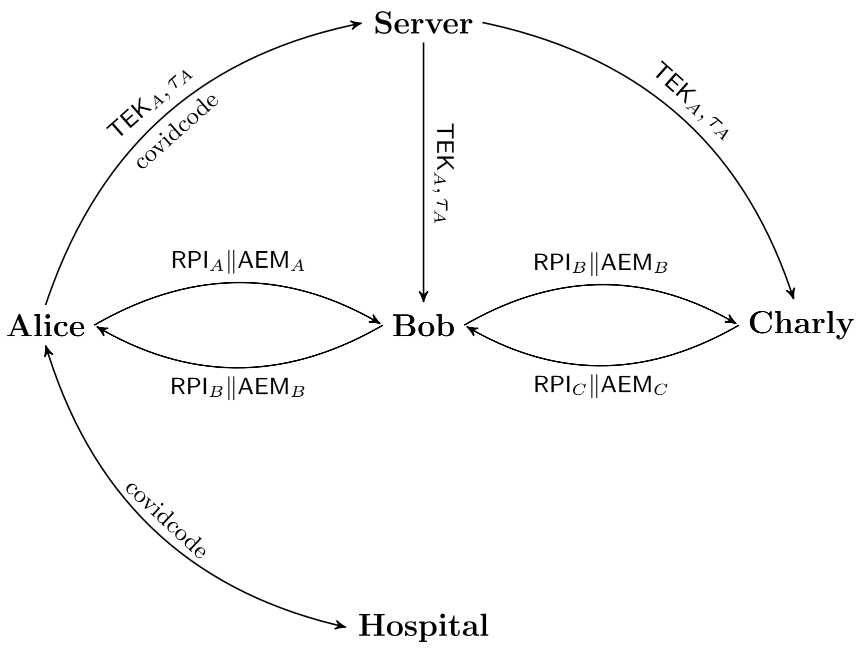

Server
TEK_A, τ_A
covidcode
TEK_A, τ_A
TEK_A, τ_A
$\mathsf{RPI}_A \| \mathsf{AEM}_A$
$\mathsf{RPI}_B \| \mathsf{AEM}_B$
Alice
Bob
Charly
$\mathsf{RPI}_B \| \mathsf{AEM}_B$
$\mathsf{RPI}_C \| \mathsf{AEM}_C$
covidcode
Hospital

Fig. 3. Exposure Notification Infrastructure

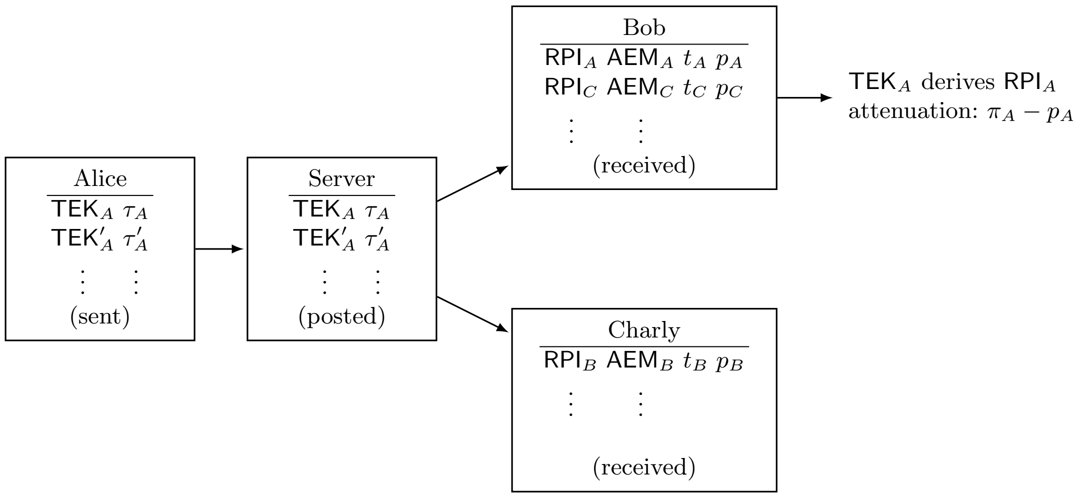

Alice
$\mathsf{TEK}_A\ \tau_A$
$\mathsf{TEK}'_A\ \tau'_A$
(sent)
Server
$\mathsf{TEK}_A\ \tau_A$
$\mathsf{TEK}'_A\ \tau'_A$
(posted)
Bob
$\mathsf{RPI}_A\ \mathsf{AEM}_A\ t_A\ p_A$
$\mathsf{RPI}_C\ \mathsf{AEM}_C\ t_C\ p_C$
(received)
TEK_A derives RPI_A
attenuation: $\pi_A - p_A$
Charly
$\mathsf{RPI}_B\ \mathsf{AEM}_B\ t_B\ p_B$
(received)

Fig. 4. Matching TEK from Server to Captured RPI

- When EN gets the downloaded diagnosis key TEK and derives the RPI, the time is compared with what is stored with a tolerance of ± 2 hours. If it matches, the receiver can decrypt AEM to recover the metadata, deduce the sending power π, then compare with the receiving power p to deduce the signal attenuation $\pi - p$.
- For Switzerland, the attenuation is compared with two thresholds. t_1 and t_2. If larger than t_2, it is considered as too far and ignored. If between t_1 and t_2, the duration is divided by two to account that the distance is not so close. If lower than t_1, the encounter is considered as very close and the duration is fully counted. SwissCovid was launched on June 25, 2020 and the sensibility of parameters has been increased twice. Since September 11, 2020, the parameters are $t_1 = 55$ dB, $t_2 = 63$ dB [2].

  With those parameters, in lab experimental settings [2], the probability to catch an encounter at various distances is as follows:

distance of encounter	1.5 m	2 m	3 m
$\Pr[\text{attenuation} < t_1]$	57.3%	51.6%	45.6%
$\Pr[\text{attenuation} < t_2]$	89.6%	87.5%	84.2%

- For Italy, only one threshold of 73 dB is used [11]. The same lab experiment as above indicate probabilities of 100%. Since July 9, the unique threshold was changed to 63 dB.
- Every scan which spotted an encounter counts for the rounded number of minutes since the last scan with a maximum of 5 minutes (possibly divided by two as indicated above). The total sum is returned and compared to a threshold of 15 minutes.

3 Techniques for False Alert Injection

We list here several strategies to inject false alerts.

Injection with Real Encounters. The adversary encounters the victim normally and the victim records the sent RPI. If the adversary manages to fill a false report with his TEK, this will cause an alert for the victim. Filling a false report can done

- either due to a bug in the system
- or by corrupting the health authority system
- or by corrupting a user who received the credentials to report.

Switzerland corrected one bug: the ability for the reporter to set verification algorithm to "`none`" in the query, instructing the server not to verify credentials. This is actually a commonly known attack on implementations using JWT (JSON Web Tokens) which is based on a dangerous default configuration.

A corruption system was fully detailed and analyzed by Avitabile, Friolo, and Visconti [9]. Either positive people who receive a covidcode could sell it to buyers who would want to run the attack, or just-tested positive people could be paid for reporting a TEK provided by an attacker. The system could be made in such a way that buyers and sellers would never meet, their anonymity would be preserved, and their transaction would be secured. The infrastructure for this black business would collect a percentage on the payment. It would run with smart contracts and cryptocurrencies.

Injection with Simulated EN. We assume that the adversary uses a device (for instance, a laptop computer with a Bluetooth dongle) which mimics the behavior of the EN system. One difference is that this device sends Bluetooth signal with high power but announces them with a low power in AEM. This way, he can send the signal from far away and the computed attenuation will be low, like in a close proximity case. The victim can receive an RPI which is sent from the device and believe in a proximity. Reporting the TEK could be done like for the real encounter attack.

A second advantage of this attack is that the sending device could be synchronized in its simulation with several other devices. This could be used by a group of adversaries (terrorists, activists, gang) to inject false alerts in many victims. All members of the group would be considered as a single person by the EN system and all their encounters would receive the same keys. In this case, it could also make sense to have one member of the group (a kamikaze) to genuinely become positive and report.

Injection with Replay Attack. Another false encounter injection attack consists of replaying the RPI of someone else. Due to the EN infrastructure, the RPI is valid for about two hours. One strategy consists of capturing the RPI of people who are likely to be reported soon. It could be people going to a test center, or people who are known to have symptoms but who did not get their test results yet. Capturing their RPI can be done from far away with a good Bluetooth receiver. The malleability of the metadata in AEM can also be exploited to decrease the announced sending power [10,15].

Injection with Belated Replay Attack. Another form of replay attack consists of replaying the RPI which are derived from the publicly posted diagnosis keys. Because GAEN only tries to match new diagnosis keys, the adversary can try with recently updated TEK which have not been downloaded yet by the victim. This is doable since the app checks only a few times for newly uploaded keys during the day. Those keys are however outdated and would normally be discarded when EN compares RPI with the ones derived from the diagnosis keys. However, we could send the phone of the victim in the past then send the outdated RPI to the phone. When the phone would be brought back to present, it would eventually raise an alert. Sending a phone in the past requires no time travel machine. It suffices to corrupt its internal clock.

4 Time-Traveling Phones

Several techniques to corrupt the date and time of a smartphone have been identified (see Park et al. [12] for detailed information). In this section, we describe four of them. Modern smartphone operating systems use at boot time NTP (Network Time Protocol) if network access is provided. NITZ (Network Identity and Time Zone) may be optionally broadcasted by mobile operators. GNSS (Global Navigation Satellite System) such as GPS can also be used. Finally, the clock can be manually set. The priority of these clock sources depends on the smartphone vendors. Some of them can be disabled by default, but in general, the priority is MANUAL > NITZ > NTP > GNSS.

4.1 Set Clock Manually

This is technically the easiest attack, but it requires a physical access to the smartphone.[9] An adversary picks a newly published TEK and computes one RPI for a date and time in the past. The adversary physically accesses the smartphone and sets the corresponding clock. Then, he replays the RPI for 15 minutes using a Bluetooth device such as a smartphone or a laptop. Finally, the adversary sets the time back to present. As soon as the smartphone updates the new TEK list, an alert is raised.

A single RPI is generally not supposed to be repeated during 15 minutes as its rotation time is shorter (typically: 10 minutes). It seems that implementations do not care if it is the case. However, we can also use two consecutive RPI from the same TEK and repeat them for their natural duration time.

Observe that in some circumstances the purpose of the attacker can be to send a fake notification to his own phone, in which case the assumption that the adversary has physical access to the phone of the victim makes perfect sense. This self-injection attack can be done to scare friends and family members, to get the permission of staying home from work, or to get priority for the COVID-19 test.[10] Furthermore, in the case of the Italian app Immuni, each risk notification is communicated to the Italian Ministry of Health: the Italian authorities keep a counter on how many risk notifications have been sent to Immuni's users.[11] Therefore, sending fake notifications even to phones controlled by the adversary represents a serious attack in itself since it allows the attacker to manipulate the official counter arbitrarily.

The Italian's counter of risk notifications only takes into account notifications sent from phones endowed with the "hardware attestation" technology, a service offered by Google. So, our attacks show a way to bypass this trusted computing mechanism to manipulate the official counters.

4.2 Rogue NTP server

If the smartphone is connected on the Internet on Wi-Fi, it may use NTP (Network Time Protocol) to synchronize clock information. If the adversary connects to the same Wi-Fi network of the victim, he may set an ARP-spoofing attack to redirect all NTP queries to a rogue server. Since NTP authentication is optional, the response from the rogue NTP server will be accepted and then, the adversary can remotely set the date and time of the smartphone. If the adversary owns the Wi-Fi network (what we call a rogue Wi-Fi network), the attack requires no ARP-spoofing. Instead, the adversary sets up an NTP server and controls time.

If the mobile network has a priority over NTP, we may assume that the victim is not connected to the mobile network. Otherwise, the adversary may have to jam it.

Depending on the smartphone vendors, NTP is used permanently or only at the boot time, then every 24 hours. Sometimes, third-party apps force constant NTP synchronization. The adversary may wait until an NTP request is sent to trigger the whole attack. Otherwise, the adversary must make the victim's smartphone reboot. Making a target to reboot can

[9] It can be done without this assumption by using a vulnerability of the phone allowing to execute a code remotely.

[10] Indeed, in Switzerland a risk notification has legal value in the sense that it gives priority for the COVID-19 test, a free test, and also subsidies when the employer does not give a salary to stay home without being sick.

[11] `https://www.immuni.italia.it/dashboard.html`

be done by social engineering (by convincing the victim to reboot). Another way is to use a Denial-of-Service attack. With DoS, the adversary can remotely reboot a smartphone.

4.3 Rogue Base Station

NITZ message are sent by mobile operator to synchronize time and date when a smartphone is connected to a new mobile network. This is generally used to set new time zone when roaming on another country.

Since the adversary must be physically close to the victim to broadcast replayed RPI, he may also set up a rogue mobile network base station to send corrupt NITZ message. Thus, when the victim is connected to the rogue mobile network, the date and time is modified. Compared to previous techniques, the adversary can now modify clock information at any time by disconnecting and reconnecting the smartphone at will. Note that since the adversary also controls mobile data, he may block update or NTP Requests to avoid potential issues.

Making sure that the victim connects to the rogue base station may require to jam the signal of the one it uses and to impersonate the network it subscribed to. Since there is no authentication of the base station, this is easily done.

4.4 Rogue GNSS

The last technique is to send a fake GNSS signal to modify internal clock of smartphones. Open source tools are available to generate GPS signals. This attack is less practical, since smartphones may not accept GNSS as trusted source clock by default. Moreover, NITZ and NTP take precedence over GNSS.

5 Master of Time

A limitation of the attacks described above is the need to stay for at least 15 minutes close to the victim to replay RPI. However, EN is not continuously scanning Bluetooth broadcast (because of power consumption). Indeed, only 5 seconds of scan is performed regularly by the smartphone.[12] The duration between two scans is random but typically in the 3–5 minutes range. Since the adversary is able to modify the date and time at any time, we define an improvement, called *Master of Time*, to accelerate the alert injection process.

The goal of the improvement is to trigger the 5-seconds scans more quickly. The adversary first goes in the past to the corresponding time and date of the replayed RPI. This generally triggers a 5-second scan. After that, a random delay is selected by the phone. However, the adversary updates the time again, but to 5 minutes later. The phone realizes it missed to scan and this triggers an immediate opportunistic scan. After this new 5-seconds scan, the adversary updates the time to 5 minutes later again and a third scan is launched. Finally in 15 seconds, the adversary can trigger enough 5-seconds scans to simulate an exposure of more than 15 minutes. This improvement can be applied to all the techniques described above.

There is actually no need to wait for the entire duration of a scan. We can actually reduce the duration between time jumps to 200ms but we should also increase the frequency of sending RPI. Hence, we broadcast over Bluetooth every 30ms. Since the duration of a time jump using a rogue base station takes 100ms, the entire attack takes less than one second in total.

[12] Technically, the scan listens for 4 seconds but an extra second may be needed to activate the scan.

6 Experiments

In this Section, we give a detailed description of the Rogue NTP Server Attack and the Rogue Base Station Attack with threat model, experimental setup and results. We tested all options of the attack we mention. We should stress that attacks are not always stable (our success rate is at least 80%) but failure cases are often due to bugs in the phone (in the app, in GAEN, or in the operating system). We found workaround to incease the reliability. However, this technology is a living matter and so are the workarounds we found.

6.1 Rogue NTP server

We list here a few assumptions.

- The adversary must be within the Bluetooth range of the victim.[13]
- The adversary must access to the same Wi-Fi network of the smartphone and redirect data traffic (by using ARP spoofing attack or rogue Wi-Fi network).
- If NITZ has a priority over NTP, we assume that the victim is not connected to the mobile network. Otherwise, the adversary may have to jam it.
- Depending on the smartphone model, the adversary may need to force NTP as explained in Section 4.2.
- We also assume that the victim has not pulled yet the last updated TEK-list which is used by the adversary. (Otherwise, the victim will not try to match it and the attack fails.)

The hardware needed for this attack is relatively simple. Only Wi-Fi and Bluetooth are needed. We tested several smartphones (Motorola z2 force, Samsung Galaxy S6, Samsung Galaxy A5). We used a Raspberry Pi Zero W (Fig. 1) to host a rogue NTP server. We use a custom Python-based NTP server to deliver the date and time retrieved from the selected TEK.

We also tested using a different hardware platform. We used an home-assembled device endowed with an ESP32 chipset available on Amazon for about 10 Euros. Full fledged devices with the ESP32 chipset are available in different sizes, for instance in watches[14], and as such are easily concealable by an attacker.

In rogue Wi-Fi settings, we set up the device as a rogue Wi-Fi access point. Then, we redirect all UDP connections on port 123 to the rogue NTP server (hosted on the same device). We also configure the access point to block TEK-list queries by the smartphone to avoid potential issues. When using a genuine Wi-Fi network, we use ARP spoofing to redirect to the rogue NTP server and also to block Internet access to the victim's phone so as to prevent the download of the TEK-list during the critical phases of the attack.

The attack then works as follows.

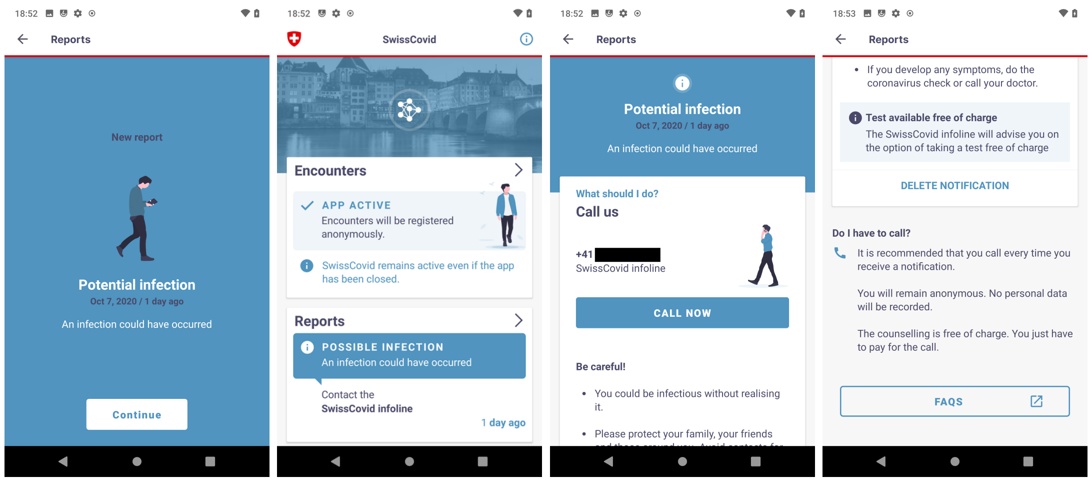
1. The adversary retrieves the updated TEK-list (which is publicly available) from the official server.
2. He picks a new TEK and derives an RPI and AEM. The emission power in AEM is set to low to improve the chances for the RPI to be accepted.

[13] This range could be enlarged using a 2.4GHz amplifier.

[14] `https://www.banggood.com/it/LILYGO-TTGO-T-Watch-2020-ESP32-Main-Chip-1_54-Inch-Touch-Display-Programmable-Wearable-Environmental-Interaction-Watch-p-1671427.html`.

3. The adversary waits for an NTP request from the smartphone[15]. He replies to NTP requests with the date and time of the RPI (i.e. in the past).
4. He sends for at least 15 minutes the RPI||AEM using the Raspberry Pi Zero W and Bluez tools.
5. The active part of the attack can stop here. The adversary can wait for the smartphone to restore the date and time by itself, then update the TEK-list and raise an alert. Alternately, the adversary can set the clock back to normal then wait for the TEK-list update.

We tested all variants of the attacks described above with success. Sometimes the smartphone has issues to update the TEK-list and it may need up to 12 hours to trigger the alert.


18:52

Reports

New report

Potential infection

Oct 7, 2020 / 1 day ago

An infection could have occurred

Continue

18:52

SwissCovid

Encounters

APP ACTIVE

Encounters will be registered anonymously.

SwissCovid remains active even if the app has been closed.

Reports

POSSIBLE INFECTION

An infection could have occurred

Contact the SwissCovid infoline

1 day ago

18:52

Reports

Potential infection

Oct 7, 2020 / 1 day ago

An infection could have occurred

What should I do?

Call us

+41 [redacted]

SwissCovid infoline

CALL NOW

Be careful!

- You could be infectious without realising it.
- Please protect your family, your friends

18:53

Reports

- If you develop any symptoms, do the coronavirus check or call your doctor.

Test available free of charge

The SwissCovid infoline will advise you on the option of taking a test free of charge

DELETE NOTIFICATION

Do I have to call?

It is recommended that you call every time you receive a notification.

You will remain anonymous. No personal data will be recorded.

The counselling is free of charge. You just have to pay for the call.

FAQS

Fig. 5. Screen Captures of SwissCovid Raising an Alert

The experiment was performed for SwissCovid and Immuni. Fig. 5 shows screenshots of an alert on SwissCovid and Fig. 6 shows pictures of an alert on Immuni. Observe that while SwissCovid allows to take screenshots of alerts, Immuni prevents that for security reasons.

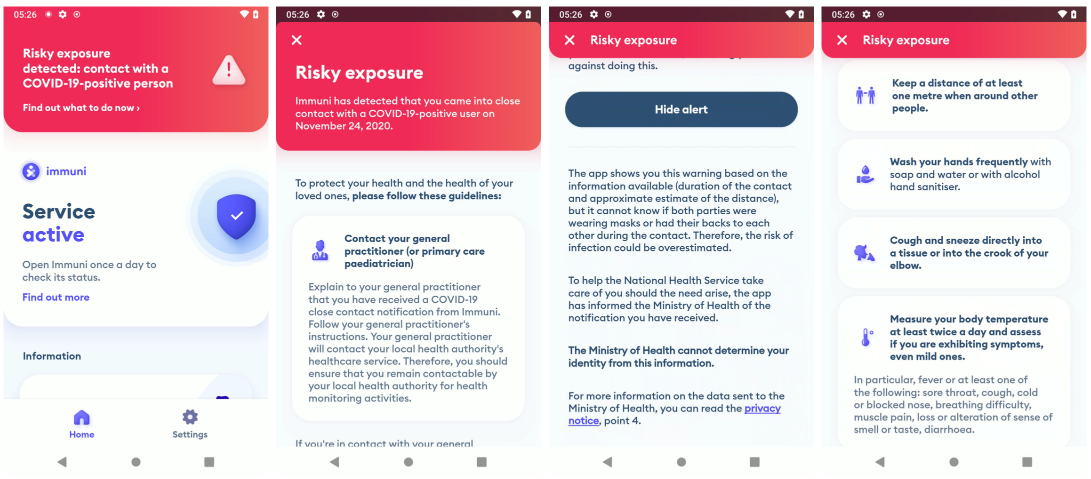
6.2 Rogue Base Station

We list here a few assumptions.

– The victim must be within the range of the adversary rogue base station and Bluetooth USB dongle (this range can be enlarged using amplifiers).
– The victim smartphone is registered to the rogue base station.
– We also assume that the victim has not pulled yet the last updated TEK-list which is used by the adversary. (Otherwise, the victim will not try to match it and the attack fails.)

For this attack we used a mini PC Fitlet2 (Intel J3455) with a Bluetooth USB dongle and the Software Defined Radio (SDR) USRP B200-mini (Fig. 7). We used several open source projects such as Osmocom suite [4], OpenBTS [3], YateBTS [8] and srsLTE [7]. We eventually setup a 2G rogue base station because of the lack of mutual authentication. Our tests were

[15] He can also trigger it with a Denial-of-Service attack to reboot the smartphone.


05:26

Risky exposure detected: contact with a COVID-19-positive person

Find out what to do now ›

immuni

Service active

Open Immuni once a day to check its status.

Find out more

Information

Home

Settings

05:26

Risky exposure

Immuni has detected that you came into close contact with a COVID-19-positive user on November 24, 2020.

To protect your health and the health of your loved ones, **please follow these guidelines:**

Contact your general practitioner (or primary care paediatrician)

Explain to your general practitioner that you have received a COVID-19 close contact notification from Immuni. Follow your general practitioner's instructions. Your general practitioner will contact your local health authority's healthcare service. Therefore, you should ensure that you remain contactable by your local health authority for health monitoring activities.

If you're in contact with your general

05:26

Risky exposure

against doing this.

Hide alert

The app shows you this warning based on the information available (duration of the contact and approximate estimate of the distance), but it cannot know if both parties were wearing masks or had their backs to each other during the contact. Therefore, the risk of infection could be overestimated.

To help the National Health Service take care of you should the need arise, the app has informed the Ministry of Health of the notification you have received.

The Ministry of Health cannot determine your identity from this information.

For more information on the data sent to the Ministry of Health, you can read the privacy notice, point 4.

05:26

Risky exposure

Keep a distance of at least one metre when around other people.

Wash your hands frequently with soap and water or with alcohol hand sanitiser.

Cough and sneeze directly into a tissue or into the crook of your elbow.

Measure your body temperature at least twice a day and assess if you are exhibiting symptoms, even mild ones.

In particular, fever or at least one of the following: sore throat, cough, cold or blocked nose, breathing difficulty, muscle pain, loss or alteration of sense of smell or taste, diarrhoea.

Fig. 6. Pictures of Immuni Raising an Alert

realized in a Faraday cage to comply with legal regulations. Since the rogue base station also manages network access, we block TEK-list updates and NTP requests. Note that the cost of the attack can be significantly reduced by using a modified Motorola C123 mobile phone as the SDR [5] or even a USB-to-VGA dongle [6]! Below we describe our attack with the Master of Time variant, Hence, the whole attack takes less than a second.

1. The adversary retrieves the updated TEK-list (which is publicly available) from the official server.
2. He picks a new TEK and derives an RPI and AEM. The emission power is set to low to improve the chances for the RPI to be accepted.
3. Using NITZ message, the adversary sends the smartphone to the past at the corresponding date and time of the RPI.
4. He sends the RPI||AEM using the Bluetooth USB dongle, following the Master of Time attack with NITZ. This requires less than a second.
5. The active part of the attack can stop here. The adversary can wait for the smartphone to restore date and time by disconnecting it from the rogue base station or sets the clock back to normal on the smartphone with a last NITZ message.
6. The app eventually updates the TEK-list. The replayed RPI will be considered as genuine since the exposure duration is more than 15 minutes, within the defined time frame and emitted with low power. Hence, an alert will be raised.

6.3 Experimenting the Attack with a Journalist

A demo of the attack on the Immuni's app was carried out in presence of a journalist of the Italian television RAI who was seemingly interested in making the public aware of the danger of replay attacks in general. (The demo and the interview did not focus on time-traveling phones and the more sophisticated technicalities we show in this paper). For this, the journalist bought a new smartphone (Samsung Galaxy A21S) on October 16, 2020 and we used it as a target. At the end of the demo, the journalist saw an alert for a close contact which was supposed to have occurred on October 14, two days before the smartphone was

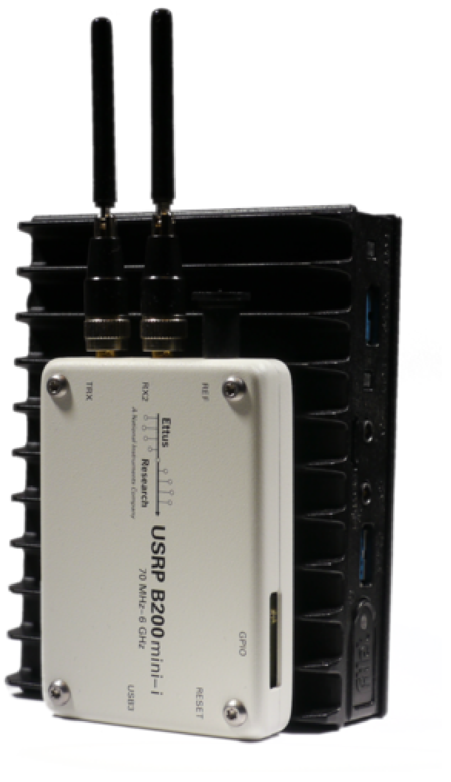
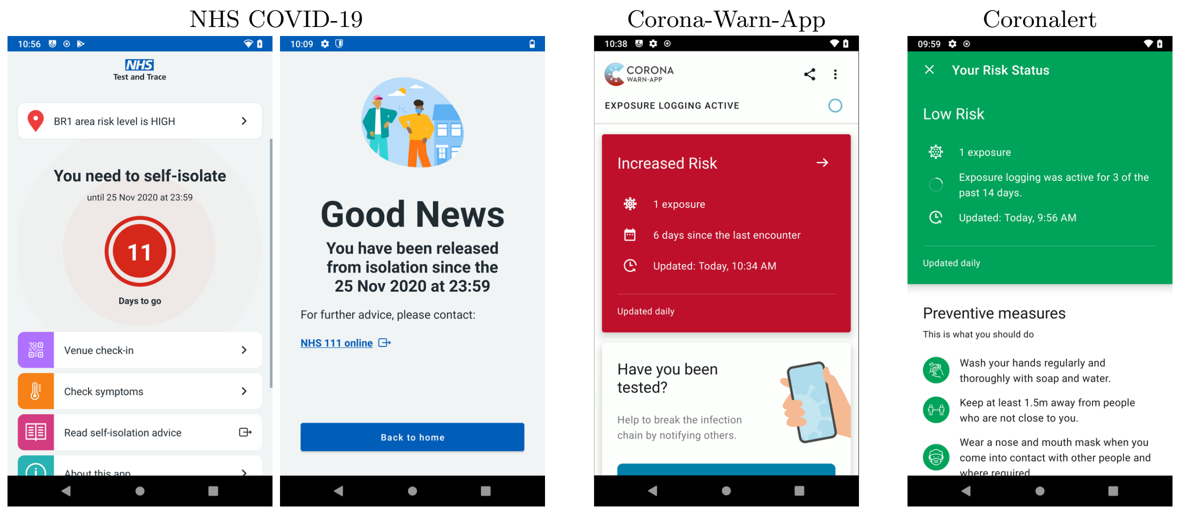


Fig. 7. Rogue Base Station USRP B200-Mini with a Fitlet2

bought! Every step, from the purchase of the phone to the display of the alert, was filmed by the journalist.[16]

6.4 Other EN-based Apps

We tried few other EN-based apps with success. With NHS COVID-19 (the app which is used in England and Wales), the app puts the user in quarantine and releases it after a few days (we used our time machine to check it). The app also shows at-risk areas (like BR1 which is in London). Fig. 8 also shows Corona-Warn-App (the app in Germany) and Coronalert (Belgium).



10:56

NHS Test and Trace

BR1 area risk level is HIGH

You need to self-isolate
until 25 Nov 2020 at 23:59

11
Days to go

Venue check-in
Check symptoms
Read self-isolation advice
About this app

10:09

Good News
You have been released from isolation since the 25 Nov 2020 at 23:59

For further advice, please contact:
NHS 111 online

Back to home

10:38

CORONA WARN-APP

EXPOSURE LOGGING ACTIVE

Increased Risk
1 exposure
6 days since the last encounter
Updated: Today, 10:34 AM
Updated daily

Have you been tested?
Help to break the infection chain by notifying others.

09:59

Your Risk Status

Low Risk
1 exposure
Exposure logging was active for 3 of the past 14 days.
Updated: Today, 9:56 AM
Updated daily

Preventive measures
This is what you should do
Wash your hands regularly and thoroughly with soap and water.
Keep at least 1.5m away from people who are not close to you.
Wear a nose and mouth mask when you come into contact with other people and where required

Fig. 8. Screen Captures of Various EN-Based Apps Raising an Alert

7 Conclusion

In this paper, we demonstrate that alert injections against the EN of Google and Apple are easy to achieve even without collaboration or corruption of infected individual, Apple, Google, or health authorities. The attack requires little equipment, is fast, and can be done by anyone. It could be done at scale too.

Moreover, people can generate alert injections on their own phones motivated by different purposes: e.g., frightening their own family members or friends, showing the alert to their employers to be exempted by work, or simply to have priority to the COVID-19 test with the terrible side effect of congesting the health system. We point out that in Italy the Ministry of Health does receive a notification when a user is alerted and being such notifications anonymous is impossible for the authorities to distinguish genuine alerts from fake ones; terrorists and criminals could generate fake alerts on phones controlled by themselves to make the Italian health authorities believe that there are much more at-risk individuals, so to induce Italian authorities to take drastic political decisions.

[16] `https://www.rai.it/programmi/report/`

The time machine's techniques we describe in this work are also useful to test the EN system offline and to figure out details that are not public since EN is not open source and Google and Apple restrict access to the EN's API to health authorities.

Acknowledgements. The authors thank Biagio Pepe, Mario Ianulardo, and Shinjo Park for useful suggestions and technical advice.

References

1. Exposure Notification. Cryptography Specification. v1.2 April 2020. Apple & Google.
   `https://www.google.com/covid19/exposurenotifications/`
2. SwissCovid Exposure Score Calculation. Version of 11 September 2020.
   `https://github.com/admin-ch/PT-System-Documents/blob/master/SwissCovid-ExposureScore.pdf`
3. Range Networks. OpenBTS.
   `http://openbts.org`
4. Osmocom Suite.
   `https://osmocom.org/`
5. OsmocomBB.
   `https://projects.osmocom.org/projects/baseband`
6. osmo-fl2k.
   `https://osmocom.org/projects/osmo-fl2k`
7. srsLTE.
   `https://www.srslte.com/`
8. Yate. YateBTS.
   `https://yatebts.com`
9. Gennaro Avitabile, Daniele Friolo, Ivan Visconti. TEnK-U: Terrorist Attacks for Fake Exposure Notifications in Contact Tracing Systems. Cryptology ePrint Archive: Report 2020/1150. IACR.
   `http://eprint.iacr.org/2020/1150`
10. Paul-Olivier Dehaye, Joel Reardon. SwissCovid: a Critical Analysis of Risk Assessment by Swiss Authorities. Preprint arXiv:2006.10719 [cs.CR], 2020.
   `https://arxiv.org/abs/2006.10719`
11. Douglas J. Leith, Stephen Farrell. Measurement-Based Evaluation Of Google/Apple Exposure Notification API For Proximity Detection In A Light-Rail Tram. PLoS ONE 15(9) 2020. `https://doi.org/10.1371/journal.pone.0239943`
12. Shinjo Park, Altaf Shaik, Ravishankar Borgaonkar, JeanPierre Seifert. White Rabbit in Mobile: Effect of Unsecured Clock Source in Smartphones. Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM@CCS 2016, Vienna, Austria, October 24, 2016, pp. 13–21. ACM 2016.
   `https://doi.org/10.1145/2994459.2994465`
13. Serge Vaudenay. Analysis of DP3T. Cryptology ePrint Archive: Report 2020/399. IACR.
   `http://eprint.iacr.org/2020/399`
14. Serge Vaudenay. Centralized or Decentralized? The Contact Tracing Dilemma. Cryptology ePrint Archive: Report 2020/531. IACR.
   `http://eprint.iacr.org/2020/531`
15. Serge Vaudenay, Martin Vuagnoux. Analysis of SwissCovid.
   `https://lasec.epfl.ch/people/vaudenay/swisscovid/swisscovid-ana.pdf`