SpreadMeNot: A Provably Secure and Privacy-Preserving Contact Tracing Protocol

Pietro Tedeschi, *Member, IEEE*, Spiridon Bakiras, *Member, IEEE*, and Roberto Di Pietro, *Senior Member, IEEE*

Abstract—A plethora of contact tracing apps have been developed and deployed in several countries around the world in the battle against Covid-19. However, people are rightfully concerned about the security and privacy risks of such applications. To address these issues, in this paper we provide two main contributions. First, we present an in-depth analysis of the security and privacy characteristics of the most prominent contact tracing protocols, under both passive and active adversaries. The results of our study indicate that all protocols are vulnerable to a variety of attacks, mainly due to the deterministic nature of the underlying cryptographic protocols. Our second contribution is the design and implementation of *SpreadMeNot*, a novel contact tracing protocol that can defend against most passive and active attacks, thus providing strong (provable) security and privacy guarantees that are necessary for such a sensitive application. Our detailed analysis, both formal and experimental, shows that *SpreadMeNot* satisfies security, privacy, and performance requirements, hence being an ideal candidate for building a contact tracing solution that can be adopted by the majority of the general public, as well as to serve as an open-source reference for further developments in the field.

Index Terms—Contact tracing, cryptography, privacy, protocols, security

1 Introduction

The sudden outbreak of the Covid-19 coronavirus has fundamentally changed our society. The high infection rate of the virus facilitated its rapid spread around the world that resulted in the most deadly pandemic in recent history [1]. Unfortunately, according to health experts, Covid-19 will almost certainly not be the last pandemic, so we, as a society, should be prepared to tackle future outbreaks more effectively and efficiently. To this end, contact tracing—followed by aggressive testing—has proven to be a very valuable tool in the battle against Covid-19 [2], [3]. In particular, contact tracing involves the early identification and notification of people that have been exposed to the virus by being in close proximity, for a given time in the near past, to a user that has tested positive [4], [5], [6].

- *Pietro Tedeschi is with Technology Innovation Institute, Secure Systems Research Center, Abu Dhabi 2022, United Arab Emirates. E-mail: pietro.tedeschi@tii.ae.*
- *Spiridon Bakiras is with the Infocomm Technology Cluster, Singapore Institute of Technology, Singapore 138683. E-mail: spiridon.bakiras@singaporetech.edu.sg.*
- *Roberto Di Pietro is with the Division of Information and Computing Technology (ICT), College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha 122104, Qatar. E-mail: rdipietro@hbku.edu.qa.*

Manuscript received 19 July 2021; revised 13 May 2022; accepted 22 June 2022. Date of publication 24 June 2022; date of current version 13 May 2023. This work was supported in part by Technology Innovation Institute, Abu Dhabi - UAE, in part under Grant NPRP-S-11-0109-180242, by QNRF-Qatar National Research Fund, a member of The Qatar Foundation, and in part by the NATO Science for Peace and Security Programme - MYP G5828 project "SeaSec: DronNets for Maritime Border and Port Security". (Corresponding author: Pietro Tedeschi.) Recommended for acceptance by S. Nepal. Digital Object Identifier no. 10.1109/TDSC.2022.3186153

Traditionally, contact tracing has been performed exclusively by public health professionals, in the form of interviews. However, relying solely on patient interviews is not a very effective approach. First, it requires an enormous workforce to trace potential infection chains when there are thousands of new cases discovered daily. Second, face-to-face or even phone interviews regarding social contacts may feel like an invasion of privacy to many people, who might refrain from sharing all the relevant information. Finally, close contact with complete strangers is a regular theme in our daily lives (public transit, supermarkets, shopping malls, etc.), and such infection chains cannot be tracked with manual contact tracing methods [7].

Alternatively, *digital* contact tracing is a technology that is gaining significant traction among governments and public health officials. In a nutshell, digital contact tracing is generally achieved via mobile applications that leverage either the Bluetooth Low Energy (BLE) protocol or the Global Navigation Satellite System (GNSS) technologies to keep track of other mobile devices in their vicinity [8]. More specifically, every device periodically broadcasts a random beacon that is intercepted and recorded by the nearby devices. If a beacon is detected to be closer than a certain threshold (e.g., 2 meters) for a significant amount of time, the event is recorded permanently in the *contact list* maintained by that device. When a user tests positive for Covid-19, the public health authorities are given access to their device in order to release their random beacons (and/or contact list) to a centralized database. Subsequently, other devices will download the released information and identify whether they have been in contact with the infected individual [9].

There have been a plethora of contact tracing apps implemented and used around the world. Singapore's TraceTogether app [10] was one of the first wide-scale deployments,

while many other big players have entered the arena, including the European Union [11], [12] and Apple/Google [13]. Nevertheless, people nowadays are very concerned about their privacy and may be unwilling to install and run a surveillance-type application [14], [15]. Further, the energy tool of such applications and the perceived battery life shortening could limit the adoption of contact tracing by the general public. A solution that on one hand offers provable security and privacy guarantees, while on the other hand introduces only a little overhead, is in dire need.

Contribution. To satisfy the above introduced requirements, we provide two main contributions. The first contribution of this work is an in-depth analysis of the security and privacy characteristics of the most prominent contact tracing protocols. We look at the fundamental mechanisms incorporated into these methods, including the type of information that is collected, the location where that information is stored (centralized versus decentralized), and the cryptographic primitives involved in the generation of the random beacons. We also consider a wide range of adversarial behavior, ranging from simple passive (eavesdropping) attacks to more elaborate and active ones, such as replay and relay attacks.

Our results indicate that, at the very least, all existing approaches are vulnerable to simple eavesdropping attacks that can de-anonymize an individual once they have tested positive for the coronavirus. Indeed, beacons are generated in a deterministic manner (e.g., using hashing) and are always transmitted in cleartext. As such, an adversary can easily intercept them with off-the-shelf antennas and tag them with time-stamped location information. Therefore, when users disclose their beacons (or contact lists), the adversary can track their location history.

Our second contribution aims to cope with the above introduced vulnerabilities. In details, we design *SpreadMeNot*, a novel contact tracing protocol with strong, provable privacy guarantees that protects the users' proximity data against sophisticated attacks. The main novelty of our protocol is the use of beacons that are generated leveraging standard, low-overhead, public-key cryptographic primitives. The probabilistic nature of public-key cryptography allows for the randomization of all previously transmitted beacons, so that they can be safely published without disclosing any identifiable information to an adversary. As such, *SpreadMeNot* is resilient to eavesdropping attacks. Furthermore, to strengthen our protocol against more powerful, active adversaries, we introduce a simple extension (based on digital signatures) that prevents an adversary from replaying previously transmitted beacons.

Finally, we demonstrate with experiments run on mobile phones that *SpreadMeNot*'s overhead in terms of computations and energy consumption is very reasonable for an average smartphone device. We also provide a formal verification in ProVerif—i.e., we generate an automated formal proof for the security properties guaranteed by *SpreadMeNot*. The verification source code is released as open-source [16].

Roadmap. The remainder of the paper is organized as follows. Section 2 introduces the technical background related to contact tracing technologies and public-key cryptography. Section 3 illustrates the generic contact tracing model and describes the adversarial behavior that we assume in this paper. Section 4 presents our in-depth study on the security and privacy of the current state-of-the-art solutions. Section 5 introduces the *SpreadMeNot* protocol and Section 6 investigates its performance in terms of computations and energy requirements. Section 7 highlights some important optimizations that mitigate the cost of public-key cryptography, and Section 8 concludes our paper.

2 Technical Background

In this section, we discuss briefly the wireless technologies that are employed by existing contact tracing protocols, including Bluetooth and satellite communications. We also introduce Elliptic Curve Cryptography (ECC), which is the underlying cryptographic primitive of *SpreadMeNot*.

2.1 Bluetooth

The wireless Bluetooth technology was conceived in the early 1990 s at Ericsson and was intended to replace the RS-232 data cable. Specifically, it allows fixed and mobile devices to exchange data over short distances using the Ultra High Frequency (UHF) spectrum. Bluetooth was first standardized as the IEEE 802.15.1 protocol, with the current standard maintained by the Bluetooth Special Interest Group (SIG).

Nowadays, there is a wide range of applications that adopt Bluetooth technology to manage wireless devices such as headphones, mice, keyboards, and printers. More importantly, Bluetooth is the underlying technology for numerous novel applications, such as indoor localization, Internet of Things (IoT), contact tracing, gaming, smart-locking, networking, and data streaming, to name a few. For example, by placing Bluetooth transceivers around large shopping areas, we can enhance user experience via Bluetooth-based mobile advertising applications. Note that, under the current standard, a master Bluetooth device can establish a one-to-one communication with a maximum of 7 devices in an ad-hoc network.

From the physical layer perspective, Bluetooth communications adopt the UHF frequency band and, in particular, the frequencies ranging from 2.402 GHz to 2.480 GHz. The modulation scheme is either Gaussian Frequency Shift Keying (GFSK) with a bit rate of 1 Mbits/sec, or Differential Phase Shift Keying (DPSK) with a maximum bit rate of 3 Mbits/sec. Furthermore, Bluetooth employs Frequency-Hopping Spread Spectrum (FHSS) and divides the spectrum into 79 channels, each with a bandwidth of 1 MHz. BLE accommodates 40 channels, each with a bandwidth of 2 MHz. The transmission range depends on the device class and its maximum transmission power, as shown in the Table 1.

The Bluetooth SIG has released several versions of the Bluetooth standard, each supporting backward compatibility. The latest release is the Bluetooth Core Specification version 5.2. Notice that, BLE was first introduced in version 4.0 and was further improved in versions 4.1 and 4.2. In this work, we consider the Bluetooth Core Specification version 4.2 standard. Under the cited standard, the modulation rate is 1 Mbits/sec and the format of the Bluetooth frame is depicted in Fig. 1.

Bluetooth v4.2 frames have an overall size of 2120 bits [17]. The message starts with the first 8 bits reserved to a *preamble* that is used to identify an upcoming Bluetooth message

TABLE 1
Bluetooth Device Class

Device Class	Max TX Power (dBm)	Range (m)
1	20	100
1.5	10	20
2	4	10
3	0	1
4	-3	0.5

and allow the receiver to synchronize with the symbols emitted by the transmitter. An *Access Address* of 32 bits denotes the correlation code tuned to the physical channel. The *Protocol Data Unit* (PDU) consists of 2056 bits, and is reserved for Data TX and Advertising. Finally, the last 24 bits are devoted to an error-detection code (CRC) and store the checksum of all bytes in the PDU. The bottom part of Fig. 1 shows the contents of the PDU. It includes the Bluetooth packet *Header* (16 bits), the link-layer *Payload* (up to 2008 bits of data), and 32 bits of a *Message Integrity Check* (MIC) [18].

Note that Bluetooth RF operations take place according to a slot-based Time Division Multiple Access (TDMA) schedule. Specifically, assuming a data rate of 1 Mbits/sec and a packet size of 265 bytes, the transfer duration is 2.12 ms.

2.2 Global Navigation Satellite System

Some proximity tracing solutions adopt GNSS technologies as a key element to approximate user location. Any smart device equipped with a GNSS module can receive RF signals originating from Medium Earth Orbit (MEO) satellites, located 19,000 to 23,000 km above Earth. Common GNSS devices are only able to receive in the Upper L-Band, i.e., in the 1.5 GHz range, with an accuracy of about $3-5$ meters for the positioning [19]. Each satellite is synchronized with atomic clocks and sends a navigation signal that contains information about the delivery time and the deviation from its expected trajectory.

Several GNSS technologies are available on the market, with the most popular one being Global Positioning System (GPS) that is operated by the United States Department of Defense. Other systems include the Russian GLONASS, the European GALILEO, and the Chinese BEIDOU. Nevertheless, GNSS technologies have some well-known weaknesses: they do not work in indoor environments, are extremely sensitive to jamming attacks, and are vulnerable to spoofing attacks if the RF signal is not encrypted [20].

GNSS technologies are widely adopted for several applications that span from location-based services to the monitoring and tracking of objects in remote areas. However, in the proximity tracing scenario, and compared to Bluetooth-based solutions, GNSS has the following drawbacks: (i) higher energy consumption due to the position sensors; (ii) lower accuracy in terms of proximity localization; and, (iii) an inability to work in close spaces, like buildings.

2.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography is a powerful approach to public-key cryptography that is based on the algebraic structure

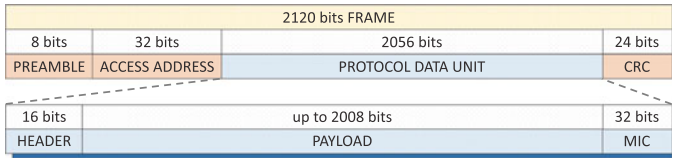

Fig. 1. Bluetooth v4.2 frame: Packet format (upper part) and protocol data unit (bottom part).

of elliptic curves over finite fields. ECC is considered as a better alternative to traditional schemes over finite fields because, for the same level of security, it offers much smaller keys and ciphertexts [21]. As such, ECC is more suitable for resource-constrained devices, such as smartphones or IoT devices. An elliptic curve defines a set of coordinates (x, y) that satisfy Eq. 1 below. In addition, a special point $\mathcal{O}$, namely point at infinity, represents the point at the ends of all lines parallel to the y-axis.

$$y^2 = x^3 + ax + b \tag{1}$$

The elliptic curve domain parameters over the finite field $\mathbb{F}_p$ are a sextuple $\mathcal{E} = (p, a, b, G, q, h)$, where: (i) $p > 3$ is an integer specifying the finite field $\mathbb{F}_p$; (ii) a, b are the elements that define the elliptic curve; (iii) $G \in \mathcal{E}(\mathbb{F}_p)$ is the generator point; (iv) prime q is the order of the subgroup generated by G; and, (v) integer h is the cofactor of the subgroup.

In ECC, the private key is generated by choosing a uniformly random number $x \in \mathbb{Z}_q^*$, i.e., a number in the interval $[1, q)$. The corresponding public key is then computed as $P = xG$. The security of ECC is based on the intractability of finding the discrete logarithm of a random elliptic curve point P with respect to a publicly known base point G. This is known as the Elliptic Curve Discrete Logarithm Problem (ECDLP). The bit-length n of the prime order q determines the security level of an ECC instantiation. We formally define the assumption behind the security of ECC schemes below.

Definition 1. *A function f is* negligible *if for every polynomial $p(\cdot)$ there exists N such that, for all integers $n > N$, it holds that $f(n) < \frac{1}{p(n)}$. We denote a negligible function of n as negl (n).*

Definition 2. *The* ECDLP assumption *is as follows: given $P, G \in \mathcal{E}(\mathbb{F}_p)$ where $P = xG$, $x \in \mathbb{Z}_q$, and $q \in \{0,1\}^n$ is the order of the elements P and G, a polynomial-time algorithm can output x with probability negl (n).*

3 System and Adversarial Model

In this section, we introduce the generic contact tracing environment assumed in our work (Section 3.1) and present the details of the underlying adversarial model (Section 3.2).

3.1 System Model

We assume a typical BLE-based contact tracing environment, as illustrated in Fig. 2. Users that have subscribed to the contact tracing system are constantly broadcasting ephemeral identifiers (or beacons[1]) that are randomly

1. Henceforth, we use the terms beacon and ephemeral ID interchangeably.

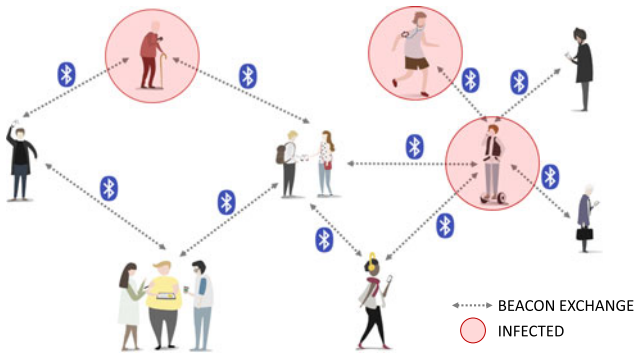

Fig. 2. BLE-based contact tracing.

generated according to *SpreadMeNot*'s protocol specifications. Additionally, all users in proximity to a broadcasted beacon will temporarily store it into their local memory.

A newly created beacon is valid for a fixed time window, in order to allow the surrounding devices to detect possible contagion events. In most existing applications, beacons are changing every 10 to 15 minutes, thus preventing malicious adversaries from tracking individual users. When a device receives the same beacon over a period of time that is considered significant by the public health authorities (and if the device is assumed to be sufficiently close to the beacon's source) the beacon is permanently stored in the *contact list* of the device. Based on the characteristics of the targeted virus, contact events can be safely erased after a few days (e.g., 14 days for COVID-19).

When a user is diagnosed as positive (e.g., the highlighted users in Fig. 2), the public health authorities will coordinate with the user to upload the stored contact list to their centralized database. The remaining users will periodically download the new contact lists from the database and check whether their own beacons are contained therein. If this is true, the user will be notified about the verified contagion event and will be given instructions for further actions.

3.2 Adversarial Model

The adversary assumed in our work is very powerful and may perform both passive and active attacks. We assume that the adversary is equipped with a powerful antenna, which can be either a regular Bluetooth handheld device, or a Software Defined Radio (SDR) that is operated through a laptop/smartphone running an SDR-compatible software tool, such as GNURadio [22].

3.2.1 Passive Attacks

Eavesdropping. We assume that the adversary is a global eavesdropper, able to detect and decode any message broadcasted on the Bluetooth communication channel. This capability allows the adversary to obtain all the information within the exchanged Bluetooth beacons, including the "rolling" standard ID adopted by Bluetooth technology and the ephemeral IDs of the underlying contact tracing protocol. In addition, the adversary locally enriches the intercepted beacon with appropriate metadata, such as time and location information. The adversary could hence conduct a cross-referencing analysis with data collected from various sources, including the "infected" beacons published by the public health authorities, background information on individual users, direct observation of user movements, etc. The goals of an eavesdropping attack can be manifold. For example, the adversary may try to de-anonymize users that have tested positive for the virus, track user locations over time, or identify social relationships among users.

3.2.2 Active Attacks

Replay. We also assume that the adversary has Bluetooth transmission capabilities. Thus, it can replay beacons previously acquired (via eavesdropping) from the Bluetooth communication channel. These beacons contain the ephemeral IDs of legitimate users and are, therefore, indistinguishable from legitimate beacons when processed by the contact tracing app. The goal of the adversary is to inject false contact events into the users' contact lists, hence increasing the chances for the user to falsely being considered a contagion risk—consequences could be, for instance, a mandatory, legally binding, period of quarantine for the attack's victim.

Relay. This attack aims at bypassing replay attack countermeasures that may incorporate timing information within the beacons, thus limiting the time window where beacons can be replayed. It can be thought of as an amplified version of a replay attack, where the adversary instantaneously disseminates every captured beacon to multiple locations under its control. The beacons are then replayed immediately within each of those locations.

The reason why replay/relay attacks are dangerous in the context of contact tracing is that they can be used to manipulate the recorded contact lists. Specifically, an adversary can install a powerful antenna at a busy location (e.g., a shopping mall) and start eavesdropping on the transmitted beacons. Then, it replays those beacons with a high transmit power so that they register as legitimate contacts to a large number of users (even if they are further away from the antenna). As a result, if one of these users has a positive diagnosis, there will be an overwhelming amount of false negative alerts that would saturate test centers (if suspect-positive users are tested) or likely result in mandatory quarantine if testing is not possible—hence imposing an unnecessary privation of personal freedom.

Social Graph. This attack allows an adversary to reconstruct a user's social interactions by leveraging the data transmitted among smart devices.

Note that, in this work, we do not address Denial of Service (DoS) attacks that may be launched by an adversary to disrupt the operation of the contact tracing network. For example, the adversary might try to jam the Bluetooth communication channel or compromise the availability of the centralized database. We also assume that the disclosure of contact lists from infected users is done securely by the public health authorities. In other words, an adversary is not able to inject false data into the centralized database. Such attacks are independent from the underlying contact tracing protocol, and hence considered out of scope with respect to our contributions.

4 Security and Privacy Analysis of Existing Solutions

In this section, we present an in-depth analysis of the security and privacy characteristics of the current state-of-the-

art contact tracing protocols. We focus our discussion on the following metrics.

Architecture (C-D). In a centralized (C) architecture, all mobile devices forward their proximity data to a centralized database that is administered by the public health authorities. Conversely, in a decentralized (D) approach, the mobile devices do not share their data with the authorities, unless they test positive for the virus. Furthermore, the contact tracing operation is performed in a fully decentralized manner at the individual devices. Finally, in a hybrid (C/D) architecture, data collection follows the decentralized approach (nothing is disclosed to the authorities), while the contact tracing operation is performed at a centralized location—by having the infected individuals reveal their entire contact history to the health authorities. Clearly, the centralized (and to a lesser extent the hybrid) architecture requires users to fully trust the authorities from a security and privacy perspective. Clearly, the centralized architecture requires users to fully trust the authorities from a security and privacy perspective.

Privacy From Authority. This is a measure of the privacy that users enjoy with regards to the information that is disclosed to the public health authorities by the contact tracing application. For instance, in the absence of a fully decentralized architecture, a central authority can harm the privacy of the users by collecting sensitive data (e.g., GPS locations, contact lists) and/or performing statistical inference on the acquired data.

Linkage Attack Resilience. A linkage attack attempts to identify users by analyzing an anonymized dataset, while cross-referencing it with external data sources, such as user-specific information, geolocation data, etc. [23]. For instance, the beacons released to the authorities by a user that had a positive test can potentially reveal his/her identity if the beacons can be linked to precise geographic locations.

Replay/Relay Attack Mitigation. As explained previously, replay and relay attacks can be very damaging to the effectiveness of contact tracing. Therefore, contact tracing protocols must incorporate appropriate cryptographic mechanisms to limit their scope or, if possible, eliminate them altogether.

Robustness Against Eavesdropping. This metric quantifies the usefulness of the beacons that are collected by an eavesdropping adversary. For example, if users upload their previously transmitted beacons to a public database (when deemed infected), an adversary can leverage them to deanonymize the users.

Robustness Against DoS Attacks. The aim of a DoS attack is to exhaust the system's resources, such as memory and network bandwidth, in order to compromise its availability.

Ephemeral Identity. An ephemeral identity is the temporary ID associated with the real user identity and is contained inside the transmitted beacons. This is the minimum privacy guarantee that all contact tracing protocols should provide. Additionally, ephemeral IDs should change frequently, to prevent the tracking of user movements.

Resilience against Social Graph Attacks. A social graph attack aims to compromise the user's privacy by re-building the connection graph with the surrounding devices.

Exposure of Geolocation Data. Some contact tracing protocols collect GPS data to perform the proximity testing operation. As such, when users are diagnosed as positive, their recent trajectories are disclosed to the public health authorities and, subsequently, to the general public. The cited approach is less private than BLE-based contact tracing, because it does not require from the adversary any effort to learn the detailed trajectory data.

Open Source. The protocol and the code are released as open source to facilitate thorough security and privacy audits from both the industry and academia.

4.1 Existing Solutions

Decentralized Privacy-Preserving Proximity Tracing (DP-3T) [12]. The DP-3T protocol is the product of a large consortium of European universities and research institutes. It is a fully decentralized protocol that leverages BLE beacons to exchange ephemeral IDs among the participating devices. As reported by Vaudenay [24], the ephemeral identifiers are generated with an `HMAC–SHA256` key, encrypted with `AES–CTR` or `Salsa20`. Further, these keys are kept in the device's memory and erased after they are no longer required by the health authorities (e.g., after 14 days). The keys are derived from a root key that is randomly generated every day. The analysis of Vaudenay highlights several security and privacy issues, including a replay attack which is possible due to the absence of message authentication codes. When a user is diagnosed as positive, DP-3T offers several options for disclosing their own beacons to the health authorities. The least private approach discloses all the past daily keys, while the more privacy-preserving option allows the user to choose which beacons to reveal.

Apple/Google [13]. Apple and Google jointly designed and implemented a decentralized proximity tracing protocol that is very similar to DP-3T. Unlike other approaches, this protocol is fully integrated within the operating system, and exposes appropriate APIs to third-party developers in order to develop their own smartphone applications. From a privacy perspective, Apple and Google have not released their source code, so users must trust these companies (and their operating systems) with respect to data analysis [23]. With regards to the cryptographic specifications, the key schedule for contact tracing is organized into 3 main phases: (i) tracing key generation, (ii) daily tracing key generation, and (iii) rolling proximity identifier generation. The tracing key generation procedure is executed when contact tracing is first enabled on the mobile device. The 32-byte tracing key is derived via a Cryptographic Random Number Generator and is stored on the device. From this tracing key, an HMAC Key Derivation Function (HKDF), such as `HMAC–SHA256`, is employed to derive a 16-byte daily tracing key that is valid for a 24-hour time window, based on Unix epoch time. Finally, the rolling proximity identifiers exchanged within the BLE beacons are 16-byte ephemeral IDs, derived from the daily tracing key through an `HMAC–SHA256` operation [25].

Berke et al. *[26].* This is a GPS-based solution that also adheres to the decentralized architecture. Specifically, the mobile devices are constantly monitoring the users' location, by recording their GPS coordinates every t minutes. Therefore, each point in the user's location history has three dimensions, namely longitude, latitude, and time. When a

user is diagnosed as infected, their location history is uploaded to the health authorities' server, thus allowing other devices to download it and look for possible contagion events. To preserve privacy, precise GPS coordinates are replaced by larger geographic areas, and the resulting point-intervals are obfuscated with a hash function like SHA256. In addition, the authors propose a technique where the server and individual clients invoke a private set intersection protocol to identify contagion events, without disclosing the patient's location history. In terms of security, an adversary can either broadcast fake GPS signals [27] (GPS spoofing) or perform a GPS jamming attack [28], in order to disrupt the proximity tracing operation.

Hamagen [29]. Hamagen is a COVID-19 exposure prevention app that was developed by Israel's Ministry of Health. It is fully decentralized and leverages GPS measurements to identify possible COVID-19 exposure events. Once an hour, the installed application downloads a file with an anonymous list of locations, times, and dates that confirmed (from the Ministry) patients have visited in the past. The app will then perform a cross-reference on the (timestamped) locations visited by the user and determine whether the user is at risk of exposure. Note that, the application exploits GPS, Bluetooth, and WiFi to improve the accuracy of geolocation. Additionally, it does not rely on any cryptographic primitives.

Corona 100m [30]. This is South Korea's centralized and GPS-based solution that publicly informs citizens (and the health authorities) of known cases within 100 meters of where they are. It is worth noticing that this application cross-references credit card data, medical records from hospitals, social networks graphs, and video surveillance footage, thus raising serious privacy concerns among the citizens.

Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) [11]. The PEPP-PT protocol is the European Union's solution to digital contact tracing. PEPP-PT is a centralized approach based on BLE technology and is compliant with the EU's General Data Protection Regulation (GDPR) privacy laws. Unlike DP-3T and Apple/Google, PEPP-PT mandates that the users' BLE beacons (ephemeral IDs) are generated and distributed by the public health authorities. Specifically, during registration, each user is assigned a random ID by the system. Then, for each contact tracing epoch t, the system selects a global secret key K_t and encrypts (using AES) all user IDs with the same key. The resulting ciphertexts are the ephemeral IDs that are distributed in advance to the individual users. When testing positive for the coronavirus, users release their logged contact lists to the authorities, who are then able to perform the contact tracing operation at the centralized server, since they can identify the IDs of all contacts by decrypting the beacons with the secret keys. Clearly, this approach is less privacy-preserving than the decentralized solutions, because it allows anyone who owns the AES keys to track all registered citizens.

Temporary Contact Numbers (TCN) [31]. TCN is a fully decentralized protocol proposed by the Coalition Network. It leverages BLE communications and its operation is very similar to DP-3T and Apple/Google. In other words, mobile devices generate their own ephemeral IDs that are broadcasted to other devices in their proximity. Each device gradually builds its own private contact list that is not shared with the authorities. After a positive diagnosis, users upload their keys to the centralized database, in order to allow other devices to reconstruct the corresponding ephemeral IDs and measure their exposure risk. The main difference in TCN is that the device generates a public and private key pair (based on Ed25519 curves) so that other devices can verify the authenticity of the reconstructed ephemeral IDs. In particular, the public-key is used as an input to the SHA256 hash function that generates the ephemeral IDs, while the private key is used to sign the report that is sent to the health authorities after the user is diagnosed as positive.

BlueTrace [32]. BlueTrace was one of the first protocols to enjoy a wide-scale deployment since it was the fundamental building block in Singapore's TraceTogether app. Its operation is quite similar to PEPP-PT, i.e., a centralized approach where the ephemeral IDs are generated by the public health authorities. As such, the contact tracing operation is performed in a centralized manner by the health authorities, after the positively diagnosed individuals upload their contact logs to the centralized server. The system maintains a global secret key for the generation of the ephemeral IDs and, during registration, every user is assigned a unique ID. Then, the per-user ephemeral IDs are generated by encrypting a message—comprising the user ID, the validity period (start and expiration), an initialization vector, and an authentication tag—with an AES256–GCM cipher and Base64 encoding the resulting ciphertext. To minimize the effect of replay attacks, the validity period of each ephemeral ID is set to 15 minutes [32].

Private Automated Contact Tracing (PACT) [33]. PACT is another contact tracing protocol that follows the decentralized approach of DP-3T and Apple/Google. In particular, every device generates its own ephemeral IDs through a Pseudorandom Function (PRF) that takes as input a random 256-bit seed (which changes every hour) and the current time measured at one-minute precision. Furthermore, a newly diagnosed patient will upload all the random seeds from the recent past to the centralized database, in order to allow other users to reconstruct the ephemeral IDs and estimate their exposure risk. Note that, similar to other approaches, PACT reduces the risk of replay attacks by including timestamps into the generation of the ephemeral IDs.

Whisper [34]. Whisper is the only proximity tracing protocol thus far in the literature that employs public-key cryptography. It is a fully distributed protocol that leverages BLE communications to monitor and log contact events. Specifically, every device periodically generates a fresh public/private key pair, based on Ed25519 elliptic curves. Additionally, unlike other protocols, Whisper operates in two phases. In the first phase, the device periodically scans the BLE channel to detect other Whisper devices in its proximity. For each discovered device, the protocol initiates a connection (phase two), during which the two devices perform a Diffie-Hellman key exchange to generate a shared key K. For each key K, the device computes two hash digests using the BLAKE2–160 hash function: the *tell-token* is the hash digest of K and the device's own public-key, while the *hear-token* is the hash digest of K and the connecting peer's public-key. After a positive diagnosis, the device uploads its own tell-tokens to the centralized database,

which enables other devices to compare them against their own hear-tokens. The major advantage of Whisper in terms of privacy is that the published tell-tokens do not reveal any information to an adversary, because it is infeasible to compute the underlying shared keys.

IoTrace [7]. IoTrace is a novel IoT-enabled approach to contact tracing. Specifically, in IoTrace, mobile devices neither receive the broadcasted beacons nor maintain individual contact lists. Instead, the deployed IoT devices collect every beacon that is transmitted around them and send all data to a centralized server. When a user tests positive for the coronavirus, the mobile device releases its transmitted beacons to the authorities who then publish all the beacons that are considered in close proximity to the user. The remaining users periodically download the updated database and perform the exposure notification function in a distributed manner. The key characteristic of IoTrace is the reduced energy cost on mobile devices, due to the absence of frequent scanning operations to identify transmitted beacons. However, in this contribution, we only focus on peer-to-peer contact tracing that is performed without the use of an existing IoT infrastructure.

ConTra Corona [35]. In this contribution, the authors propose a simulation-based security notion via an ideal contact tracing protocol. The anonymity property of the user is achieved by adopting ephemeral identifiers (generated with a public key algorithm starting from a real identifier) that are uploaded to a central server in an anonymous way. Further, ConTra Corona adopts a secret sharing scheme for the ephemeral identifiers to reconstruct the original identifier of the exposed user.

DESIRE [36]. This protocol is a hybrid between the centralized and decentralized models. More specifically, the risk exposure and notification services are managed by a central server. DESIRE relies on Private Encounter Tokens (PETs), which are privately generated by users and are, thus, unlinkable (i.e., the schema is robust against social graph attacks). PETs are computed via a standard Diffie-Hellman key exchange protocol. The data stored on the server (and on the smartphone devices) are encrypted with cryptographic keys.

DIMY [37]. DIMY or "Did I Meet You" is a new privacy-preserving digital contact tracing protocol that employs several well-known primitives, including Diffie-Hellman key exchange, Shamir secret sharing, Bloom filters, and blockchain. First, the Diffie-Hellman key exchange is used to establish a secret contact ID between two devices. In addition, during this data exchange, the protocol leverages Shamir's secret sharing scheme in order to provide data privacy and secure the communication channel. Next, the contact IDs are stored into a Bloom filter and, if a user tests positive for the coronavirus, a single aggregate Bloom filter is uploaded on a blockchain-based back-end. Users query the blockchain for exposure by submitting their own aggregate Bloom filters. The authors demonstrate that DIMY is resilient against common attacks, such as linkage, enumeration, social graph construction, and replay. However, the protocol is still affected by some security issues such as:

1) Bloom filters are set membership structures, so they allow for efficient set membership queries. Any adversary that has stored the plaintext contact IDs that it has observed, can easily identify them within a given Bloom filter.
2) Bloom filters are probabilistic data structures that can result in false positives. In the context of contact tracing, false positives are not desirable.
3) The accuracy of Bloom filters can be compromised by transmitting a large number of fake contacts. If a Bloom filter is overloaded (the number of contacts exceeds its designed capacity), the false positive rate will become significant.

A qualitative survey of the inherent privacy limitations of contact racing applications is provided in [38], whereas a quantitative comparison among different solutions and applications for contact tracing, such as PEPP-PT [11], DP-3T [12], Apple/Google [13], Hamagen [29] and BlueTrace [32] is presented in [7]. This latter study focused on the adopted wireless technology, the deployed architecture, the RF energy consumption, and the security and privacy aspects of elements such as location and health status. The analysis is further enriched by taking into account the storage requirements and computational costs incurred by the adoption of given cryptographic primitives. Further, the authors experimentally assessed the performance of considered protocols on the Bluetooth SoC nRF51822 and the GPS SiP nRF9160 (for Hamagen) hardware platforms.

4.2 Security and Privacy Analysis

In this section, we discuss the security and privacy characteristics of the aforementioned protocols. Our results are summarized in Table 4.2.

GNSS Solutions. GNSS-based protocols rely on geolocation data to perform proximity tracing. As such, there are several concerns regarding their accuracy, security, and privacy. First, it is worth noting that, with the exception of Corona 100 m which is a centralized approach, GNSS protocols offer perfect privacy to all citizens that do not contract COVID-19. This is because mobile devices do not send any information to the health authorities. However, for the individuals that test positive, these protocols reveal their (partial or full) trajectories over an extended period of time to the health authorities. As such, they are prone to linkage attacks that may cross-reference the published geolocation data with background information about the users. Even though the protocol by Berke *et al.* employs hashing to obfuscate the trajectories, it is still vulnerable to brute-force attacks that can easily de-anonymize the locations.

In terms of security, GNSS solutions are not susceptible to replay/relay and eavesdropping attacks, because they do not broadcast any data on the wireless channel [39]. On the other hand, GNSS signals can be spoofed by a malicious adversary, due to the lack of encryption/authentication in consumer GNSS products. Such attacks may allow an adversary to manipulate the GPS measurements that are collected by the contact tracing application. Another limitation of GNSS-based contact tracing is that generic devices such as smartphones, are provided with a recreational grade (accuracy within $\pm 7.6\ m$) GNSS receiver. This is not sufficient to determine "real" contagion events and may

Comparison of our solution against the state-of-the-art approaches. The ✓ symbol indicates the fulfillment of a particular feature, the ✗ symbol denotes that the related feature is missing, while the "−" symbol indicates that the feature is not applicable.

Features	DP-3T [12], A/G [13]	Berke *et al.* [26], Hamagen [29]	Corona 100m [30]	TCN [31]	BlueTrace [32], PEPP-PT [11]	PACT [33]	Whisper [34]	DIMY [37]	ConTra Corona [35]	DESIRE [36]	IoTrace [7]	*SpreadMeNot*
Architecture (C/D)	D	D	C	D	C	D	D	C/D	C/D	C/D	C/D	D
Wireless Technology	BLE	GPS	GPS	BLE	BLE	BLE	BLE	BLE	BLE	BLE	BLE	BLE
Privacy from Authority	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓
Linkage Attack Resilience	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
Replay Attack Mitigation	✗	✓	–	✗	✗	✓	✓	✓	✗	✓	✓	✓
Relay Attack Mitigation	✗	✓	–	✗	✗	✗	✓	✗	✗	✗	✗	✓
Robustness against Eavesdropping	✓	–	–	✓	✗	✗	✓	✓	✓	✓	✓	✓
Robustness against DoS	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Ephemeral Identity	✓	–	–	✓	✓	✓	✓	✓	✓	✓	✓	✓
Full Social-Graph Resilience	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓
Exposure of Geolocation Data	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
Open Source	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓
Crypto Primitive(s)	AES, SHA	SHA	–	SHA	AES	SHA	ECC, BLAKE2	ECC	–	ECC, SHA	SHA, ECC	SHA, ECC

result in an overwhelming amount of false-negatives. Additionally, in an urban/sub-urban scenario, the shadowing and multipath fading effects further undermine the geolocation accuracy. Finally, the risk of COVID-19 exposure is significantly higher in indoor environments where, unfortunately, GNSS localization is infeasible.

Centralized Solutions. PEPP-PT, BlueTrace and Corona 100 m are centralized protocol appearing in our literature review. This is not surprising, since centralized approaches come with significant security and privacy concerns. For example, in the case of Corona 100 m, the app continuously reports its location to the public health authorities and is notified whether a confirmed case is within 100 m of that location. As a result, every citizen that is actively using the app can be tracked by the authorities. Consequently, Corona 100 m offers no privacy from the authority and it is also vulnerable to linkage attacks, because geolocation data can be cross-referenced with external sources to reveal sensitive user information. Even in an ideal world where the authorities are honest and trusted by the general public, the vast amount of valuable data stored at the centralized database serves as an open invitation to malicious hackers. Further, both PEPP-PT and BlueTrace incorporate timestamped data into the generated ephemeral IDs, which limits the effect of replay attacks (although the timing information is relatively coarse). However, both the cited protocols fail to protect against relay attacks since they do not consider geolocation data.

Decentralized Solutions. Here we focus on BLE-based protocols that utilize symmetric key algorithms, including DP-3T, Apple/Google, TCN, PACT. As explained previously, their algorithms are quite similar and, therefore, they share the same security and privacy characteristics. First, similar to the decentralized GNSS-based protocols (Hamagen and Berk *et al.*), they offer perfect privacy to the vast majority of users who never contract the coronavirus. Furthermore, the beacons collected by an eavesdropping adversary cannot be used to de-anonymize them, because they never disclose their own beacons to the health authorities. Conversely, positively diagnosed users do not enjoy any privacy, because the disclosed ephemeral IDs can place them at precise locations by an eavesdropping adversary (or regular users who may trace those beacons inside their stored contact logs). As a result, decentralized protocols are not resilient against linkage attacks. In terms of replay attacks, all protocols use time as an input to the beacon generation process, thus reducing the risk of contact log manipulation by an active adversary. Note, however, that time is approximated in the scale of minutes, so there is still a sufficiently large time window that attackers can exploit. On the other hand, none of the aforementioned protocols utilize geolocation and they are, therefore, vulnerable to relay attacks.

A notable advantage of BLE technology with regards to localization is its accuracy, which makes it an ideal candidate for effective contact tracing. Specifically, every transmitted beacon contains information about the time of arrival and the Received Signal Strength (RSS). The latter is a key parameter for wireless positioning and aids in deriving the distance between two Bluetooth enabled devices, by leveraging a radio propagation model [40]. However, from a privacy perspective, it is crucial that the device's operating system implements a rolling MAC address protocol (e.g., by randomizing its last 3 bytes), in order to prevent the tracking of individual users. It is also recommended that the schedule of the ephemeral IDs follows that of the MAC

address randomization. A final remark concerns the security of the database server. Even though contact tracing is performed in a decentralized manner, the server plays a crucial role as a proxy for the distribution of the "infected" beacons. As such, the security of the health authorities' infrastructure is of paramount importance, even in the decentralized architecture [41].

Public-key Solutions. Whisper is one of the first contact tracing protocols in the literature that employs public-key cryptography. Specifically, it allows any two mobile devices to exchange a secret contact ID (tell-token and hear-token) that is infeasible to compute without knowledge of the underlying public keys. As a result, when diagnosed users disclose their own tell-tokens to the centralized server, the authorities cannot map them to precise locations. In other words, Whisper offers excellent privacy from authority and is immune to eavesdropping attacks. However, the tokens that are stored by the mobile devices may contain timing/location information (e.g., using a malicious application that adds such metadata), so individual mobile devices may be able to launch linkage attacks against the published tokens that are present in their memory. Another limitation of Whisper is that it does not employ any mechanism to mitigate replay/relay attacks.

Furthermore, Whisper is a complex protocol that does not rely simply on broadcasted beacons. Instead, to compute a new pair of tokens, the two devices have to establish a peer-to-peer connection and perform a Diffie-Hellman key exchange. Alternatively, other Diffie-Hellman based protocols, such as ConTra Corona, DIMY, and DESIRE, do not require explicit peer-to-peer connections. However, these protocols also generate deterministic contact IDs (through the key exchange protocol) and are vulnerable to linkage attacks as well.

On the contrary, *SpreadMeNot* does not employ a key exchange protocol, but rather utilizes public key cryptography to generate beacons that are re-randomizable. This beacon re-randomization is the unique property that differentiates *SpreadMeNot* from all existing contact tracing protocols. As such, it avoids the pitfalls of Diffie-Hellman based solutions and is an excellent candidate for secure and privacy-preserving contact tracing.

In order to protect the privacy of users who upload their beacons to the centralized database after a positive test (via the use of a token provided by the health authorities), the authors in [42] proposed a solution that generates anonymous tokens. As such, their methods can be used independently by any existing contact tracing protocol, including ours. However, the anonymous tokens are only applicable to this specific phase of contact tracing (beacon upload) and cannot protect against eavesdropping or replay/relay attacks.

5 The SpreadMeNot Protocol

In this section, we describe in detail our proposed solution for privacy-preserving contact tracing. In Section 5.1 we give a brief overview of the scheme and present the cryptographic experiment that will be the basis of our security proof. In Section 5.2 we introduce our construction and formally prove its security. In Section 5.3 we analyze the privacy properties of our protocol in comparison to the existing state-of-the-art approaches and, finally, in Section 5.4, we provide a formal verification of the security properties achieved by *SpreadMeNot*.

5.1 Overview

The *SpreadMeNot* protocol consists of a tuple of algorithms $\Pi = (\texttt{GenKey}, \texttt{GenBeacon}, \texttt{RandBeacon}, \texttt{Test})$ that operate as follows.

1) `GenKey`: It takes as input a security parameter n and outputs a private key x and a public-key P.
2) `GenBeacon`: It takes as input a public-key P and outputs a beacon $\mathcal{C}$.
3) `RandBeacon`: It takes as input a beacon $\mathcal{C}$ and outputs a randomized version $\mathcal{C}'$.
4) `Test`: It takes as input a beacon $\mathcal{C}$ and a private key x, and outputs *true* if the beacon is generated under public-key P.

Initially, every user executes the `GenKey` algorithm to generate their public/private key pair. However, in this setting, the public-key is irrelevant to the rest of the users and does not need to be shared (but it does not have to be secret). At regular intervals (e.g., every 15 minutes), the user's app will invoke the `GenBeacon` algorithm to generate a *fresh* random beacon. That beacon will be broadcast continuously until the next interval, in order for the surrounding devices to detect the user as a potential contact. Note that only *significant* events are stored on each device, i.e., beacons that are very close to the user (say, within 2 meters) for a sufficiently long time duration. Based on the characteristics of the virus, old contact events (e.g., older than two weeks) are automatically erased.

If a user tests positive for COVID-19, he publishes his contact list on the centralized database managed by the public health authorities. The list consists of all the beacons that are currently stored on his device. More importantly, the published list is first *permuted* and all its entries are randomized. Specifically, for each stored beacon $\mathcal{C}$, algorithm `RandBeacon` is invoked to produce a new beacon $\mathcal{C}'$ that is indistinguishable from $\mathcal{C}$. Finally, the published contact list is downloaded by all mobile devices that subsequently apply algorithm `Test` on every beacon. If any instance of `Test` outputs *true*, the user infers that he has been in close proximity to the infected individual.

In terms of security, we want beacons generated by different public keys to be indistinguishable from each other. Therefore, we define an eavesdropping adversary $\mathcal{A}$ that has access to all transmitted and (randomized) published beacons. The objective of $\mathcal{A}$ is to distinguish beacons that are generated from a specific public-key P. Experiment $\texttt{Exp}^{\texttt{eav}}_{\mathcal{A},\Pi}(n)$, as shown in Fig. 3, formulates the security of our protocol $\Pi = (\texttt{GenKey}, \texttt{GenBeacon}, \texttt{RandBeacon}, \texttt{Test})$ with security parameter n, under an eavesdropping adversary $\mathcal{A}$. We say that $\mathcal{A}$ succeeds in this experiment if

$$\Pr[\texttt{Exp}^{\texttt{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \texttt{negl}(n)$$

Conversely, our scheme is secure if $\mathcal{A}$ fails in this experiment.

5.2 Construction and Security Proof

Assume an elliptic curve group $\mathbb{G}$ of prime order q and let G be a generator of $\mathbb{G}$. This information is public and shared by all users. Also, n is the security parameter that determines the precise construction of the group $\mathbb{G}$. Our protocol Π is instantiated as follows.

1) `GenKey`: On input a security parameter n, choose a uniformly random private key $x \in \mathbb{Z}_q^*$ and set the public-key $P = xG$.
2) `GenBeacon`: On input a public-key P, choose a uniformly random $r \in \mathbb{Z}_q^*$ and output beacon $\langle rG, rP\rangle$.
3) `RandBeacon`: On input a beacon $\langle rG, rP\rangle$, choose a uniformly random $r' \in \mathbb{Z}_q^*$ and output $\langle r'rG, r'rP\rangle = \langle r''G, r''P\rangle$.
4) `Test`: On input a beacon $\langle rG, rP\rangle$ and a private key x, compute $A = xrG$. If $A = rP$, output *true*; otherwise, output *false*.

It can be easily verified that the protocol is *correct*, i.e., if algorithm `Test` returns *true*, the input beacon is generated under public-key P, since $xG = P$. The following theorem proves the security of our scheme.

Theorem 1. *There is no polynomial-time adversary $\mathcal{A}$ that succeeds in* $\mathsf{Exp}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$.

Proof. The only information that the adversary has is a beacon $\langle rG, rP\rangle$. If $\mathcal{A}$ succeeds in the experiment, it means that it can distinguish between $rx_0\ G$ and $rx_1\ G$ with non-negligible probability. Note that, it is infeasible to derive any of the secret values r, x_0, x_1, due to the ECDLP assumption. Since r, x_0, x_1 are uniformly random in $\mathbb{Z}_q^*$, both $rx_0\ G$ and $rx_1\ G$ would appear as random elements in $\mathbb{G}$. The probability of distinguishing between any two random group elements is negligible in $\log q$, where q is the order of $\mathbb{G}$ and $\log q$ is the order of the security parameter n. Since $\mathcal{A}$ is polynomial-time, the experiment succeeds with probability $\mathtt{negl}(n)$ larger than a random guess, which concludes our proof. □

The downside of using public-key cryptography is that it facilitates impersonation attacks. That is, if an adversary has knowledge of a user's public key, he can trivially generate valid beacons on his behalf. Even worse, knowledge of a user's public key is not really necessary; instead, the adversary may intercept a single beacon and then use algorithm `RandBeacon` to generate new, valid beacons at will. In addition to impersonation attacks, it is also essential to protect against other active attacks, including replay, relay, and social graph attacks.

To this end, we propose a simple solution—based on timestamps, localization data, and digital signatures—that mitigates existing active attacks. First, we assume that the users' mobile devices are synchronized within a few seconds, and every device knows its approximate location (longitude and latitude). Then, every beacon is transmitted with an attached Unix timestamp, which represents the current time, and the user's current coordinates. In addition, the beacon includes a digital signature that is computed as follows.

1) Let $\langle rG, rP\rangle$ be the current beacon, let T be the corresponding Unix timestamp, and let L be the user's current location.

Beacon Indistinguishability
Experiment $\mathsf{Exp}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$

1) On input a security parameter n, GenKey generates two private keys x_0, x_1 and the corresponding public keys P_0, P_1; the public keys are given to $\mathcal{A}$.
2) $\mathcal{A}$ calls GenBeacon an arbitrary number of times.
3) The challenger selects a random bit $b \leftarrow_{\$} \{0, 1\}$.
4) If $b = 0$, the challenger uses GenBeacon to generate a beacon $\mathcal{C}_0$, under public-key P_0. If $b = 1$, the challenger generates $\mathcal{C}_1$ under public-key P_1.
5) $\mathcal{A}$ continues calling GenBeacon and eventually outputs a bit b'.
6) Return 1 if $b = b'$ and 0 otherwise.

Fig. 3. Beacon indistinguishability experiment.

2) Let $d = rx \bmod q$ be an ephemeral private key associated with the ephemeral public-key $P' = rP = rxG$, i.e., the second term of the beacon. Note that, this ephemeral key pair is derived from the user's permanent private key x.
3) Use the ECDSA algorithm to generate the signature σ of message $L||T||rG||rP$, under private key d.
4) Broadcast beacon $\mathcal{C} = (L, T, rG, rP, \sigma)$.

At the receiver side, the device will first verify signature σ, using public-key $P' = rP$. Furthermore, it will extract the timestamp T and verify that it is within the synchronization threshold $\mathcal{T}_{thr}$. Finally, it will verify that L is within a distance threshold $\mathcal{D}_{thr}$ from the device's current location. If any of the above verifications fail, beacon $\mathcal{C}$ is rejected. It is worth emphasizing that the public-key that generates the signature changes whenever a fresh beacon is generated. As such, users cannot be tracked by linking successive beacons to the same device, and any attempt from an adversary to re-use that beacon (or a randomized version of it) at subsequent timestamps (or remote locations) will fail.

We should stress that the signature verification key is part of the beacon itself (i.e., they are bound), so an attacker cannot replace the signature with its own. If the attacker replaces the signature, it also has to replace the ephemeral public key rP with another key $r'P'$ for which it knows the underlying private key. While this is sufficient to pass signature verification, it results in a useless beacon that will not generate any positive matches for the original beacon, since $rxG \neq r'P'$.

Note that, to save valuable resources, it is not necessary to verify a signature as soon as it is received. Recall that a beacon is only stored in the contact list if it has been received multiple times over a sufficiently long time interval. Therefore, the device can simply defer the signature verification process until a beacon is inserted into the contact list. Algorithm 26 summarizes *SpreadMeNot*'s detection mechanism against replay and relay attacks. In this pseudcode, $\mathcal{H}$ is a generic hash function, such as `SHA-256`.

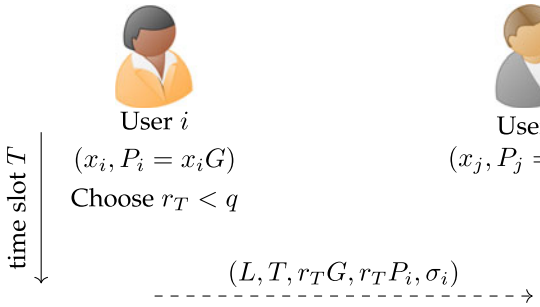

Fig. 4. Sequence diagram of the *SpreadMeNot* scheme.

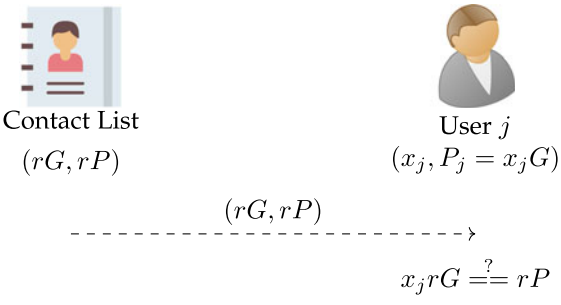

Fig. 5. *SpreadMeNot* check contact list procedure.

Algorithm 1. Replay/Relay Attack Detection Mechanism

Input: BLE Beacon $\mathcal{C} = (L, T, rG, rP, \sigma)$;
Result: Outcome: $\{\perp, \mathcal{A}_p, \mathcal{A}_l\}$;
Select $\mathcal{C}$;
// Signature Verification
$rP \leftarrow$ *extract_public_key*($\mathcal{C}$);
$\sigma \leftarrow$ *extract_signature*($\mathcal{C}$);
$v_\sigma \leftarrow$ *verify_signature*($\mathcal{H}(\mathcal{C}), rP, v_\sigma$);
if $v_\sigma \neq \perp$ **then**
// Replay/Relay Attack Verification
  $T \leftarrow$ *extract_timestamp*($\mathcal{C}$);
  if *diff*(T, *current_timestamp*())$\geq \mathcal{T}_{thr}$;
  then
    $\mathcal{A}_p \leftarrow 1$;
    print("Potential Replay Attack");
  else
    $\mathcal{A}_p \leftarrow \perp$;
  end
  $L \leftarrow$ *extract_position*($\mathcal{C}$);
  if *diff*(L, *current_location*())$\geq \mathcal{D}_{thr}$;
  then
    $\mathcal{A}_l \leftarrow 1$;
    print("Potential Relay Attack");
  else
    $\mathcal{A}_l \leftarrow \perp$;
  end
else
  print("Invalid Beacon Signature");
  return $\perp$;
end
return $\mathcal{A}_p, \mathcal{A}_l$;

Finally, it is worth noticing that *SpreadMeNot* is designed to obfuscate sensitive data, such as the real identity of the user, against content-based inference attacks, e.g., *social graph attacks*. Indeed, a social graph attack allows an adversary to infer a user's real identity by leveraging the data transmitted among smart devices. Under *SpreadMeNot*, every device generates a series of ephemeral public keys (and the corresponding private keys) that are linked to the real identity of the user. This approach makes an "indistinguishability network" where users are linked to others in a privacy-preserving manner. In other words, the exchanged information is not sensitive to social graph attacks, since the real identity of the user is obfuscated with a random and constantly changing public key.

As shown in Fig. 4, to construct the beacon at time slot T, user i retrieves and stores in L its GPS position, it chooses a random value r_T, such that $r_T < q$, and computes the tuple $\langle r_TG, r_TP_i \rangle$ that is broadcast by user i at time slot T. Finally, it appends the computed beacon signature σ. As depicted in Fig. 5, when a user j downloads a new contact list, he must check whether any of his own beacons are included in that list. This is done as follows: for every tuple $\langle rG, rP \rangle$ in the contact list, the user multiplies the first term with his own secret key x_j. If the result is equal to the second term, the user confirms that he is the owner of that particular beacon. It is important to note that the user cannot compare any other beacon against his own list and, thus, can not deduce any information about the person(s) in the published list.

Finally, we should emphasize that the core beacon (the pair of elliptic curve points) is the only information needed in the test function. The metadata and signatures are removed from the mobile device after they have been verified. The metadata and signatures are only used to prevent replay/relay attacks and are never transmitted to the centralized server.

5.3 Security and Privacy Analysis

We now discuss the security and privacy characteristics of our protocol, with respect to Table 4.2. In terms of privacy, *SpreadMeNot* has several advantages over the current state-of-the-art approaches. First, it is immune to eavesdropping attacks because the published information (from infected users) is randomized and, thus, cannot be linked to any previously intercepted beacon by an adversary. As such, *SpreadMeNot* also protects the privacy of the users who have been diagnosed as positives. Second, the infected user never reveals his/her own beacons and, therefore, cannot be traced inside a malicious user's contact list, which may include detailed timestamp and location information for each contact. More importantly, the published contact lists are permuted and do not include any identifiable information besides the two elliptic curve points (i.e., the signatures and all metadata are removed). As a result, even if an adversary matches a beacon inside the published contact list, it is impossible to tie that beacon to a specific location and timestamp. Consequently, our protocol is very resilient against linkage attacks. Finally, by attaching a digital signature in every transmitted beacon, *SpreadMeNot* is the first protocol in the literature that mitigates both replay (because we include the beacon's timestamp) and relay (because we include the beacon's location) attacks.

On the other hand, protocols that employ Diffie-Hellman key exchange, such as Whisper and DIMY, utilize contact IDs that are deterministically generated. As such, an

adversary who interacted with an infected user can tie the encounter to a specific location/time.

A final note regards the privacy implications of including time and location data on the transmitted beacons. Mobile location data comes from various sources, including cellphone towers, GPS signals, and WiFi signals. In *SpreadMeNot*, we choose to transmit coarse geo-location information to prevent relay attacks without compromising the users' privacy. For example, the location can be encoded with GeoHash, a convenient way of expressing a location—anywhere in the world—using a short alphanumeric string (with greater precision obtained via longer strings). This does not need to be kept secret, because we assume that the receiver of the BLE frame (i.e., the adversary's antenna) is physically close to the sender. Therefore the inclusion of location data in the BLE beacons does not reveal any valuable information to an adversary. Indeed, an adversary that intercepts a BLE beacon from a random device can easily compute an accurate location for this device (more precise than a GeoHash location), using the adversary's antenna location and the beacon's RSSI. And, according to our threat model, the adversary is capable of geo-tagging all intercepted beacons.

5.4 Formal Verification Through ProVerif

The security properties provided by *SpreadMeNot*, i.e., beacon authenticity, protection against replay and relay attacks, have been formally verified via `ProVerif` [43]. `ProVerif` is an automated verification tool that is adopted in the scientific literature to formally verify the security properties of cryptographic protocols [44], [45], [46]. In particular, `ProVerif` assumes the Dolev-Yao attacker model, i.e., the attacker can read, modify, delete, and forge new packets to be delivered on the communication channel. `ProVerif` checks whether the attacker can break the security goals of the protocol defined by the user. In case an attack is found, `ProVerif` also provides a detailed description of the attack.

We implemented *SpreadMeNot* in `ProVerif` to verify the authenticity of the beacons broadcast by a smart device. According to the logic of the `ProVerif` tool, we defined the following two main events:

1) *releaseBeacon(C)*: Indicates that a generic smart device has accepted to run *SpreadMeNot* with other smart devices by sending beacon C.
2) *acceptBeacon(C)*: Indicates that a generic smart device that received beacon C has terminated *SpreadMeNot* and verified that an authentic surrounding device generates the message.

In line with the logic of `ProVerif`, we established message authenticity by verifying several security properties, such as *sender authentication*, *replay* and *relay* attacks. For the former case, we verified that *acceptBeacon(C)* cannot be executed after the execution of *releaseBeacon(C)* by modelling the property in `ProVerif` as follows:

event(last_event ()) ==> event(previous_event ()) is true: meaning that function *last_event* is executed only when another function, namely *previous_event* is executed.

An excerpt of the output of the `ProVerif` tool is shown in Fig. 6.

Verification summary

```
Query event(acceptBeacon(beacon))
==>event(releaseBeacon(beacon)) is true.
```

Fig. 6. Excerpt of the output provided by the `ProVerif` tool.

The fulfilment of the query in Fig. 6 demonstrates that the message sent by the smart device is authentic. We handled replay attacks through the verification of the freshness of the timestamp, and we handled relay attacks through the verification of the location distance. We have also released the source code of *SpreadMeNot* in the `ProVerif` tool as open-source [16], to allow interested readers to verify our claims and further re-use our code.

6 Performance Evaluation

In this section we provide an experimental evaluation of the *SpreadMeNot* protocol. It is worth noticing that, compared to the aforementioned solutions proposed in the literature, *SpreadMeNot* is not the most computational or energy efficient solution—while still being largely viable. But, *SpreadMeNot* sports provable security and privacy features unmatched by competing solutions. Indeed, we want to remark that *SpreadMeNot* is a trade-off approach that mitigates the main security and the privacy issues that emerged by our analysis of the existing contact tracing solutions, at the expense of a slight increase in the energetic and computational overhead. Further possible performance improvements are left as future work.

We evaluated the cryptographic component of *SpreadMeNot* on an LG Google Nexus 5X, equipped with a hexa-core 64-bit CPU (4×1.4 GHz Cortex-A53 and 2×1.8 GHz Cortex-A57) ARMv8-A. The wireless communications module was evaluated on a Qualcomm Atheros QCA6174 A SoC hardware platform with built-in IEEE 802.11ac radio, Bluetooth v4.2, and Bluetooth Low Energy. The device supports the standard 256-QAM modulation and utilizes 1,216 KB RAM and 448 KB ROM for Wi-Fi, and 192 KB RAM and 672 KB ROM for Bluetooth [47].

In particular, the cryptographic operations of the *SpreadMeNot* protocol where implemented in *C++*, with the well known *OpenSSL 1.1.1 d* C library [48], using Android Native Development Kit (NDK) with *Android 8.1 Oreo* OS [49]. For the experimental evaluation, we selected four elliptic curves, i.e., `secp128r1`, `secp160r1`, `secp192k1`, and `secp256k1`. According to NIST's latest guidelines, these curves provide security levels of 64, 80, 96, and 128 bits, respectively [50]. Furthermore, following NIST's recommendations, we adopted the `SHA256` hash function for the digital signatures. We considered the following two performance metrics for our protocol: (i) CPU time for the beacon generation function; and, (ii) energy consumption for the computations and TX/RX operations.

Specifically, to estimate the energy consumption due to the public-key cryptographic primitives, we focused on the protocol's basic operation—the elliptic curve point-scalar multiplication—which is by far the most CPU intensive operation. To this end, we leveraged the power profile

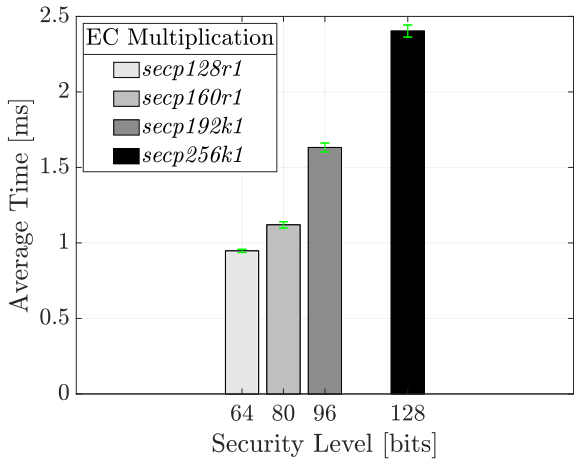

Fig. 7. Average CPU time for elliptic curve point-scalar multiplication.

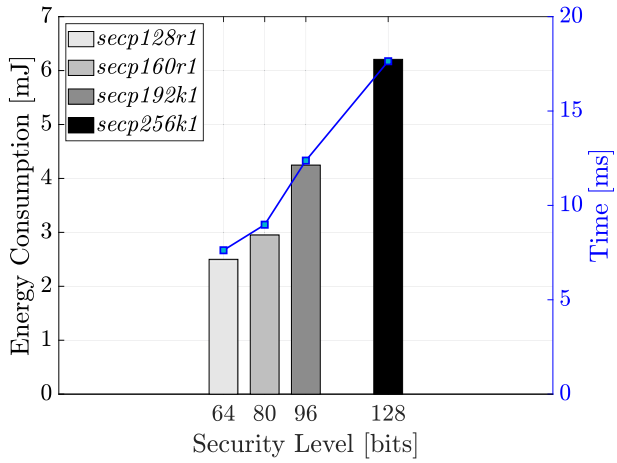

Fig. 8. Total time and energy consumption (TX, RX, and Idle Status) of *SpreadMeNot* on the LG Google Nexus 5X smartphone.

component of the Android OS that outputs the current consumption values for sensors and CPUs and the approximate battery drain caused by the component over time. The power profile is provided by the device manufacturer [51]. After performing the scalar-point multiplication, we estimated the current drained at 105.87 mA. Therefore, we concluded that this value is the instantaneous current drained by the CPU to perform the elliptic curve operation. Additionally, we measured the battery voltage at 3.9 V, using a common battery monitoring application.

Fig. 7 illustrates the CPU time that is required to perform a single scalar-point multiplication for various security levels. At 80 bits security, the operation takes 1.1196 ms to complete and it consumes ≈ 0.4623 mJ of energy. When we require 128 bits security, the cost is approximately doubled, i.e., 2.4030 ms of CPU time and ≈ 0.9922 mJ of consumed energy. The tests related to the CPU time were repeated 5,000 times and, in Fig. 7, we report the mean value and the 95% confidence intervals.

To measure the energy consumption and transmission time during the beacon broadcast operation, we assumed the Bluetooth v4.2 standard, where the data transmission rate is 1 Mbps. Nevertheless, given the multipath and slow fading effects that negatively affect RF communications (due to scattering, reflection, and diffraction), we adopted a conservative stance and assumed that the maximum throughput is 0.8 Mbps. We evaluated *SpreadMeNot* at the MAC layer, where the Bluetooth v4.2 standard sets the maximum frame size at 265 bytes (using packet length extension). This translates to a maximum payload size of 251 octets, where 4 bytes are devoted to the L2CAP Header, 3 bytes to the ATT Header, and up to 244 bytes to the ATT Data for transmitting the *SpreadMeNot* payload [52]. Thus, if the protocol needs to transmit larger payloads, they must be fragmented. Note that, the amount of transferred data is related to the desired security level, i.e., the communication cost increases for more secure elliptic curve groups. However, one technique to reduce the payload size is to employ point compression, where only one coordinate is transmitted for each elliptic curve point.

The energy consumption of the TX and RX operations for the Qualcomm Atheros QCA6174 A SoC module can be computed from the area underlying the current consumption curve, as depicted in Eq. 2.

$$E[mJ] = 1.8\ V \cdot \int_0^{\tau} i(t)dt \tag{2}$$

Here, E is the energy consumption (measured in mJ), $i(t)$ is the instantaneous current drain (in mA), τ is the operation duration, and 1.8 V is the minimum voltage of the QCA6174 A board (in Volts). This translates to ≈ 46 mA in TX mode and ≈ 42 mA in RX mode, for the Bluetooth v4.2 protocol.

Fig. 8 illustrates the total energy consumption and the time required to complete one beacon exchange under the *SpreadMeNot* protocol. First, the energy consumption ranges from a minimum of ≈ 2.5 mJ to a maximum of ≈ 6.2 mJ, based on the underlying security level. Similarly, for 64 bits security, the average time to complete the exchange is 7.623 ms, where 5.6837 ms are required for the cryptographic operations (i.e., beacon generation, and signature generation and verification), while 1.9393 ms are devoted to the exchange of the crypto material. For 128 bits security, the time to complete the protocol increases to 17.638 ms, where 14.418 ms are devoted to the cryptographic computations and 3.22 ms are related to the exchange of the crypto material. We emphasize that the overall duration of the beacon exchange can be affected by the configuration of the operating system features at the MAC link-layer, and also various RF phenomena at the PHY layer.

Assuming the standard-compliant MAC frames in Bluetooth v4.2, and the EC point compression technique, Table 2 summarizes the different costs associated with a *single beacon exchange* in the *SpreadMeNot* protocol, while Table 3 provides additional configuration details. The results confirm that, even at the highest security level, *SpreadMeNot* is feasible for resource-constrained smart devices. Indeed, the battery capacity of the LG Google Nexus 5X smartphone is 41,796 J (2,700 mAh), so, assuming the highest level of security (128 bits), a run of our protocol consumes $\approx 1.5 \cdot 10^{-7}\%$ of the battery capacity. Also, note that the payload size is well within the standard's limit, so beacons can be transmitted within a single MAC frame.

TABLE 2
SpreadMeNot Cost Summary for One Beacon Exchange

Feature	Security Level (bits)			
	64	**80**	**96**	**128**
Payload Size (B)	76	92	108	140
Energy Consumption (mJ)	2.500	2.952	4.247	6.208
Time Duration (ms)	7.623	8.977	12.373	17.638

7 Discussion

Our experimental evaluation detailed in the previous section illustrates the feasibility of public-key cryptography in the context of contact tracing. Compared to the state-of-the-art symmetric key approaches, such as Apple/Google, *SpreadMeNot* shows a more sustained overhead, though compensated from much stronger security and privacy guarantees. In our evaluation, we focused on the beacon exchange process, which is the most frequently performed operation, as in the vast majority of contact tracing apps. However, one operation where the performance of *SpreadMeNot* is orders of magnitude more expensive with respect to other solutions, is the exposure notification operation. Though, it should be noted that this operation is invoked much less frequently, and could be easily outsourced, as discussed later.

In detail, in the event of a positive diagnosis, the app has to re-randomize all beacons in its contact list prior to uploading them to the centralized server. Assuming an average of 200 significant contact events per day, over a period of 14 days, this amounts to 2,800 entries. Recall that the `RandBeacon` function necessitates two point-scalar multiplications and, based on the results of Fig. 7, the entire process may consume (for 128 bits security) up to 13.44 s of computing time and 5.55 J of energy. Additionally, the communication cost to upload the beacons is 179.2 KB. Again, these costs may be dramatically reduced: cut by half if we choose 80 bits security, or completely removed if we resort to outsourcing, as discussed in the following.

An even more resource-demanding operation is the actual contact tracing, where individual devices have to identify possible contagion events after downloading the latest contact lists of recently diagnosed patients. Assuming that this operation is performed on a daily basis, the typical number of beacons that need to be matched would be in the order of millions (e.g., 5.6 million, if there are 2,000 new cases with 2,800 stored beacons each). In this example, the smartphone would consume (for 128 bits security) approximately 3.7 h of computing time and 5,556 J of energy, i.e., by spending at most $\approx 13.29\%$ of energy from the overall LG Google Nexus 5X battery capacity. However, such an activity could be easily delegated by the user to its own laptop or other trusted plugged/more resourceful device—it is not mandatory to have it run on the user's mobile. There is also a communication cost of 360 MB to download the beacons from the centralized server. Again, these costs may be dramatically reduced: cut by half if we choose 80 bits security, or completely removed if we resort to outsourcing, as discussed in the following.

Despite the highlighted cost, we strongly believe that *SpreadMeNot*'s superior security and privacy properties offset any argument regarding its performance. More importantly, the cited costs are easily mitigated by separating the *online* and *offline* components of the protocol. Specifically, it is worth noting that all the point-scalar multiplications in the beacon generation process can be performed offline. There are a total of three such operations: two for computing the beacon itself and one for computing the digital signature (which is input-independent). Therefore, we can precompute offline a large number of random elliptic curve points that may be used during the online phase (beacon exchange) by the device. Consequently, the beacon generation process can be reduced to computing just one hash function and two integer multiplications modulo q. Both are very cheap operations—involved in the computation of the digital signature.

Similarly, all the remaining expensive operations (contact list maintenance and contact tracing) may be safely moved into the offline module. Contact list maintenance involves the verification of all signatures for the beacons that are inserted into the list (to detect replay/relay attacks), and the

TABLE 3
Bluetooth 4.2 Frame and *SpreadMeNot* Payload Notation

Acronym	Content/Size	Description
PRB	`1` B	Preamble used for synchronization
AA	`0x8E89BED6`	Access Address for broadcasted packets
PDUH	`2` B	Protocol Data Unit Header
DCP	Up to `251` B	Data Channel Payload
MIC	`0x00000000` – `0xFFFFFFFF`	Message Integrity Check
CRC	`3` B	Cyclic Redundancy Check
Data Channel Payload		
L2CAPH	`4` B	Logical Link Control and Adaptation Protocol Header
ATTH	`3` B	Attribute Protocol Header
ATTD	Up to `244` B	Attribute Data (*SpreadMeNot*) Payload
***SpreadMeNot*Payload (ATTD)**		
L (*lat*, *lon*)	`8` B	GPS Coordinates — Latitude, Longitude
T	`4` B	Message Timestamp
rG	`16-32` B	Beacon First EC Point
rP	`16-32` B	Beacon Second EC Point
σ	`32-64` B	Beacon Signature

re-randomization of the accepted beacons in order to minimize the online computational cost in the event of a positive diagnosis. More importantly, contact tracing is also independent of the protocol's everyday operations and may be performed in an offline fashion. To this end, we propose the following two approaches for handling the offline operations:

- *Night mode:* The offline work will be performed during night time, when the smartphone is charging and connected to the home WiFi network.
- *Desktop app:* A companion desktop app will be developed to handle the offline tasks. The user will periodically sync the smartphone with the desktop app, in order to upload and download the necessary information. Note that this one would be the preferred solution, as modern desktop CPUs can perform the cryptographic operations significantly faster. Also, the app will implement multi-threading to take advantage of the multiple CPU cores, since all the protocol's operations are highly parallelizable.

8 Conclusion

In this paper, we have provided two main contributions in the domain of digital contact tracing.

We first performed a thorough evaluation of the privacy and security characteristics of the main contact tracing apps, focusing on their basic mechanisms. Our results showed that the most prominent solutions fail to protect the privacy of positively diagnosed individuals, as well as being vulnerable to a variety of active and passive attacks. To cope with the above-highlighted shortcomings, our second contribution was the design of *SpreadMeNot*, a novel, provably secure contact tracing protocol that protects the privacy of *all* users while at the same time being resilient against most passive and active attacks, including eavesdropping and replay/relay attacks. *SpreadMeNot* leverages low-overhead public-key cryptographic primitives. Experimental results support the fact that the proposed solution can be easily handled by modern smartphones. In addition, we have demonstrated that all resource-intensive operations can be safely moved into an offline module, thus rendering *SpreadMeNot*'s online proximity tracing task extremely lightweight. Given the strong and provable security and privacy properties, combined with the experimentally proved sustainable overhead, we argue that *SpreadMeNot* is the ideal candidate for digital contact tracing. Finally, the open source nature of this project will also pave the way for further solutions in the domain.

Acknowledgments

The authors would like to thank the anonymous reviewers, that helped to improve the quality of the manuscript. The findings reported herein are solely responsibility of the authors.

References

[1] H. Wen, Q. Zhao, Z. Lin, D. Xuan, and N. Shroff, "A study of the privacy of COVID-19 contact tracing apps," in *Proc. Int. Conf. Secur. Privacy Commun. Netw.*, 2020, pp. 297–317.

[2] L. Ferretti *et al.*, "Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing," *Science*, vol. 368, no. 6491, 2020, Art. no. eabb6936.

[3] T. Martin, G. Karopoulos, J. L. Hernández-Ramos, G. Kambourakis, and I. Nai Fovino, "Demystifying COVID-19 digital contact tracing: A survey on frameworks and mobile apps," *Wireless Commun. Mobile Comput.*, vol. 2020, 2020.

[4] Z. Su, K. Pahlavan, and E. Agu, "Performance evaluation of COVID-19 proximity detection using bluetooth LE signal," *IEEE Access*, vol. 9, pp. 38 891–38 906, 2021.

[5] Z. Akhavan, M. Esmaeili, D. Sikeridis, and M. Devetsikiotis, "Internet of Things-enabled passive contact tracing in smart cities," *Elsevier Internet Things*, vol. 18, 2021, Art. no. 100397.

[6] P. C. Ng, P. Spachos, and K. N. Plataniotis, "COVID-19 and your smartphone: BLE-Based smart contact tracing," *IEEE Syst. J.*, vol. 15, no. 4, pp. 1–12, Apr. 2022.

[7] P. Tedeschi, S. Bakiras, and R. Di Pietro, "IoTrace: A. flexible, efficient, and privacy-preserving IoT-enabled architecture for contact tracing," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 82–88, Jun. 2021.

[8] M. Cunche, A. Boutet, C. Castelluccia, C. Lauradoux, D. Le Métayer, and V. Roca, "On using bluetooth-low-energy for contact tracing," Ph.D. dissertation, Inria Grenoble Rhône-Alpes; INSA de Lyon, 2020.

[9] N. Ahmed *et al.*, "A survey of COVID-19 contact tracing apps," *IEEE Access*, vol. 8, pp. 134 577–134 601, 2020.

[10] TraceTogether, 2020. Accessed: Jul 16, 2021. [Online]. Available: https://www.tracetogether.gov.sg/

[11] PEPP-PT Team, Pan-puropean privacy-preserving proximity tracing, Accessed: Jul. 16, 2021. [Online]. Available: https://www.pepp-pt.org/

[12] Decentralized privacy-preserving proximity tracing: Overview of data protection and security, 2020. Accessed: Jul 16, 2021. [Online]. Available: https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf

[13] Apple Google, Privacy-preserving contact tracing, 2021. Accessed: Jul 16, 2021. [Online]. Available: https://www.apple.com/covid19/contacttracing

[14] R. Sun, W. Wang, M. Xue, G. Tyson, S. Camtepe, and D. Ranasinghe, "Vetting security and privacy of global COVID-19 contact tracing applications," 2020, *arXiv:2006.10933*.

[15] L. Garg, E. Chukwu, N. Nasser, C. Chakraborty, and G. Garg, "Anonymity preserving IoT-Based COVID-19 and other infectious disease contact tracing model," *IEEE Access*, vol. 8, pp. 159 402–159 414, 2020.

[16] P. Tedeschi, S. Bakiras, and D. Pietro Roberto, "Proverif - formal verification code of SpreadMeNot," 2021. [Online]. Available: https://github.com/pietrotedeschi/spreadmenot

[17] SIG Bluetooth, Bluetooth specification version 4.2, 2022, [Online]. Available: http://www.bluetooth.com

[18] N. K. Gupta, *Inside Bluetooth Low Energy*. Norwood, MA, USA: Artech House, 2016.

[19] US Government, Department of Defense: Washington, DC, USA, "Global positioning system standard positioning service performance standard (GPS SPS PS), 5th ed." 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://www.gps.gov/technical/ps/2020-SPS-performance-standard.pdf

[20] D. Schmidt, K. Radke, S. Camtepe, E. Foo, and M. Ren, "A survey and analysis of the GNSS spoofing threat and countermeasures," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 1–31, 2016.

[21] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Berlin, Germany: Springer, 2003.

[22] W. Tuttlebee, *Software Defined Radio: Enabling Technologies*. Hoboken, NJ, USA: Wiley, 2002.

[23] Y. Gvili, "Security analysis of the COVID-19 contact tracing specifications by apple Inc. and Google Inc." cryptology ePrint archive, Report no. 2020/428, 2020.

[24] S. Vaudenay, "Analysis of DP3T," Cryptology ePrint Archive, Report no. 2020/399, 2020.

[25] Apple Google, Contact tracing - cryptography specification, 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ContactTracing-CryptographySpecification.pdf

[26] A. Berke, M. Bakker, P. Vepakomma, K. Larson, and A. 'Sandy' Pentland, "Assessing disease exposure risk with location data: A proposal for cryptographic preservation of privacy," 2020, *arXiv:2003.14412*.

[27] G. Oligeri, S. Sciancalepore, O. A. Ibrahim, and R. Di Pietro, "Drive me not: GPS spoofing detection via cellular network: (architectures, models, and experiments)," in *Proc. 12th Conf. Secur. Privacy Wireless Mobile Netw.*, 2019, pp. 12–22.
[28] H. Hu and N. Wei, "A study of GPS jamming and anti-jamming," in *Proc. 2nd Int. Conf. Power Electron. Intell. Transp. Syst.*, 2009, pp. 388–391.
[29] Israeli health ministry, 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://govextra.gov.il/ministry-of-health/hamagen-app/download-en/
[30] South Korea, Corona 100 m, 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://www.theguardian.com/commentisfree/2020/mar/20/south-korea-rapid-intrusive-measures-Covid-19
[31] TCN Coalition, Temporary contact numbers protocol, 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://github.com/TCNCoalition/TCN
[32] J. Bay *et al.*, "BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders," Government Technol. Agency-Singapore, Mapletree Business City, Singapore, Tech. Rep., vol. 18, 2020.
[33] MIT, PACT:Private automated contact tracing, 2020 Accessed: Jul. 16, 2021. [Online]. Available: https://pact.mit.edu/
[34] Nodle, Whisper tracing - an open and privacy first protocol for contact tracing, 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://docsend.com/view/nis3dac
[35] W. Beskorovajnov, F. Dörre, G. Hartung, A. Koch, J. Müller-Quade, and T. Strufe, "ConTra Corona: Contact tracing against the coronavirus by bridging the centralized–decentralized divide for stronger privacy," Cryptology ePrint Archive, Report 2020/505, 2020.
[36] C. Castelluccia *et al.*, "DESIRE: A 3rd way for a european exposure notification system leveraging the best of centralized and decentralized systems," May 2020, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02570382/
[37] N. Ahmed *et al.*, "DIMY: Enabling privacy-preserving contact tracing," *J. Netw. Comput. Appl.*, vol. 202, p. 103356, 2022.
[38] Y. Bengio *et al.*, "Inherent privacy limitations of decentralized contact tracing apps," *J. Amer. Med. Informat. Assoc.*, vol. 28, pp. 193–195, 2020.
[39] M. A. Azad *et al.*, "A first look at privacy analysis of COVID-19 contact tracing mobile applications," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15 796–15 806, 2020.
[40] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert, "Bluetooth positioning using RSSI and triangulation methods," in *Proc. IEEE 10th Consum. Commun. Netw. Conf.*, 2013, pp. 837–842.
[41] N. Ahmed *et al.*, "A survey of COVID-19 contact tracing apps," *IEEE Access*, vol. 8, pp. 134 577–134 601, 2020.
[42] T. Silde and M. Strand, "Anonymous tokens with public metadata and applications to private contact tracing," Cryptology ePrint Archive, Report 2021/203, 2021.
[43] B. Blanchet, "Automatic verification of correspondences for security protocols," *J. Comput. Secur.*, vol. 17, no. 4, pp. 363–434, 2009.
[44] A. Aziz, P. Tedeschi, S. Sciancalepore, and R. D. Pietro, "SecureAIS - securing pairwise vessels communications," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2020, pp. 1–9.
[45] N. Saxena, M. Conti, K.-K. R. Choo, and N. S. Chaudhari, "BASVAS: A novel secure protocol for value added service delivery to mobile devices," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1470–1485, 2020.
[46] H. Tan, M. Ma, H. Labiod, A. Boudguiga, J. Zhang, and P. H. J. Chong, "A secure and authenticated key management protocol (SA-KMP) for vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9570–9584, Dec. 2016.
[47] Qualcomm, Qualcomm QCA6174 A Wi-Fi/Bluetooth SoC, 2019. Accessed: Jul. 16, 2021. [Online]. Available: https://www.qualcomm.com/media/documents/files/qca6174a-product-brief.pdf
[48] OpenSSL Software Foundation, "OpenSSL - cryptography and SSL/TLS toolbox," 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://www.openssl.org/
[49] Google, Android (operating system), 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://www.android.com/
[50] E. Barker, "Recommendation for key management: Part 1 - general," NIST, Tech. Rep., May 2020. Accessed: Jan. 16, 2022.
[51] Measuring power values | android open source project, 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://source.android.com/devices/tech/power/values
[52] Higher speed - how fast can it be?, 2020. Accessed: Jul. 16, 2021. [Online]. Available: https://www.bluetooth.com/blog/exploring-bluetooth-5-how-fast-can-it-be/

Pietro Tedeschi (Member, IEEE) received the bachelor's degree in computer and automation engineering, in 2014, and the master's degree (with honors) in computer engineering both from the "Politecnico di Bari", Italy. From 2017 to 2018, and the PhD degree in computer science and engineering (Cybersecurity) from Hamad Bin Khalifa University (HBKU), Doha, Qatar, in 2021. He is senior security researcher with Technology Innovation Institute, Secure Systems Research Center, Abu Dhabi, United Arab Emirates. He worked as security researcher with CNIT (Consorzio Nazionale Interuniversitario per le Telecomunicazioni), Italy, for the EU H2020 SymbIoTe project. His research interests span over UAV/Drone Security, the *Wireless Security*, *Internet of Things (IoT)*, *Applied Cryptography*, and *Cyber-Physical Systems*.

Spiridon Bakiras (Member, IEEE) received the BS degree in electrical and computer engineering from the National Technical University of Athens, in 1993, the MS degree in telematics from the University of Surrey, in 1994, and the PhD degree in electrical engineering from the University of Southern California, in 2000. He is currently an associate professor with the Infocomm Technology Cluster at the Singapore Institute of Technology. Before that, he held teaching and research positions with Hamad Bin Khalifa University, Qatar, Michigan Technological University, the City University of New York, the University of Hong Kong, and the Hong Kong University of Science and Technology. His current research interests include database security and privacy, mobile computing, and spatiotemporal databases. He is a member of the ACM, and a recipient of the U.S. National Science Foundation (NSF) CAREER award.

Roberto Di Pietro (Senior Member, IEEE) ACM distinguished scientist, is full professor in Cybersecurity at HBKU-CSE. Previously, he was in the capacity of global head security research aith Nokia Bell Labs, and associate professor (with tenure) of computer science with the University of Padova, Italy. He also served more than 10 years as senior military technical officer. Overall, he has been working in the cybersecurity field for more than 25 years, leading both technology-oriented and research-focused teams in the private sector, government, and academia (MoD, United Nations HQ, EUROJUST, IAEA, WIPO). His main research interests include security and privacy for wired and wireless distributed systems (e.g., Blockchain technology, Cloud, IoT, On-line Social Networks), virtualization security, applied cryptography, computer forensics, and data science. Other than being involved in M&A of start-up and having founded one (exited), he has been producing more than 250 scientific papers and more than 15 patents over the cited topics, and coauthored three books, edited one, and contributed to a few others. He is serving as EiC for Security and Communication Networks (Wiley-Hindawi), and AE for *Computer Communications*, *Computer Networks*, *Pervasive and Mobile Computing*, *Journal of Computer Security*, and other Intl. journals. In 2011-2012, he was awarded a chair of Excellence from University Carlos III, Madrid. In 2020, he received the Jean-Claude Laprie award for having significantly influenced the theory and practice of Dependable Computing.