Contact Tracing Made Un-*relay*-able

Marco Casagrande
Department of Mathematics
University of Padua
Padua, Italy
m.casagrande.1993@gmail.com

Mauro Conti
Department of Mathematics
University of Padua
Padua, Italy
conti@math.unipd.it

Eleonora Losiouk
Department of Mathematics
University of Padua
Padua, Italy
eleonora.losiouk@unipd.it

***Abstract*—Automated contact tracing is a key solution to control the spread of airborne transmittable diseases: it traces contacts among individuals in order to alert people about their potential risk of being infected. The current SARS-CoV-2 pandemic put a heavy strain on the healthcare system of many countries. Governments chose different approaches to face the spread of the virus and the contact tracing apps were considered the most effective ones. In particular, by leveraging on the Bluetooth Low-Energy technology, mobile apps allow to achieve a privacy-preserving contact tracing of citizens. While researchers proposed several contact tracing approaches, each government developed its own national contact tracing app.**

In this paper, we demonstrate that many popular contact tracing apps (e.g., the ones promoted by the Italian, French, Swiss government) are vulnerable to relay attacks. Through such attacks people might get misleadingly diagnosed as positive to SARS-CoV-2, thus being enforced to quarantine and eventually leading to a breakdown of the healthcare system. To tackle this vulnerability, we propose a novel and lightweight solution that prevents relay attacks, while providing the same privacy-preserving features as the current approaches. To evaluate the feasibility of both the relay attack and our novel defence mechanism, we developed a proof of concept against the Italian contact tracing app (i.e., *Immuni*). The design of our defence allows it to be integrated into any contact tracing app. To foster the adoption of our solution in contact tracing apps and encourage developers to integrate it in the future releases, we publish the source code.

***Index Terms*—Contact Tracing Apps, Bluetooth Low-Energy, Relay Attacks, Android Platform**

I. Introduction

The SARS-CoV-2 pandemic caught governments and healthcare systems unprepared. One of the main issues of the virus concerns the speed of diffusion, which is very high, especially in comparison to the time required to find all the people that have been in contact with an infected person. Thus, the very first strategy adopted by some governments was to put citizens in a strict lockdown. However, the lockdown was only a temporary solution and governments started looking for alternative approaches aimed at the containment of the virus, which means: (i) rapid identification of the infected individuals, with the consequent quarantine, (ii) identification of the people that have been close to an infected person in the previous days and weeks, (iii) decontamination of places where the infected individual has been. Guaranteeing the containment is a difficult task, which might lead to errors, especially if performed manually, as it happened at the start of the virus spread. As a result, governments expressed their need for Automatic Contact Tracing (ACT) solutions, which can help with better monitoring the spread of the virus if applied together with social distancing.

Several ACT solutions were proposed and adopted by different countries according to their economical, technological and cultural status. The most promising tool for collecting contact tracing data that concern the citizens was the smartphone, since most people own one. Tracing citizens contacts through the smartphone location (i.e., GPS signal, Wi-Fi routers, cellular networks) seemed a feasible solution [1], which, however, introduced significant privacy issues: citizens had to partially give up on their privacy to protect others. As illustrated in [2], such privacy issues could even lead to public identification of diagnosed patients or to mass surveillance. Thus, researchers proposed an alternative approach based on the Bluetooth Low-Energy (BLE) technology, which is already available on most smartphones. BLE allows two devices that are close to each other to exchange their identifiers, while requiring a limited amount of battery power. Several communication protocols have been proposed to support ACT on mobile devices: BlueTrace [3], ROBust and privacy-presERving proximity Tracing (ROBERT) [4] and Decentralized Privacy-Preserving Proximity Tracing (DP-3T) [5]. In addition, Google and Apple designed the Google Apple Exposure Notification (GAEN) [6] protocol. GAEN enables the interoperability among devices running Android and iOS, by providing a common API in the underlying platform to be used by contact tracing apps. All the above-mentioned protocols follow the same workflow: each smartphone has its own pseudonym, shared with other smartphones when they get close to each other. After a while, the smartphone updates its pseudonym with a new one, which is seemingly independent. Thus, each smartphone will soon have a database of the announced pseudonyms and a database of the received pseudonyms. When a person is found infected, all the smartphones that have been in contact with the smartphone of the infected person should be notified, also receiving a risk score. By proving the unlinkability of pseudonyms, the above approach guarantees a good level of privacy. Despite its privacy-preserving nature, the BLE-based ACT approach has several limitations which reduce the citizens consensus to adopt it. Among such limitations, there is the resilience to security attacks. In particular, the absence of an authentication procedure in ACT apps paves

the way for replay and relay attacks. Previous works [7], [8], [9] designed possible solutions to defend against such attacks. However, they either rely on an interactive approach and on sharing contact time and location between the parties [7], or involve complex cryptographic schemes which might make them unfeasible in a real-world scenario [8]. Finally, all of them [7], [8], [9] are theoretical approaches that have not been implemented on a real ACT solution.

In this paper, we focus on the vulnerability to relay attacks of ACT apps and we prove that many popular ones (e.g., those promoted by the Italian, French, Swiss government) are not able to defend against such attacks. To tackle this security issue, we designed *ACTGuard*, the first solution that (i) does not require an interactive approach between the two devices; (ii) does not imply the sharing of location and time information between the two devices; (iii) relies on a simple hashing function, without the need of additional cryptographic schemes; (iv) is feasible in a real-world scenario, as proven by its proof of concept implemented over a real ACT app.

ACTGuard effectively prevents relay attacks by acquiring the location data concerning contacts between pairs of people. In particular, it saves on the mobile device only a hash of the contact, generated by providing as the input of a hash function the pseudonyms of the two smartphones, the timestamp of the contact and the location of the contact. Whenever a person is found infected, the hashes of the contacts over the previous days are shared with a remote server and locally downloaded by other smartphones. If the pseudonym of the infected person has been spoofed and used in a relay attack (i.e., being retransmitted by the attacker in a different location than the person real one), the hashes of the relay attack victims will mismatch the ones of the infected person due to the different location. This way, *ACTGuard* will prevent the relay attack victims from receiving a false positive alert. Moreover, by saving and sharing only the hashes of the contacts, *ACTGuard* provides a privacy-preserving solution. To assess the feasibility of the relay attack against ACT apps and the reliability of *ACTGuard*, we developed two proofs of concept focusing on *Immuni*, the Italian contact tracing app based on GAEN.

Contributions. The contributions of the paper are as follows:

- We analyzed the design of popular ACT apps based on GAEN and identified a vulnerability against relay attacks;
- We designed the first solution that is compliant with the current GAEN internal architecture and prevents relay attacks, called *ACTGuard*;
- We implemented a proof of concept of a relay attack and of *ACTGuard* against *Immuni*, the Italian ACT app;
- We released the source code of the attack and of the defence, as well as a demo video of both[1].

Organization The rest of the paper is organized as follows: in Section II, we describe the BLE technology, and the existing proximity tracing solutions; Section III introduces the system and threat model of our relay attack; Section IV outlines the design of *ACTGuard*; Section V describes our implementation of the relay attack against *Immuni* and *ImmuniGuard*; Section VI illustrates the related work; in Section VII, we conclude the paper with a discussion about ACT apps.

[1]https://github.com/SPRITZ-Research-Group/ImmuniGuard

II. Background

In this section, we illustrate the BLE protocol (i.e., Section II-A) and the proximity tracing protocols (i.e., Section II-B). We focus on the Android Operating System (OS), since both the attack and the defence target this platform.

A. Bluetooth

Bluetooth protocol. The Bluetooth protocol always involves a client and a server. Usually, the server keeps running in discoverable mode until a client sends it an inquiry. Then, the server sends its information to the client (e.g., list of services), that attempts to read server's data. The access to such data is regulated by a set of permissions: no permission (the client can access server's data); authentication required (the client starts the *pairing procedure*); authorization required (any implementation concerning the authorization is left to developers). The *pairing procedure* allows a device to authenticate a different one, before connecting to it, and to share with the other device the long-term keys that will encrypt the communication. During the authentication, the user is asked to verify the identity of the remote device. This process might go through an interaction with the remote device display or through the input of a value into the remote device, according to its features. Alternatively, the *pairing procedure* might also happen through the *just works* mode, in case the remote device has neither a display nor an input. When the *pairing* is completed, the two devices go through the *bonding procedure*, during which they exchange the long-term keys. Unless a device is manually reset or unpaired, it will keep the key stored and use it to encrypt the communication at the link layer.

Use of Bluetooth in Android. In Android, applications willing to use the Bluetooth communication channel have to rely on the `android.bluetooth` API package, that enables the interaction with the system process under `/packages/apps/Bluetooth`. Moreover, since Bluetooth is classified as a protected resource, applications have to declare specific permissions: `BLUETOOTH` (to support Bluetooth communications), `BLUETOOTH_ADMIN` (to discover nearby Bluetooth devices and to change smartphone Bluetooth setting), `BLUETOOTH_PRIVILEGED` (to avoid any user interaction with the device during the *pairing procedure*). When an application is granted the required permissions, it can start communicating with external Bluetooth devices. If an external device requires a *pairing procedure*, this is usually handled by a single application. However, the bonding information of the external device is saved in a shared location, that can be accessed by any application with Bluetooth related permissions. As soon as it has the permissions, any application can communicate with an external Bluetooth device, if this has been already paired with the smartphone.

B. Proximity Tracing Protocols

BlueTrace [3], ROBERT [4], DP-3T [5] and GAEN [6] are among the most well-known proximity tracing protocols worldwide. Proximity tracing consists in registering any physical contact between individuals, and it is a core part of the contact tracing process. Despite sharing the same technology (i.e., BLE on mobile devices), such protocols differ in the role assigned to the centralized server, which is controlled by the national health authority. According to this, the protocols can be classified into two different approaches, both shown in Fig. 1: *centralized* and *decentralized*. The general workflow of proximity tracing protocols can be summarized as follows:

- Set up and configuration: in the *centralized* approach, the app registers to the health authority server and it receives multiple Ephemeral Bluetooth Identifier (EBID). On the contrary, the *decentralized* approach assumes that the app locally generates its own EBID.
- Contact between two individuals: when two app users meet each other, the respective apps share their own EBID.
- Announcement of a new positive: when an app user is found infected, the *centralized* approach requires the user to share with the health authority server all his own EBID and the ones of the people met. On the contrary, the *decentralized* approach requires the infected user to share only his own EBID remotely.
- Risk score calculation: in the *centralized* approach, the health authority server calculates the risk for all the EBID that have been in contact with the infected one and it sends the notifications accordingly. In the *decentralized* approach, the apps periodically download the EBID of the new infected people and locally calculate the risk score.

We will now provide more details about each proximity tracing protocol.

BlueTrace. BlueTrace [10] is a proximity tracing protocol designed by the Government Digital Services team at the Government Technology Agency of Singapore and it adheres to the *centralized* approach. It was implemented on the first national ACT app deployed in the world, i.e., TraceTogether [3]. In this case, the *centralized* approach was chosen for two reasons: 1) it allows for a human-in-the-loop design during epidemiological surveillance, and 2) it allows to better monitor the adoption and usage of the app, by logging daily requests to the server. During app registration, the centralized server assigns the user a unique User IDentifier (UserID), linked to his phone number. Then, the app generates multiple Temporary IDentifier (TempID) from the UserID, thus allowing the centralized server to trace back the UserID from the TempID. The messages exchanged between apps as advertisement packets contain the following data: TempID (to prevent third-party tracking); Device Model (to improve the calculation of distance estimates); Organization Code (to indicate the country and health authority); BlueTrace Protocol Version.

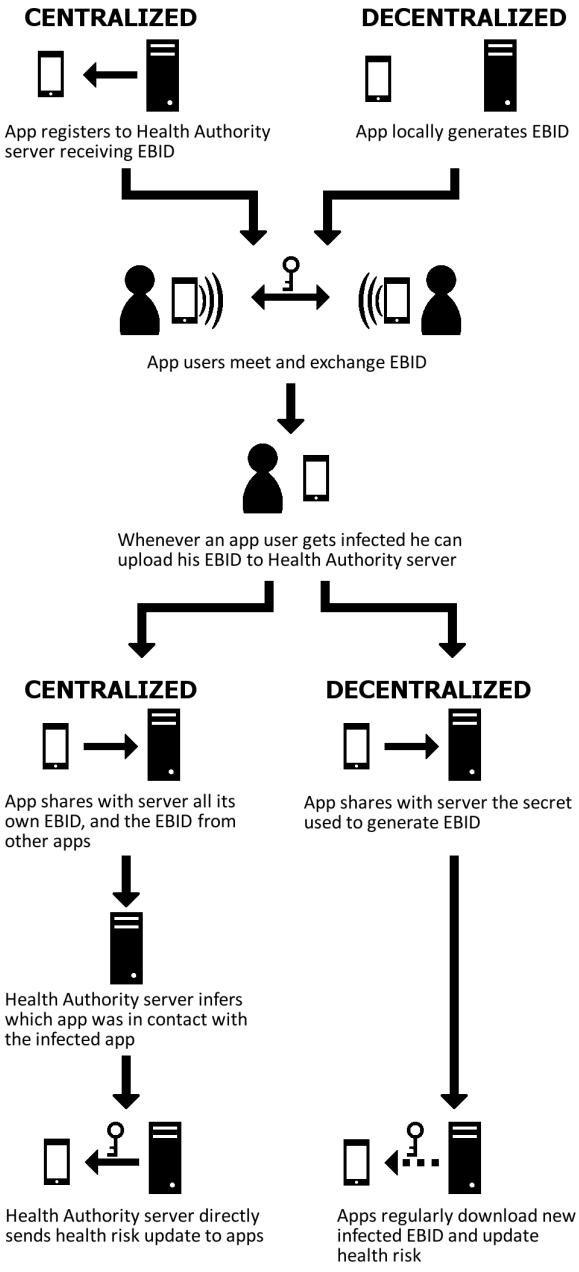

Fig. 1: Comparison between centralized and decentralized approaches.

When a user is found positive to SARS-CoV-2, BlueTrace allows him to share his contacts history with the centralized server, together with additional data (e.g., RSSI, device model, timestamp). The server decrypts the TempID in order to retrieve the UserID and the validity period. Then, it finds the closest contacts according to time of exposure and distance. During the medical interview of infected users, proximity and duration filtering thresholds are fine-tuned, before any in-app exposure warning is sent.

ROBERT. PEPP-PT [11] is a non-profit organization, based in Switzerland, with 130 members across eight European

countries. The objective of PEPP-PT is to develop a privacy-preserving digital proximity-tracing architecture. Part of the PEPP-PT proposal involved the design of a *centralized* contact tracing protocol, named ROBERT [4]. During app registration, the centralized server assigns to the user a permanent IDentifier (ID), saved in its database, and several EBID, which are valid within a specific time window. Apps implementing the ROBERT protocol regularly broadcast packets containing their EBID, and store EBID received from other users. Whenever a user is found positive to SARS-CoV-2, ROBERT requires the user to upload the list of EBID collected from other users in the past weeks. The centralized server is able to trace back the ID from uploaded EBID, flags them as exposed, calculates the risk score and warns app users accordingly.

DP-3T. DP-3T [5] is a European consortium of technologists, legal experts, engineers and epidemiologists with the primary objective of developing a proximity tracing protocol able to prevent mass surveillance. The DP-3T protocol adheres to the *decentralized* approach and provides three slightly different designs: (i) a *low-cost decentralized proximity tracing* protocol (good privacy and very small bandwidth required); (ii) an *unlinkable decentralized proximity tracing* protocol (far better privacy than the first solution, but with an increased bandwidth requirement); (iii) a *hybrid decentralized proximity tracing* protocol (a combination of the two previous designs). From now on, we will implicitly refer to the *hybrid decentralized proximity tracing* design, as it is more well-rounded: it features a high level of protection against linking EBID to the real identity of individuals, while still retaining a low-cost philosophy. An app implementing DP-3T does not communicate with the centralized server during the setup process. On the contrary, it locally generates a Secret Key (SK), used to generate several EBID, which are valid within a specific time window. Apps implementing DP-3T regularly broadcast packets containing their EBID, and store the EBID received from other users. Whenever a user is found positive to SARS-CoV-2, DP-3T requires him to upload past SK, along with the associated time windows. The centralized server simply acts as a database, allowing other app users to connect and download the newly uploaded data. By knowing a SK and its time windows, apps can generate EBID and compare them to the ones they collected during their proximity tracing activity. The health risk calculation is performed locally by the app, and the health status is updated immediately.

GAEN. The result of the collaboration between Google and Apple is the GAEN [6] protocol, which aims at enabling BLE interoperability between Android and iOS devices. GAEN provides a set of Application Programming Interface (API) that are restricted only to the developers authorized by their own Government to release a national ACT app. GAEN shares several design features with the DP-3T *hybrid decentralized proximity tracing* and it adheres to the *decentralized* approach (for more details on the GAEN protocol, please, refer to Section III).

ACT Apps Overview. BlueTrace, ROBERT, DP-3T and GAEN are proximity tracing protocols, which can be implemented by any national ACT app. Table I shows an overview of the ACT apps that we analyzed. During our selection, we aimed to include entries for each of the most popular proximity tracing protocols, to represent a wide variety of countries and to showcase the various combination of technologies and privacy approaches with respect to the following criteria: the adopted proximity tracing protocol, the technology used for performing the contact tracing, the set of personal data required to be shared by the app user and, finally, the resilience to replay and relay attacks. As shown in the table, almost all apps are vulnerable to relay attacks. This is the main motivation that encouraged us to provide a design for a defence mechanism, that prevents such attacks, while being compliant to existing protocols (for more details on the ACT apps, please, refer to Section VI).

III. System Model and Threat Model

In this section, we illustrate the system model (i.e., Section III-A) and the threat model (i.e., Section III-B) we considered for designing a relay attack against ACT apps and for our defence mechanism, i.e., *ACTGuard*. In particular, among the different proximity tracing protocols, we chose to focus on GAEN-based contact tracing apps, as GAEN is the most widely adopted solution in Europe.

A. System Model

Most GAEN-based apps merely act as an interface to the underlying protocol. Thus, our security analysis and design of *ACTGuard* can be applied to any GAEN-based contact tracing app. Here, we provide the details of the GAEN protocol.

Pseudonym Generation and Exchange. Once a GAEN-based app is installed on a device, and the onboarding process is completed, it generates a 16-byte Temporary Exposure Key (TEK), which is valid for a single day and then replaced by a new one. As shown in Figure 2, a GAEN-based app generates a 16-byte Rolling Proximity Identifier Key (RPIK) and an Associated Encrypted Metadata Key (AEMK) from its current TEK. Finally, the app derives the Rotating Proximity Identifier (RPI) from the RPIK. Since RPI are not linked to a specific person or device, they can be exchanged as pseudonyms during contact tracing. Every time the smartphone BLE MAC randomized address changes, the current RPI needs to be updated accordingly, and a new one is derived again from the RPIK, with a two-hour validity window. Advertisement packets are broadcasted by the app via BLE, and stored locally by other nearby apps.

Announcement of a New Positive Person. When a new user is found positive to SARS-CoV-2, he can choose to upload his recent TEK (also referred as Diagnosis Key) to the centralized server. Other apps periodically download the TEK of newly infected users, infer the associated RPI and check if those RPI match with the ones from the app's own recent contacts. If a match is found, the app uses the AEMK to decrypt the Associated Encrypted Metadata (AEM) and to evaluate the transmission signal strength, which contributes to the final risk score, alongside with the duration of the contact.

ACT App	Country	Proximity Tracing Protocol	GPS	Bluetooth	Data Shared	Replay	Relay
TraceTogether [3]	Singapore	OpenTrace	✗	✗	Phone Number	✓	✓
Safe Paths [2]	United States	PACT	✓	✗	GPS Location	✓	✗
Hamagen [12]	Israel	Proprietary Protocol	✓	✗	GPS Location	✓	✓
Aarogya Setu [13]	India	Proprietary Protocol	✓	✗	Personal Data	✓	✓
StopCovid [14]	France	ROBERT	✓	✓	Pseudonyms	✓	✓
SwissCovid [15]	Switzerland	GAEN	✗	✓	Pseudonyms	✓	✓
Covid Alert NY [16]	United States	GAEN	✗	✓	Pseudonyms	✓	✓
Immuni [17]	Italy	GAEN	✗	✓	Pseudonyms	✓	✓

TABLE I: Overview of the Analyzed ACT Apps (✓ means that the ACT app supports the specified feature, ✗ otherwise).

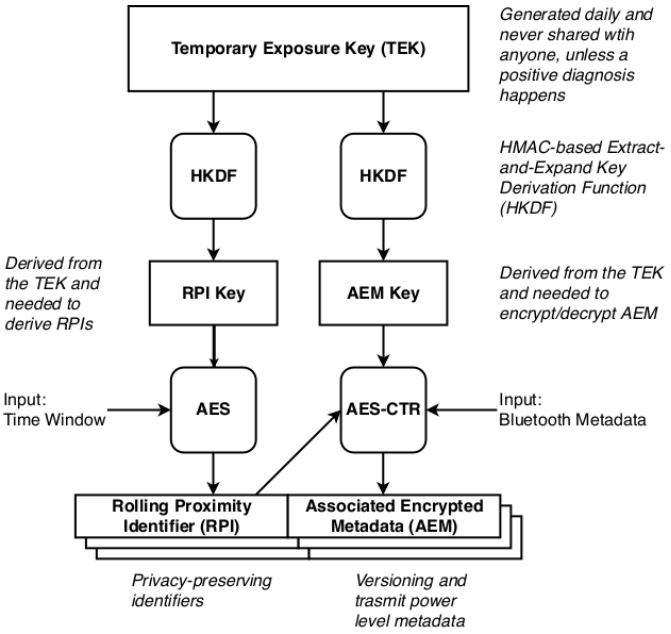

Fig. 2: GAEN - Key Generation Workflow.

B. Threat Model

In our threat model, as in the scenarios described in Section IV, we assume to have the components shown in Fig. 3: a set of three honest users, among which one is positive to SARS-CoV-2; two malicious attackers; a set of servers (e.g., the health authority one); a GAEN-based app; the *ACTGuard* app.

Our relay attack against GAEN-based apps utilizes the threat model shown in Fig. 4. It involves the actors detailed in Table II: three honest users (i.e., A, B and C), with an active GAEN-based app on their smartphone; two adversaries (i.e., Adv1 and Adv2) without any GAEN-based app on their smartphones.

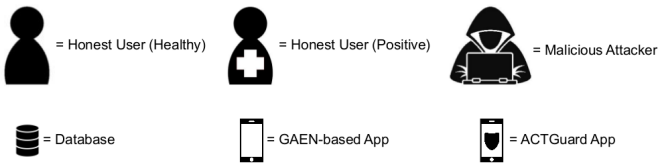

Fig. 3: Symbols Explanation.

Actor	Alignment	GAEN-based App
Alice	Honest	✓
Bob	Honest	✓
Carlos	Honest	✓
Adv1	Malicious	✗
Adv2	Malicious	✗

TABLE II: GAEN Relay Actors (✓ means the actor has the specified equipment, ✗ otherwise).

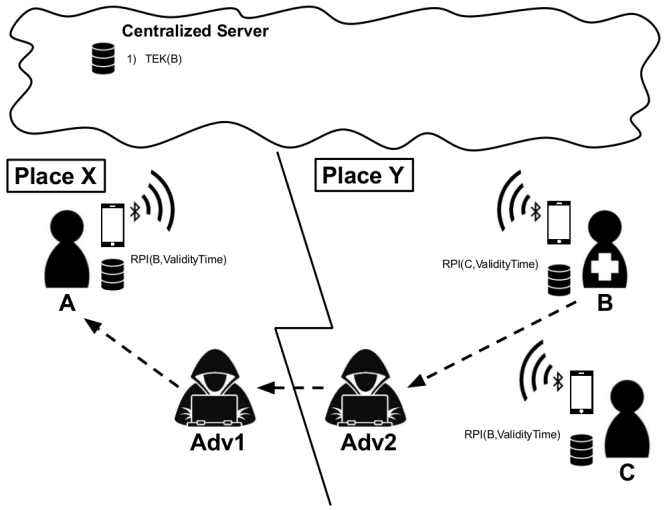

Fig. 4: GAEN Relay Attack.

In this scenario, we assume that A and Adv1 are positioned in Place X, while B, C and Adv2 are positioned in Place Y. In our threat model, we assume that Adv2 stands near B and relays his advertising data to Adv1. In particular, Adv2 aims at stealing the signal transmitted by B, who is going to be found positive in a near future. Once acquired, Adv2 shares such signal with Adv1, which starts advertising it in a different location, exposing the victims to an infected pseudonym, eventually inflating their risk score and creating a false positive case. The attack is performed as described below.

Proximity Tracing. A meets Adv1 in Place X: GAEN-App(A) broadcasts BLE advertising packets containing his current RPI, i.e., RPI(A,ValidityTime). By exploiting the signal intercepted by Adv2, Adv1 maliciously broadcasts advertising packets containing RPI(B,ValidityTime), which is locally stored by the GAEN-App(A). Similarly, B meets C

in Place Y: GAEN-App(B) advertises RPI(B,ValidityTime), while storing RPI(C,ValidityTime), and GAEN-App(C) does it the other way around.

Positive Diagnosis. At some point, B gets diagnosed as positive to SARS-CoV-2 and uploads his past TEK to the centralized server, as they are needed to retrieve the corresponding RPI. GAEN-App(C) finds a match with RPI(B,ValidityTime). Consequently, C is notified about the new exposure. In this case, GAEN-App(C) is working as intended. However, GAEN-App(A) finds a match with RPI(B,ValidityTime), too. Consequently, A is notified about the new exposure. However, in reality, A only met Adv1, that was impersonating B during the relay attack. In this case, GAEN-App(A) is not working as intended.

Relay Attack Impact. Relay attacks make ACT apps significantly less reliable, as they can inflate health risk scores by creating contacts that have never occurred. Adversaries performing relay attacks can choose sensible locations: Place Y could be a location with a high exposure risk (i.e. a hospital, a quarantined city), whereas Place X could be a location with many individuals coming and going (i.e. a train station, an airport). Apart from inflating health risk scores, the attackers might also push false positive warnings to specific individuals, instead of targeting whole groups. Possible attack scenarios could be the following ones: a student enforcing a professor to the self-quarantine, thus leading to the cancellation of an important exam; the trading of pseudonyms derived from infected people in the dark net; the real-time monitoring of a person movements by locating several devices in places often visited by the targeted victim.

IV. *ACTGuard* Design

ACT apps using BLE are vulnerable to relay attacks by design: attackers can intercept and relay BLE signals involved in a contact between two users. Solutions based on authentication, location or timing data have limitations: integrating authentication mechanisms in the communication protocol introduces privacy issues; location data can be used to validate the distance between users, but it cannot be shared with anyone else, since it would still cause privacy issues; timing data could be useful, but RPI have a two-hour validity window, which could be long enough for an attacker to execute a relay attack.

We designed *ACTGuard*, considering the following two objectives: 1) enabling GAEN-based apps to defend against relay attacks; 2) guaranteeing the same privacy level as the current GAEN-based apps. Our intuition behind the design of *ACTGuard* relies on the collection of location data related to a contact between two app users. More specifically, by saving the GPS coordinates of a contact between two users, we can detect when RPI are transmitted at the same time in two different places. This method prevents the delivery of false positive alerts to victims that received relayed RPI. The design of *ACTGuard* requires the app user to save the following data for each contact occurred between user A and user B:

hash{ `RPI(A,ValidityTime), RPI(B,ValidityTime), Location, ContactTime` }

To avoid any inconsistency, RPI are ordered alphabetically. *ACTGuard* saves the set of hashes associated to contacts involving its owner. If any user is found infected, he uploads his past hashes to a central server. Other *ACTGuard* users periodically download the hashes of the infected people and check them against the ones they saved locally. If all the information concerning a contact between two users (i.e., both RPI, location, time) is equal, then hash values will perfectly match and *ACTGuard* will confirm that the contact occurred for real. If this does not happen, there are two possible reasons: 1) the infected user app did not upload the hashes to the centralized server; 2) the infected user uploaded the hashes to the centralized server, and a relay attack was performed. Besides preventing relay attacks, *ACTGuard* provides also a privacy-preserving solution, since each user only shares hashes. Thus, the centralized server is able to infer none of the contact information as hash functions are not reversible by design.

To better explain the *ACTGuard* resilience to relay attacks, we consider our threat model, which is illustrated in Fig. 4, and analyze several scenarios that differ according to which actors are equipped with *ACTGuard*. In particular, we assume that:

- A, B and C always use a GAEN-based app.
- Adv1 and Adv2 use neither a GAEN-based app nor *ACTGuard*.
- A always uses *ACTGuard*, since he is the victim and he has to defend against relay attacks.
- B and C might not use *ACTGuard*.

With the above-mentioned assumptions, we identified the following four scenarios:

- Scenario 1 - A, B and C all using *ACTGuard*.
- Scenario 2 - A and C use *ACTGuard*, but B does not.
- Scenario 3 - A and B use *ACTGuard*, but C does not.
- Scenario 4 - A uses *ACTGuard*, but B and C do not.

In the rest of the section, we will discuss Scenario 1 (i.e., Section IV-A) and Scenario 2 (i.e., Section IV-B). Since C is not infected, thus not going to upload his contact information, his adoption of *ACTGuard* does not affect the defence against the relay attack. Thus, we do not discuss Scenario 3. Similarly, the focus of the Scenario 4 would be B not having *ACTGuard*, which is already analyzed in Scenario 2. In our scenarios, we always assume that the person who will result positive to SARS-CoV-2 is not colluding with the adversary. In our proposal, we do not aim to prevent the upload of forged data, but for the attacker it is more difficult to forge data in the *ACTGuard* server due to the information required.

A. Scenario 1: All Users Having ACTGuard

In Scenario 1, we assume that all honest users (i.e., A, B and C) are equipped with a GAEN-based app and *ACTGuard*, while the attackers (i.e., Adv1 and Adv2) have none of them.

Scenario 1 is illustrated in Fig. 5 and the details concerning the different actors are summarized in Table III.

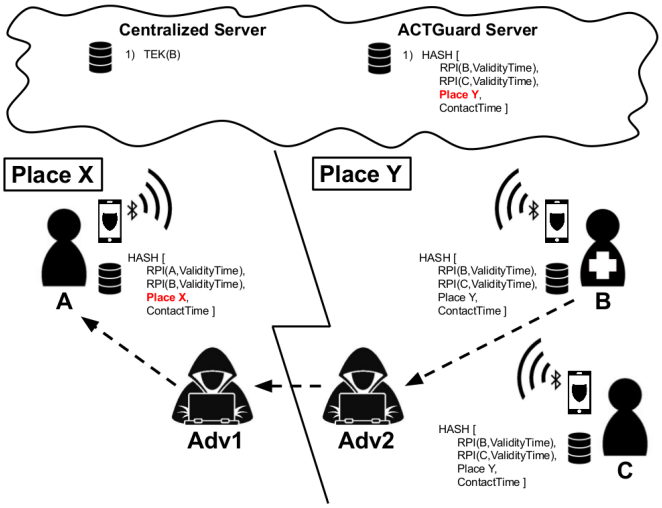

Fig. 5: Scenario 1 - All Users Having *ACTGuard*.

Actor	Alignment	GAEN-based App	ACTGuard
A	Honest	✓	✓
B	Honest	✓	✓
C	Honest	✓	✓
Adv 1	Malicious	✗	✗
Adv2	Malicious	✗	✗

TABLE III: Actors of Scenario 1 - All Users Having *ACTGuard* (✓ means the actor has the specified equipment, ✗ otherwise).

Contact. A meets Adv1 in Place X: GAEN-App(A) is advertising RPI(A,ValidityTime) and Adv1 is maliciously advertising RPI(B,ValidityTime). GAEN-App(A) stores RPI(B,ValidityTime), while ACTGuard(A) stores the hash of the contact (i.e., RPI(A,ValidityTime), RPI(B,ValidityTime), Location, ContactTime). Similarly, B meets C in Place Y: GAEN-App(B) is advertising RPI(B,ValidityTime) and GAEN-App(C) is advertising RPI(C,ValidityTime). Consequently, GAEN-App(B) stores RPI(C,ValidityTime) and GAEN-App(C) stores RPI(B,ValidityTime). At the same time, ACTGuard(B) and ACTGuard(C) both store the exact same hash (i.e., RPI(B,ValidityTime), RPI(C,ValidityTime), Location, ContactTime).

Positive Diagnosis. We assume that B gets diagnosed as positive to SARS-CoV-2 and through his GAEN-based app, he uploads his past TEK to the centralized server. B and C had a real contact. Thus, GAEN-App(C) downloads TEK(B), derives all possible RPI and finds a match with RPI(B,ValidityTime) saved in its local storage. Consequently, C is notified about a potential health risk. In this case, GAEN-App(C) is working as intended. At the same time, GAEN-App(A) finds a match with the RPI(B,ValidityTime) saved in the local storage, thus leading to a potential health risk warning, as well. However, in reality, A only met Adv1, while he was impersonating B during the relay attack. In this case, GAEN-App(A) is not able to discriminate between a real contact and a forged one. Meanwhile, ACTGuard(C) finds a correct match and it confirms the health warning from GAEN-App(C). On the contrary, ACTGuard(A) does not find a match in the hashes due to the different location of the contacts. Consequently, ACTGuard(A) does not confirm the health risk warning from GAEN-App(A). Thanks to *ACTGuard*, the relay attack against A is detected and a false positive health risk warning is prevented.

B. Scenario 2: Positive Person without ACTGuard

In Scenario 2, we assume that the honest users A and C are equipped with a GAEN-based app and *ACTGuard*, that B is only a GAEN-based app user and that the attackers (i.e., Adv1 and Adv2) have none of them. Scenario 2 is illustrated in Fig. 6 and the details concerning the different actors are summarized in Table IV.

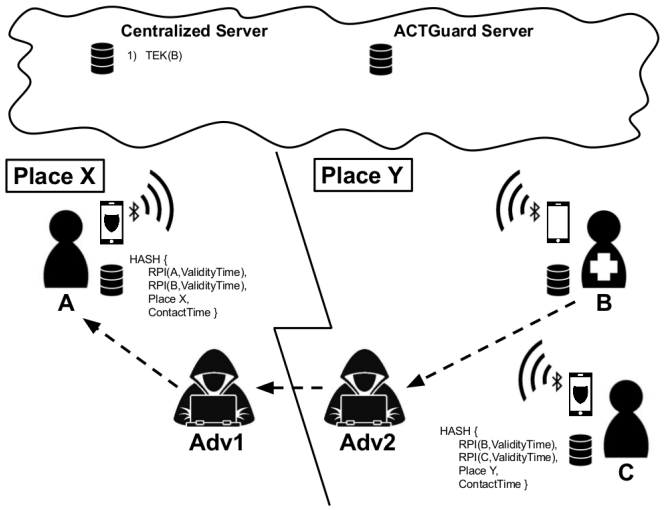

Fig. 6: Scenario 2 - Positive Person without *ACTGuard*.

Actor	Alignment	GAEN-based App	ACTGuard
A	Honest	✓	✓
B	Honest	✓	✗
C	Honest	✓	✓
Adv 1	Malicious	✗	✗
Adv2	Malicious	✗	✗

TABLE IV: Actors of Scenario 2 - Positive Person without *ACTGuard* (✓ means the actor has the specified equipment, ✗ otherwise).

Contact. A meets Adv1 in Place X: GAEN-App(A) is advertising RPI(A,ValidityTime) and Adv1 is maliciously advertising RPI(B,ValidityTime). GAEN-App(A) stores RPI(B,ValidityTime), while ACTGuard(A) stores the hash of the contact (i.e., RPI(A,ValidityTime), RPI(B,ValidityTime), Location, ContactTime). Similarly, B meets C in Place Y: GAEN-App(B) is advertising RPI(B,ValidityTime) and GAEN-App(C) is advertising RPI(C,ValidityTime).

Consequently, GAEN-App(B) stores RPI(C,ValidityTime) and GAEN-App(C) stores RPI(B,ValidityTime). At the same time, ACTGuard(C) stores the hash of the contact (i.e., RPI(B,ValidityTime), RPI(C,ValidityTime), Location, ContactTime).

Positive Diagnosis. We assume that B gets diagnosed as positive to SARS-CoV-2 and through his GAEN-based app he uploads his past TEK to the centralized server. B and C had a real contact. Thus, GAEN-App(C) downloads TEK(B), derives all possible RPI and finds a match with RPI(B,ValidityTime) saved in its local storage. Consequently, C is notified about a potential health risk. In this case, GAEN-App(C) is working as intended. At the same time, GAEN-App(A) finds a match with the RPI(B,ValidityTime) saved in the local storage, thus leading to a potential health risk warning, as well. However, in reality, A only met Adv1, while he was impersonating B during the relay attack. In this case, GAEN-App(A) is not able to discriminate between a real contact and a forged one. Both ACTGuard(A) and ACTGuard(C) do not find a match in the hashes shared by B, since B does not use *ACTGuard* and did not upload any hash. Since ACTGuard(A) is not able to confirm the health risk warning from GAEN-App(A), it cannot confirm if a relay attack actually happened.

The Scenario 2 opens up two further situations:

- A use case: A met Adv1 while he was performing the relay attack. Since B does not use *ACTGuard*, A cannot have any confirmation about the health risk alert received after the contact with the impersonator of B.
- C use case: C really met B. However, since B does not use *ACTGuard*, C cannot have any confirmation about the health risk alert received after the contact with the real B.

V. Use Case Study: the Italian Contact Tracing App Immuni

In this section, we provide an overview of *Immuni* [17], the Italian official ACT app we chose as a case study (i.e., Section V-A), the relay attack we developed against it (i.e., Section V-B) and the implementation details of *ImmuniGuard*, an instance of *ACTGuard* referring to the *Immuni* use case (i.e., Section V-C).

A. Immuni *Overview*

Immuni is a mobile contact tracing app based on the GAEN protocol [6], thus adhering to the *decentralized* approach and relying on the BLE technology for proximity tracing.

Onboarding. Once installed, *Immuni* detects the user language, checks for required updates and provides a basic explanation of how *Immuni* works. Then, *Immuni* shows the privacy note and the user agreement, asking if the user is more than fourteen years old. At last, the user submits his province of residence and enables the required app permissions.

Immuni interacts with a back-end architecture involving the following services: Exposure Ingestion Service, Exposure Reporting Service, Backend One Time Password (OTP) Service, Analytics Service, App Configuration Service.

Exposure Ingestion Service. The Exposure Ingestion Service provides a set of APIs for *Immuni* apps to enable the upload of TEK generated over the past 14 days, when an app user is found infected and he is willing to share his keys. Contextually, the infected user uploads the epidemiological information from the previous 14 days. If any epidemiological information is indeed uploaded, the user province of domicile is uploaded, too. This process can only take place with an authorised OTP from the Backend OTP Service. The Exposure Ingestion Service is also responsible for the periodical generation of TEK chunks, to be published by the Exposure Reporting Service. The TEK chunks are assigned a unique incremental index and are immutable. The province of domicile and the epidemiological information are forwarded to the Analytics Service.

Exposure Reporting Service. The Exposure Reporting Service makes the TEK chunks created by the Exposure Ingestion Service available to other apps. Only TEK chunks from the past 14 days are made available.

Backend OTP Service. The Backend OTP Service provides a set of API to the National Healthcare Service for authorising OTP that can be used to upload data from *Immuni* apps via the Exposure Ingestion Service. *Immuni* generates the OTP, that the app user personally communicates to a healthcare operator. Then, the healthcare operator inserts the OTP into the Italian Health Information System, registering it on the Backend OTP Service. The OTP automatically expires after a defined time period.

Analytics Service. The Analytics Service provides a set of API for *Immuni* apps to upload data without identifying users, both during regular operations and especially when a match is found between TEK chunks and RPI. To ensure the proper functioning of the system, maximize the effectiveness of the exposure notifications and provide an optimal healthcare assistance to users, the following information is sent to the server: TEK, epidemiological information, operational information.

TEK are required to allow other *Immuni* users to calculate their risk of being positive to SARS-CoV-2. Whenever TEK are uploaded, the following epidemiological information is uploaded, as well: exposure day, exposure duration, signal attenuation information (to estimate the distance between the two users' devices during the exposure).

Whenever an exposure detection has been completed, the following operational information is uploaded:

- Whether the device runs iOS or Android.
- Whether permission to leverage the GAEN protocol is granted.
- Whether the device BLE is enabled.
- Whether permission to send local notifications is granted.
- Whether the user was notified of a risky exposure after the last exposure detection (i.e., after the app has downloaded new TEK from the server and detected if the user has been exposed to users positive to SARS-CoV-2).
- The date on which the last risky exposure took place, if any.

Along with genuine analytics uploads, after every exposure detection event, *Immuni* apps may perform dummy analytics uploads, indistinguishable from the genuine ones.

App Configuration Service. The App Configuration Service updates the Configuration Settings every time the app starts a new background or foreground session. Such settings can be used for tuning traffic-analysis mitigation measures and update weights used in the risk prediction model.

B. Relay Attack against Immuni

The relay attack we designed against *Immuni* (the source code is available online[2]) involves the following actors:

- Immuni App 1: this is the first victim of the relay attack, since its BLE packet is sniffed and re-transmitted by the attacker.
- Malicious App 1: this is the first malicious actor, responsible for sniffing the BLE packet, which will be re-transmitted afterwards.
- Malicious Database: this database is used by the two attackers to share the sniffed BLE packet, even though they are located in different places.
- Malicious App 2: this is the second malicious actor, responsible for the re-transmission of the sniffed BLE packet.
- Immuni App 2: this is the second victim of the relay attack, since it receives the sniffed BLE packet.

As shown in Fig. 7, Immuni App 1 keeps advertising its BLE packets, which contain its current RPI. Malicious App 1 is located nearby the Immuni App 1, thus being able to intercept its advertising packets and send them to the Malicious Database. Meanwhile, Malicious App 2 downloads the sniffed BLE packets from the Malicious Database and starts advertising them, pretending to be Immuni App 1. Finally, Immuni App 2 becomes the victim of the relay attack by receiving the sniffed BLE packets from Malicious App 2.

Even though our implementation refers to the *Immuni* use case, the above-mentioned relay attack can be applied on any GAEN-based app. Since such apps do not track the user location, they are not able to detect relay attacks. Attackers can broadcast sniffed packets from any location, pretending to be the original owner of the BLE packet, as soon as it is re-transmitted within the time window of about two hours.

C. ImmuniGuard*: An* ACTGuard *Implementation for* Immuni

Our implementation of the *ACTGuard* design is called *ImmuniGuard* and it was developed considering the Italian ACT app, i.e., *Immuni*. We implemented *ImmuniGuard* as an Android app, requiring an Android SDK version equal to 29. The experiments were performed on a Xiaomi Redmi 5 Plus. Source code of the attack and defence, and a complete demo video, are available online[3].

ImmuniGuard requires the following permissions: `INTERNET` and `INTERNET_NETWORK_STATE`, to interact with an online database; `BLUETOOTH` and `BLUETOOTH_ADMIN`, for Bluetooth scanning and advertising features; `ACCESS_FINE_LOCATION`, for GPS tracking (the granularity of the GPS information can be customized); `FOREGROUND_SERVICE`, for regular monitoring of GPS locations.

Below, we will go through the main steps involved in the *ImmuniGuard* workflow.

Scanning for *Immuni* advertisement packets. *ImmuniGuard* uses the standard BLE library, provided by the Android platform, to regularly perform scans. During the scans, *ImmuniGuard* specifically looks for *Immuni* advertisement packets, identified by a value equal to "0xFD6F" in the Service UUID, as any other GAEN-based app.

Storing RPIs. For each contact with another *Immuni* app user, *ImmuniGuard* saves the advertised RPI, the current location (many ACT apps already require the GPS for scanning Bluetooth devices. *ImmuniGuard* just periodically acquires the location) and the current time. This data is stored in two databases: MyContactsTable, containing information related to contacts with other *Immuni* app users, and PositiveTable, storing information uploaded by positive users to *ImmuniGuard* database. New entries are added to MyContactsTable every time a new contact occurs, including their hash value (only the hashes of the contacts over two weeks are stored and each hash is 32-byte long). On average, calculating the MD5 of a 100MB file for 100 times requires 14 Watt [18]. *ImmuniGuard* only calculates MD5 of the contact information (100 bytes) for every new contact. PositiveTable is updated whenever *ImmuniGuard* tries to download new information from the *ImmuniGuard* online database. *ImmuniGuard* should also store the RPIs broadcasted by the *Immuni* app, but this information is currently restricted. In our current proof of concept, *ImmuniGuard* randomly generates its own dummy RPI, unrelated to *Immuni*.

Uploading data if infected. Whenever *ImmuniGuard* users are diagnosed as SARS-CoV-2 positive, they can upload the content of the MyContactsTable database, locally stored on the app. In a real scenario, the data upload should be validated by public health authorities. In our proof of concept, this functionality can be performed anytime.

Periodically downloading new infected data. *ImmuniGuard* needs to regularly download information about new users diagnosed as SARS-CoV-2 positive. In a real scenario, this process would be automated, but in our proof of concept, this functionality can be performed anytime.

Updating health risk status. The health risk status is calculated by comparing and matching the hashes in MyContactsTable and in PositiveTable, to find if the owner of *ImmuniGuard* was in proximity of an infected user. In our proof of concept, the health risk status is represented by the number of infected contacts.

D. ImmuniGuard *Limitations*

ImmuniGuard is a standalone app, and it is not allowed to access to the list of RPI broadcasted by the active instance

[2]https://github.com/SPRITZ-Research-Group/ImmuniGuard
[3]https://github.com/SPRITZ-Research-Group/ImmuniGuard

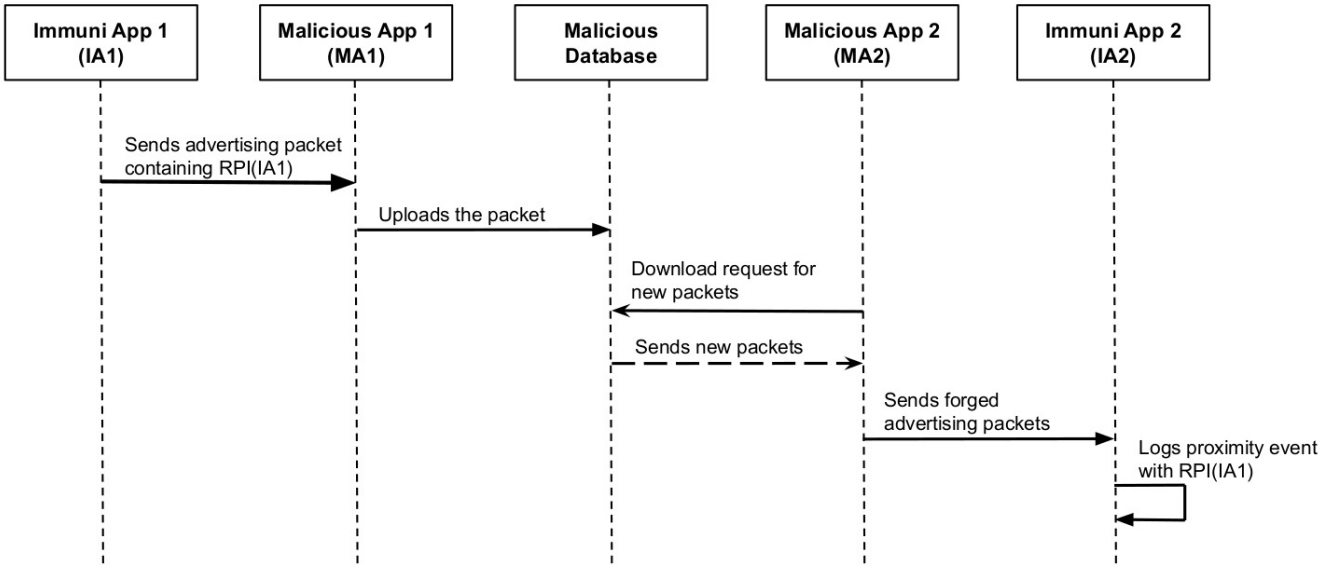

Fig. 7: Workflow of the Relay Attack we Designed against *Immuni*.

of *Immuni* app installed on the user's device. *ImmuniGuard* needs that list, in order to compute its own contact hashes, store them inside MyContactsTable and compare them with the infected hashes downloaded from the *ImmuniGuard* server. Otherwise, the detection of relay attacks is not possible. This issue can be easily solved with the integration of our solution as a feature of the official app, thus gaining direct access to the list of RPI broadcasted by *Immuni*. Since only authorized parties can use GAEN, setting up a large-scale simulation of *ImmuniGuard* would have required the development of all the components involved in contact tracing (e.g., the ACT app logic, the server). Besides requiring a significant effort, such a simulation would not add much to the proof of feasibility of *ImmuniGuard*. As an alternative, we might have stolen RPIs of real positive people and attacked the national health care system, but this approach is both unethical and illegal.

Finally, since *ImmuniGuard* has been designed as a wrapper for existing ACT apps to prevent relay attacks, some of their limitations might be inherited: among those, we can encounter the relationship between the effectiveness of our solution and the number of its users.

VI. Related Work

In this section, we describe in details the ACT apps we analyzed (i.e., Section VI-A), specifically focusing on the privacy (i.e., Section VI-B) and security (i.e., Section VI-C) issues that affect them. Finally, we illustrate previous works proposing location-based solutions against replay and relay attacks (i.e., Section VI-D).

A. Contact Tracing Apps

TraceTogether. TraceTogether [3] is the first contact tracing app based on BLE technology and developed in Singapore, one of the first countries affected by SARS-CoV-2. It implements the BlueTrace protocol, exchanging temporary identifiers through BLE advertisement packets and allowing user to willingly upload them on a centralized health authority server. During interviews with patients positive to SARS-CoV-2, medical operators ask if the patient installed TraceTogether and are able to dynamically adjust proximity and duration filtering thresholds for a better contact tracing performance.

Safe Paths. Safe Paths [2] is a secure location logging technology, based on the "Private Automated Contact Tracing (PACT)" protocol, developed by MIT researchers. It currently supports only GPS tracking, with plans for other location and proximity technologies. Safe Paths stores the GPS movement trails of its owner, who can willingly share them with an authorized public health authority as part of the contact tracing process.

StayHomeSafe. StayHomeSafe [19] is a mobile app developed by the Government of the Hong Kong Special Administrative Region. The Hong Kong procedure for compulsory quarantine requires wearing a wristband connected to the StayHomeSafe app. The app utilizes an innovative geofencing technology. The initial setup requires the user to slowly walk around the house so that the app can record data signals (e.g., Wi-Fi, Bluetooth, GPS) as a form of unique signature of that house, and detects whenever the user leaves this virtual fence. StayHomeSafe does not collect sensitive data, nor discloses the location of its user.

Hamagen. Hamagen [12] is a mobile app developed by the Israel Ministry of Health, which employs both GPS and BLE technology. Hamagen regularly saves the device location coordinates and compares them with the GPS trails of infected people. Additionally, it can perform proximity tracing through BLE. All data, which the Ministry of Health has access to, is explicitly shared by infected patients.

Aarogya Setu. Aarogya Setu [13] is a mobile app developed by the Government of India. The app is mandatory for all Indian citizens, and it is also used by employers to monitor the health risk status of their employees. Aarogya Setu uses both GPS tracking and BLE technology. During the initial setup, the user is required to submit personal data such as his/her name and phone number.

StopCovid. StopCovid [14] is a mobile app developed by the National Institute for Research in Digital Science and Technology (INRIA), as requested by the Government of France. The app uses BLE technology, and implements the ROBERT protocol, as the Government considers a centralized approach more secure.

SwissCovid. SwissCovid [15] is a mobile app developed by the Switzerland Federal Office of Public Health. The app uses BLE technology, GAEN in particular, to exchange anonymous temporary identifiers. Whenever a user is infected, he can willingly reveal and submit his temporary identifiers in order to warn other app users met in the past days.

Covid Alert NY. Covid Alert NY [16] is a mobile app commissioned by the New York State Department of Health. Akin to many other American contact tracing apps, it implements GAEN. A special feature allows users to willingly disclose personal information, anonymously, such as the user county, gender, age-range, ethnicity, and symptoms. Technical data can be also be anonymously shared with the centralized server, if the user chooses to do so.

B. Privacy Issues of Contact Tracing Apps

Since contact tracing apps aim to monitor citizens' movements, they suffer from a range of privacy issues.

Mass surveillance. Since the Government is in charge of the app and of the centralized server, mass surveillance is a plausible threat. G. Avitabile et al. [20] enumerate several types of mass surveillance scenarios applied to the DP-3T framework. Infected patients can be easily tracked by an attacker possessing a pervasive infrastructure (e.g., a corporation). The only requirement is a sufficiently large set of devices able to collect BLE signals. Infected patients upload their SK to the centralized server, and other users are able to calculate EBID from them. An attacker can track the movements of infected patients, if they stay nearby the set of devices. Another scenario involves the centralized server colluding with health authorities to map anonymous app users with real identities, since the health authorities themselves perform tests and allow data uploads.

Social graph. In a *centralized* approach, the centralized server stores all infected patients EBID. S. Vaudenay [21] argues that the *centralized* server would obtain various lists of pseudonyms unlinkable to real users. Thus, a large monitoring infrastructure and heavy data mining would be required in order to extract meaningful information. The issue has a more limited impact over *decentralized* approaches, since the ACT apps store EBID locally.

Sharing unwanted data. ACT apps may ask for, and share, more data than necessary, and potentially disclose them to third parties. Even though many ACT app developers released their app source code, most of them did not release the source code of the centralized server. Thus, there is no guarantee that the disclosed code is the real one. The Indian official ACT app, i.e., Aarogya Setu, asks for a lot of sensible information (i.e., name, phone number, age, gender, profession, workplace, recent travels) without proper motivations behind this design choice [22]. It also collects plenty of data using GPS and BLE, and the centralized server code is yet to be disclosed. D. Leith et al. [23] analyze the data shared through the GAEN API. On Android devices, the GAEN API relies on the Google Play Services, which connect to Google servers multiple times a day, sharing quite a lot of sensible data, such as: email address, phone number, IP address (used to retrieve location), phone IMEI, hardware serial number, SIM serial number and Wi-Fi MAC address.

C. Security Issues of Contact Tracing Apps

The ACT apps anonymity constraint allows any attacker to send false advertisement packets, or tamper with existing ones. Security countermeasures are scarce because of the traditionally low computational power of BLE devices.

Denial-of-Service. M. E. Garbelini et al. [24] uncover how a faulty software implementation of several BLE system-on-chip vendors exposes devices to two families of vulnerabilities. Malicious attackers in radio range can manufacture and send specific BLE packets to trigger deadlocks, crashes and buffer overflows. Smartphones may incorporate one of the affected system-on-chip, as a part of their Bluetooth module.

Replay and Relay attacks. BlueTrace researchers acknowledge the susceptibility of their protocol to replay and relay attacks [10], but do not propose any countermeasure, apart from a human-in-loop methodology. P. Dehaye et al. [25] highlight the same issue with replay and relay attacks, and illustrate the idea of tampering with the RPI and the AEM. Tampering with the transmission power level value alters the victim's risk score, making it completely unreliable.

D. Defense against Replay and Relay Attacks

Previous works [7], [8], [9] proposed defences against replay and relay attacks applied over contact-tracing solutions. S. Vaudenay [7] proposes a DP-3T extension that relies on an interactive scheme to prevent replay attacks. For an incoming advertising packet, the receiving app replies with a challenge, which is used by the advertising app to compute a Message Authentication Code (MAC) having a SK, a challenge and the contact time. The receiving app stores the MAC, the advertising app EBID and the challenge. The contact time information allows the receiving app to prevent replay attacks when local data is checked against the ones shared by a newly infected user. To prevent relay attacks, the same protocol needs to be extended by introducing an additional round of interaction where both devices know their location and the receiving app shares it with the advertising one. *The interactive approach proposed by S. Vaudenay suffers from efficiency and*

security reasons. Moreover, it relies on the sharing of location and time information between the two parties.

K. Pietrzak [8] improved the solution proposed by S. Vaudenay introducing a non-interactive protocol that relies on a weak commitment scheme and on a standard MAC to prevent replay attacks. To prevent relay attacks, the solution requires GPS coordinates, which are however not shared in clear-text. *Despite the improvements, the defence proposed by K. Pietrzak is a theoretical approach, which has not been implemented on any of the existing contact-tracing solutions.*

Finally, the work proposed by P. Madhusudan et al. [9] considers a different approach based on a digital signature procedure. The app generates an EBID, a SK and a digital signature, obtained by signing context data (i.e., location and time). EBID, context data and digital signature are then broadcasted to the receiver app that verifies their legitimacy. In case of replay or relay attacks, context data is inconsistent and allows the attack detection. *This solution relies on a set of complex cryptographic schemes, that might make its application in a real-world scenario unfeasible.*

ACTGuard is the first solution that (i) does not require an interactive approach between the two devices; (ii) does not imply the sharing of location and time information between the two devices; (iii) relies on a simple hashing function, without the need of additional cryptographic schemes; (iv) is feasible in a real-world scenario, as proven by its proof of concept implemented over a real ACT app.

VII. Discussion and Conclusion

Contact tracing is used to monitor the spread of SARS-CoV-2, but suffers from scalability issues, since it requires human intervention all the way through. Mobile apps are extremely powerful assets in the scope of ACT: they benefit from the popularity of mobile devices among the population, allow for pervasive proximity tracing and gather data in complete autonomy. Many European and Asian countries already released their own national contact tracing app, since each government could choose among the several protocols proposed by different actors: government agencies and commissions (i.e., BlueTrace), researchers (i.e., PACT, ROBERT, DP-3T), or vendors of the leading mobile OS (i.e., GAEN). To be successful, ACT solutions require consensus from the citizens, since they have to install and use them on a voluntary basis. Thus, adoption rate is the most valuable metric: A. Galanopoulos et al. [26] describe how the current adoption rate varies between 5% and 20% for many countries, while the scientific community debates that, to be effective, ACT solutions require at least an adoption rate of 60%. TraceTogether (Singapore) was downloaded by more than one million users, representing the 17% of Singaporean citizens. Aarogya Setu (India) achieved a total of 160 million downloads, amounting to 12% of the Indian population. Immuni (Italy) reached an adoption rate of 12.5% in October 2020, with more than 8 million downloads. COVID Tracker [27] (Ireland) is possibly the contact tracing app with the highest adoption rate in Europe, standing at 34%. It identifies approximately 6 close contacts per single positive case, while the South Korean mass surveillance system identifies slightly more than 10 close contacts per case [28]. There are multiple causes for such low adoption rates. User reticence is related to ACT unreliability and to the fear of privacy violation. D. Zeinalipour-Yazti et al. [29] tested Swiss, German and Italian contact tracing apps risk score calculation, and argue that BLE signal strength has little correlation to physical proximity of app users. If that was the case, the foundations of the whole ACT process would fail. Citizens are worried about the Government spying on them, and exploiting the critical situation to start a mass surveillance program. Bluetooth proximity tracing protocols offer better privacy than GPS ones, since they do not share any location data. However, they are still vulnerable to collusion between the Government and the Public Health Authority, and to classic security threats, such as relay, replay and Denial-of-Service (DoS) attacks. Possible consequences would be the disclosure of sensitive information, surveillance, quarantine enforcement, identity theft and data tampering.

In this paper, we focused on relay attacks against ACT solutions, in particular on GAEN-based apps. We found out their vulnerability to such attacks and developed a proof of concept against *Immuni*, the Italian GAEN-based app. We managed to capture the *Immuni* advertising packets from a victim app with a first malicious app and to relay them, through a second malicious app, to a designed victim. We impersonated an official GAEN-based app just by re-transmitting the original BLE advertising packets. To defend against relay attacks, we designed *ACTGuard*, a solution compliant with current GAEN design and with the privacy features of existing GAEN-based apps. *ACTGuard* locally stores the location of each contact between two app users, together with RPI of those two users and the time of the contact. However, by saving such information as the result of a hash function, *ACTGuard* fully guarantees the privacy of its users, even when that information is shared with a remote server, functioning as a database for infected user hashes. We implemented a proof of concept of *ACTGuard* by considering *Immuni*, thus releasing the *ImmuniGuard* app. We demonstrated how Immuni is vulnerable to relay attacks and how *ImmuniGuard* can effectively detect a relay attempt by finding inconsistencies between data downloaded from *Immuni* server and from *ImmuniGuard* server.

References

[1] E. S. Canlar, M. Conti, B. Crispo, and R. Di Pietro, "Crepuscolo: A collusion resistant privacy preserving location verification system," in *2013 International Conference on Risks and Security of Internet and Systems*. la Rochelle, France: IEEE, 2013, pp. 1–9.

[2] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke, D. Greenwood, C. Keegan, S. Kanaparti, R. Beaudry, D. Stansbury, B. B. Arcila, R. Kanaparti, V. Pamplona, F. Benedetti, A. Clough, R. Das, K. Jain, K. Louisy, G. Nadeau, V. Pamplona, S. Penrod, Y. Rajaee, A. Singh, G. Storm, and J. Werner, "Apps gone rogue: Maintaining personal privacy in an epidemic," 2020.

[3] G. of Singapore, "Bluetrace," 2020. [Online]. Available: https://bluetrace.io/

[4] Indria and F. AISEC, "Robert protocol," 2020. [Online]. Available: https://github.com/ROBERT-proximity-tracing/documents

[5] C. Troncoso, M. Payer, J. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, L. Barman, S. Chatel, K. Paterson, S. Capkun, D. Basin, J. Beutel, D. Jackson, B. Preneel, N. Smart, D. Singelee, A. Abidin, S. Gürses, M. Veale, C. Cremers, M. Backes, N. O. Tippenhauer, R. Binns, C. Cattuto, A. Barrat, G. Persiano, D. Fiore, M. Barbosa, and D. Boneh, "Dp3t - decentralized privacy-preserving proximity tracing," 2020. [Online]. Available: https://github.com/DP-3T/documents

[6] Apple and Google, "Exposure notification system," 2020. [Online]. Available: https://covid19.apple.com/contacttracing

[7] S. Vaudenay, "Analysis of dp3t," Cryptology ePrint Archive, Report 2020/399, 2020, https://eprint.iacr.org/2020/399.

[8] K. Pietrzak, "Delayed authentication: Preventing replay and relay attacks in private contact tracing," Cryptology ePrint Archive, Report 2020/418, 2020, https://eprint.iacr.org/2020/418.

[9] P. Madhusudan, P. Miao, L. Ren, , and V. Venkatakrishnan, "Privacy-preserving secure contact tracing, contrail project," 2020, https://github.com/ConTraILProtocols/documents/blob/master/ContrailWhitePaper.pdf.

[10] J. Bay, J. Kek, A. Tan, C. Hau, L. Yongquan, J. Tan, and T. Quy, "Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders," 2020. [Online]. Available: https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf

[11] PEPP-PT, "Pepp-pt," 2020. [Online]. Available: https://www.pepp-pt.org/

[12] I. M. of Health, "Hamagen," 2020. [Online]. Available: https://govextra.gov.il/ministry-of-health/hamagen-app/download-en/

[13] G. of India, "Aarogya setu," 2020. [Online]. Available: https://aarogyasetu.gov.in/

[14] N. I. for Research in Digital Science and T. (France), "Stopcovid," 2020. [Online]. Available: https://www.inria.fr/en/le_projet_stopcovid

[15] S. F. O. of Public Health, "Swisscovid," 2020. [Online]. Available: https://www.bag.admin.ch/bag/en/home/krankheiten/ausbrueche-epidemien-pandemien/aktuelle-ausbrueche-epidemien/novel-cov/swisscovid-app-und-contact-tracing.html\#-1032598416

[16] N. D. of Health Privacy Policy, "Covid alert ny," 2020. [Online]. Available: https://coronavirus.health.ny.gov/covid-alert-ny/

[17] P. M. O. Bending Spoons, "Immuni," 2020. [Online]. Available: https://www.immuni.italia.it/

[18] R. Damasevicius, G. Ziberkas, V. Stuikys, and J. Toldinas, "Energy consumption of hash functions," *Elektronika ir Elektrotechnika*, vol. 18, pp. 81–84, 12 2012.

[19] G. of the Hong Kong Special Administrative Region, ""stayhomesafe" mobile app user guide," 2020. [Online]. Available: https://www.coronavirus.gov.hk/eng/stay-home-safe.html

[20] G. Avitabile, V. Botta, V. Iovino, and I. Visconti, "Towards defeating mass surveillance and sars-cov-2: The pronto-c2 fully decentralized automatic contact tracing system," Cryptology ePrint Archive, Report 2020/493, 2020. [Online]. Available: https://eprint.iacr.org/2020/493

[21] S. Vaudenay, "Centralized or decentralized? the contact tracing dilemma," Cryptology ePrint Archive, Report 2020/531, 2020. [Online]. Available: https://eprint.iacr.org/2020/53

[22] S. Deb, "Privacy prescriptions for technology interventions on covid-19 in india," 2020, internet Freedom Foundation.

[23] D. Leith and S. Farrell, "Contact tracing app privacy: What data is shared by europe's gaen contact tracing apps?" 2020.

[24] M. E. Garbelini, C. Wang, S. Chattopadhyay, S. Sumei, and E. Kurniawan, "Sweyntooth: Unleashing mayhem over bluetooth low energy," in *2020 USENIX Annual Technical Conference*. Virtual Conference: USENIX Association, jul 2020, pp. 911–925. [Online]. Available: https://www.usenix.org/conference/atc20/presentation/garbelini

[25] P. Dehaye and J. Reardon, "Swisscovid: a critical analysis of risk assessment by swiss authorities," 2020. [Online]. Available: https://arxiv.org/pdf/2006.10719.pdf

[26] A. Galanopoulos, V. Valls, G. Iosifidis, and D. J. Leith, "Measurement-driven analysis of an edge-assisted object recognition system," in *ICC 2020 - 2020 IEEE International Conference on Communications*. Virtual Conference: IEEE, 2020, pp. 1–7.

[27] G. of Ireland, "Covid tracker," 2020. [Online]. Available: https://www2.hse.ie/conditions/coronavirus/testing/contact-tracing.html

[28] B. Perrigo, "U.s. states are rolling out covid-19 contact tracing apps. months of evidence from europe shows they're no silver bullet," 2020. [Online]. Available: https://time.com/5898559/covid-19-contact-tracing-apps-privacy/

[29] D. Zeinalipour-Yazti and C. Claramunt, "Covid-19 mobile contact tracing apps (mcta): A digital vaccine or a privacy demolition?" in *2020 21st IEEE International Conference on Mobile Data Management*. Versailles, France: IEEE, 2020, pp. 1–4.