Risk and Architecture Factors in Digital Exposure Notification

Archanaa S. Krishnan[1](✉), Yaling Yang[1], and Patrick Schaumont[2]

[1] Virginia Tech, Blacksburg, VA 24060, USA
{archanaa,yyang8}@vt.edu
[2] Worcester Polytechnic Institute, Worcester, MA 01609, USA
pschaumont@wpi.edu

Abstract. To effectively trace the infection spread in a pandemic, a large number of manual contact tracers are required to reach out to all possible contacts of infected users. Exposure notification, a.k.a. digital contact tracing, can supplement manual contact tracing to ease the burden on manual tracers and to digitally obtain accurate contact information. We study how risk emerges in security, privacy, architecture, and technology aspects of exposure notification systems. We provide potential overhead in using Bluetooth-based systems and discuss the architectural support required for other types of systems, and we wrap up with a discussion on architecture aspects to support these solutions.

Keywords: Exposure notification · Contact tracing · Privacy preserving · Privacy friendly architectures

1 Introduction

In recent months, the SARS-CoV-2 virus has infected four million people claiming over 250,000 lives. At the onset of the SARS-CoV-2 virus infection, the governments around the world placed entire states under lockdown to prevent the spread of the infection. Although this strategy was effective in curtailing the spread of the infection, it has adversely affected other aspects of life, increased the unemployment rate, stressed the medical infrastructure, affected the global supply chain, created both food shortage, and brought about food waste. To ease the lockdown and to support long-term infection management, governments are next considering and implementing a test-trace-isolate strategy. The aim of this strategy is, first, to reduce the infection rate and, second, to limit the confinement to exposed individuals and communities instead of countrywide lockdown. The effectiveness of this strategy depends on widespread testing and contact tracing.

The main goal of contact tracing is to interrupt ongoing transmission, reduce the spread of infection, and study the epidemiology of the infection in a particular population. Contact tracing has been effectively used against tuberculosis, sexually transmitted diseases, vaccine preventable diseases, bacterial and viral infections. The World Health Organization has used contact tracing to prevent transmission of infection. It is performed by public health workers in three steps [33]. First, the public health workers, a.k.a contact tracers, work with infected individuals to identify contacts by revisiting their movements before the onset of illness. Second, all potential contacts of the infected individual are informed of their contact and advised about early care. Third, contact tracers periodically follow-up with the contacts for signs of symptoms and test for the illness.

In this paper, we focus on digital contact tracing and its security and privacy. We refer to digital contact tracing as *exposure notification* [36]. We present the components required to build a secure and privacy friendly exposure notification system through the following contributions:

- We study the state-of-the-art proposals in privacy preserving solutions and differentiate their architectural and privacy design.
- We analyze the risks involved in exposure notification. In particular, we analyze the security and privacy risks involved in collecting location data.
- We consolidate open research challenges in security and privacy friendly architectures for the exposure notification system. We present a roadmap to achieve secure and private architectures that may serve useful beyond the COVID-19 pandemic.

Organization: The rest of the paper is organized as follows. Section 2 presents the state-of-the-art exposure notification proposals and their security and privacy design. Section 3 provides a classification of risks involved in exposure notification based on several factors. Section 4 studies the role of architecture in designing, selecting, and implementing these proposals. Section 5 summarizes a few open research questions and presents an agenda to achieve secure and privacy friendly architectures.

2 Summary of Proposals

2.1 Preliminaries

We generalize exposure notification systems as follows. The main parties involved in exposure notification systems are the app users, health authorities, and backend servers. The system is implemented as a mobile app and it operates in three phases [13] - collecting user location data, reporting infected user location data to the server, and computing exposure risk. Exposure risk is used to identify if any user was in contact with one or more infected users. We refer to app users who have tested positive for infection as infected users and the remaining as uninfected users. We refer to all types of contact information with its timestamp of collection as location data in the rest of the paper unless explicitly differentiated.

Architectural Differences in Exposure Notification: The proposed exposure notification systems can be broadly differentiated based on the architecture used to collect contact information. The architecture includes Bluetooth, geolocation, QR code, WiFi access point, or a combination of these techniques. The difference in architecture can be mostly observed in phase one of the system operation. For example, Bluetooth-based systems collect the Bluetooth pseudo IDs scanned from encountered phones whereas QR code-based systems collect the codes scanned from the visited public facilities. In all types of notification systems, the timestamp is used to mark the start of the required retention period for contact data. In notification systems other than Bluetooth-based ones, the timestamp is also a required parameter to properly define the location coordinates of users in space and time. Bluetooth ping-based systems are directly between users, and hence do not require space/time location coordinates.

Security Models Used in Exposure Notification: A majority of the existing proposals use a semi-trusted, also called honest-but-curious, server. Such a server is assumed to not add or remove information shared by infected users, but the server will not protect the privacy of all users [9]. The privacy of users in such models is independent of the trust placed in the server as the server only stores anonymized user data. In exposure notification systems, a central authority, such as the health authority, maintains the server. Based on the trust in the central authority, the proposals are broadly classified as centralized and decentralized models, with a notable difference in phase three of the system operation. In a *decentralized model*, the exposure risk is computed locally, by the user's mobile app, without revealing uninfected user's location data to the server. In a *centralized model*, the exposure risk is computed by the server and it notifies each user who queries.

2.2 Existing Deployments

There are several apps deployed by governments and academia/private sector. All the government apps are based on the centralized model, where the server is trusted to an extent to maintain user privacy and to compute the exposure risk [16,25,28,29,34,35]. Only a few of the governments have published their reference implementation [25,28,34], where Singapore's effort is notable for its transparency in reference implementation, underlying protocol [18], and privacy design. Even though the exposure notification initiatives were conceived to protect the citizens, it creates an avenue for exploiting their privacy when there is a lack of transparency in design, implementation, and evaluation. Recently, Arogya Setu, the app from the Indian government, was hacked to reveal the infection rate at any location with a precision of a meter [32]. The apps deployed by academia and private sector [6–9,12,15] are based on the decentralized model and are well documented with reference implementation, white papers, and open discussion on improving their privacy. A majority of them are based on the protocols discussed below.

2.3 Exposure Notification Protocols

Decentralized Privacy-Preserving Proximity Tracing (DP-3T) [9], Privacy-Sensitive Protocols And Mechanisms for Mobile Contact Tracing (PACT) [23], Private Automated Contact Tracing - the PACT protocol [11], the TCN(temporary contact number) protocol, and the Apple—Google collaboration framework [24] are proposals for secure and decentralized Bluetooth based tracing that minimize security and privacy risks in digital exposure notification. ROBust and privacy-presERving proximity Tracing protocol (ROBERT) [13] is a centralized protocol with similar design and federated servers. These designs guarantee user privacy by generating ephemeral random pseudo IDs using AES and SHA-2 based algorithms. Bluetooth based protocols only relying on anonymous pseudo IDs, are susceptible to various risks discussed in Sect. 3.

Epione [44] and TraceSecure [20] propose techniques beyond using ephemeral pseudo IDs to protect user privacy in Bluetooth-based systems. The former uses private information retrieval (PIR) with homomorphic or symmetric encryption, whereas the latter uses a set-based protocol with public key encryption and additive homomorphic encryption.

The privacy of GPS based proposals, such as PrivateKit [10], SafeTrace [14]and [21,43], differ from Bluetooth-based proposals because geolocation data itself is not anonymized. Instead, they rely on private set intersection (PSI), PIR, homomorphic encryption, and multi-party computation(MPC) protocols to achieve user privacy, similar to the approach of TraceSecure [20] and Epione [44].

3 Risks in Exposure Notification

Figure 1 illustrates the potential attacks that exploit the risks presented in this section. Replay, trolling, linkage, coercion, eavesdropping, collusion, deanonymization, battery drain, and tracing attacks are discussed by Gvili [27] and Vaudenay [45]. Denial of Service(DoS), BleedingBit [41], and BlueBorne [40] attacks are particular to Bluetooth technology.

3.1 Security Risks

Integrity Based Risks: An uninfected user may add bogus location data to their location data to check if a particular location or user is infected. A malicious infected user may add bogus location data to their location data before they share it with the exposure database. A malicious server may modify the exposure database. The server may add bogus location to induce false positives or the server may delete certain locations to induce false negatives. The former could be used to create panic in a certain location and the latter may be exploited by certain services such as cafes and restaurants to delete their location data from the database to maintain a steady influx of customers. Integrity based risks may be exploited for trolling attacks, replay attacks, and to stigmatize infected users.

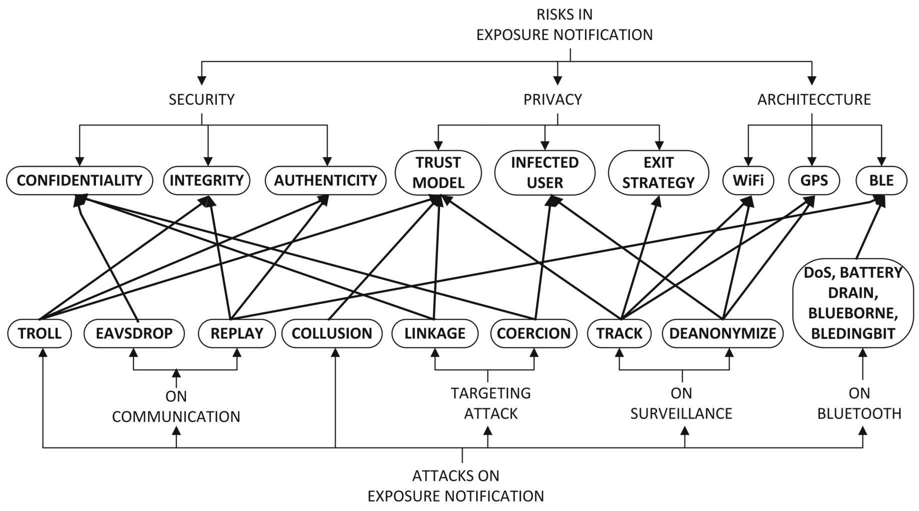

Fig. 1. A flow chart of risks in and attacks [27,40,41,45] on exposure notification systems

Authenticity Based Risks: The authenticity of the exposure database and exposure risk notification is at risk. Unverified uploading of location data to the exposure database can be exploited to upload bogus location data to the server. An attacker may impersonate the server and provide false positive and false negative exposure notification to users. The same attacker may also send malicious exposure database entries to the user and influence the user to compute a false positive/negative exposure notification. These risks may be exploited for trolling attacks.

Confidentiality Based Risks: An attacker can passively observe the communication between the app and the server. The attacker can obtain the location data when it is shared with the server. The attacker may also have access to the victim's terminal device, from which the stored location data may be extracted. All unencrypted data is vulnerable to the attacker. This data can be used for replay attacks, linkage attacks, and coercion threats.

3.2 Privacy Risks

Server Trust Model Risks: All the exposure notification proposals use a server in a centralized or decentralized trust model to store anonymous location data. In both centralized and decentralized models, a malicious server may introduce bogus location data to aid in trolling attacks, as described in authenticity based risks [37]. It may also store the metadata, such as source IP address, related to users, and break user anonymity. This can be further exploited by the server to build social graphs of the users without their consent. A social graph represents the relationship between app users which leaks user privacy. A server in the centralized model compares infected user location data and uninfected user location data to compute exposure risk. An untrusted centralized server has access to all user location data, which can be exploited by the central authority to sell location data for profit. The server trust model is inherently derived from the assumed attacker model. The attacker (app user or a third party) may collude with the server to perform collusion attacks to deanonymize location data using metadata, to obtain the social graph of all users, and obtain location data of all users. Collusion attacks may also be used to exploit the inherent risks of MPC protocols based exposure notification systems [43].

Infected User Privacy Risks: In decentralized exposure notification proposals, the infected user's location data is broadcasted to the public. The attacker can associate location data of each user with an identifier when there is a match in location data between the attacker's app and public database, the attacker is able to identify the infected user using their identifier. This may be exploited to target a particular victim, where the attacker only interacts with the victim and has location data only from the victim. When this location data is compared with the public database the attacker is able to identify if the victim is infected or not [27,45]. This risk could be exploited to deanonymize the infected user using linkage attacks and perform one-entry attacks [13]. A potential solution is not to publish the location data of infected users, but to have a federated server perform the exposure risk calculation using both user and infected user location data. This type of solution has its own risks mentioned above.

Sunset Provision Risks: The purpose of exposure notification systems is to limit the spread of an infection. Without preset deadlines to stop contact tracing, these systems may be exploited to track users beyond the scope of infection control.

3.3 Architecture and Technology Risks

Bluetooth Based Risks: A majority of proximity tracing proposals [9,11,13,23,25] are based on Bluetooth technology. Bluetooth can be exploited to attack the exposure notification system by extending Bluetooth discoverability using a directional antenna. The attacker can create false positive contact data and to potentially perform trolling attacks. Using Bluetooth, an attacker can attack the general operation of a user's mobile phone as the Bluetooth is always on for the operation of exposure notification systems. Bluetooth is vulnerable to Denial of Service (DoS) attacks where the attacker can flood a victim's Bluetooth with a large volume of messages. The device consumes power in analyzing these messages, storing valid messages in memory which overloads the memory, and discarding invalid messages. This may drain the battery on the mobile phone, keep the phone occupied and lead to slow or no response to its regular operation. Bluetooth-based exposure notification system may be used to exploit Bluetooth vulnerabilities such as BleedingBit [41] and BlueBorne [41] which affects regular mobile phone operation. It is also vulnerable to passive tracking and identity exposure [19].

WiFi Access Point Based Risks: WiFi Access Point based proximity tracing systems [17] keep track of the network identifiers of WiFi Access Points. If the attacker has access to these network identifiers and their collection timestamp, they can identify the exact location of the user which leaks user privacy [42].

Geolocation Based Risks: When exposure notification proposals use geolocation based location data, they have access to the absolute user location data. The geolocation data is a commodity to be sold which can be used to construct social grahphs. This maybe exploited as a monitoring tool by nation state and corporations.

4 Role of Architecture

Tracing geolocation data, such as GPS, is considered to be intrusive and in wrong hands, it may be used to track the user. The attacker can use geolocation data to discern a victim's home address, workspace, their daily activities, and social interactions. Short-wavelength technology based proximity tracing was thought to be an alternative to GPS based contact tracing. The ubiquitousness of mobile phones with Bluetooth technology was an off-the-shelf solution adopted by a majority of exposure notification systems. In this section, we analyze the role architecture and technology plays in the design and choice of exposure notification proposals. We provide back-of-the-envelope calculations for the cost of using Bluetooth in exposure notification. We present the architectural requirements for using geolocation based location data for privacy friendly designs.

4.1 Bluetooth, Privacy, and Overhead

The privacy of the user is guaranteed in Bluetooth-based proposals by using ephemeral and random pseudo IDs, typically generated using SHA-2 [9], AES [24] or a pseudo random function [11]. Although modern mobile phone technology is equipped with hardware acceleration for AES and SHA-1, it is optimized for computation on large data. Since the IDs are typically 16B they can be generated with minimal overhead in software. Bluetooth Low Energy (BLE) is the recommended Bluetooth technology to be used in implementing these proposals. BLE is known for consuming low energy when compared to Bluetooth Classic by essentially operating in sleep mode. Table 1 lists the potential overhead of deploying the Google—Apple framework [24] in iPhone 5S. We differentiate the overhead based on two Bluetooth operation - broadcasting and scanning. We compute the broadcasting overhead with a 0.25 s broadcast interval, the recommended broadcast interval is between 0.2–0.27 s [24]. The scanning overhead is computed at an assumed scan interval of 1 s. For simplicity, we only consider the storage overhead of the BLE IDs without the timestamp and other metadata.

The overhead is listed in terms of the size of memory required based on a 14-day retention period and the percentage of battery drain per hour for each Bluetooth operation. The broadcasted ID storage includes the Temporary Exposure Keys (TEK) used to compute the Rolling Proximity Identifiers(RPID) in the past 14-days. Since TEKs are computed daily, the net broadcast ID storage overhead is computed as a product of TEK size in bytes (16B) and the number of days in retention period (14 days) which comes to 224B. The scanned ID storage is computed as a product of the scanned ID size (16B), number of contacts per day (assumed to be 1000), and number of days in retention period (14 days), which equals 224,000B.

The percentage battery drain is computed using technical reports from Aislelabs [1]. iPhone 5 S is equipped with a 1560 mAh battery [2]. When BLE broadcasts its RPIDs at 0.25s broadcasts interval, it uses 0.013% battery per hour [5]. The same iPhone uses 3.75% battery per hour to scan for incoming RPIDs in 1s scan interval [4]. During each scan operation, the phone may detect multiple beacons and process them which could be attributed to a higher battery drain during scanning. The percentage battery drain for Bluetooth operations can be reduced by increasing the broadcast and scan intervals [3], but this may adversely affect the protocol operation.

We highlight that Bluetooth communication is not free even in BLE. Apart from the proximity communication and storage overhead, there are overheads in communicating with the server, computing new IDs, and computing exposure risk.

Table 1. An example of overhead incurred by the Google—Apple framework in iPhone 5S. ID storage size includes the keys used to derive the broadcasted IDs and the scanned IDs in 14-days. The percentage battery drain was computed using Aislelabs technical reports [4,5]. *We assume 1000 contacts/day and a 1 s scan interval

14-day retention	**Broadcast Overhead @ 0.25s interval**	**Scan Overhead @ *1s interval**	**Units**
ID storage size	224	*224,000	B
Battery drain	0.013	3.75	%/hr

4.2 Hardware Acceleration for Privacy Protection

Apart from relying on AES and SHA-256 for security, some proposals use MPC protocols, PSI, and PIR to protect user privacy [37,43]. They use garbled circuits [44] and homomorphic encryption [20] as their underlying cryptographic primitives in their protocol design. There are several advantages to these protocol designs. First, it protects infected users from deanonymization. Since these proposals do not reveal any information about infected users, an attacker cannot infer infected users from their contacts. Second, homomorphic encryption based solutions are effective in using geolocation data without compromising user privacy. Third, homomorphic encryption solutions may be post-quantum secure, which is an added benefit in designing futuristic applications.

A major drawback in implementing such proposals is the computational overhead involved both at the app and the server side. For example, homomorphic computations generate large ciphertext with a message expansion factor of more than two [39]. The encryption, decryption, and evaluation using homomorphic encryption algorithms are also time consuming. Modern mobile phones are only equipped with cryptographic hardware acceleration for AES and SHA-1 or they are equipped for software acceleration for the same using custom instructions. Hardware acceleration for garbled circuit [30] and homomorphic encryption [38] exists only in academic literature. HEAX [38] is a novel architecture for homomorphic computation on encrypted data. It contains hardware modules for high throughput Number Theoretic Transform(NTT), homomorphic multiplication, and key switching. Prior work on hardware implementation for homomorphic encryption mostly designed accelerators for large number [47], polynomial [31] or integer [22] multiplication.

While the ubiquitousness of Bluetooth technology in mobile phones has lead to its widespread adaptations in privacy preserving proposals, the dearth of garbled circuit and homomorphic hardware may be attributed to the reason behind their limited use in the same field.

4.3 Secure Location Data Storage

A few security risks from Sect. 3 can be exploited by the attacker when they have access to read and to write local location data storage. The attacker may add bogus locations to the exposure database to perform trolling attacks. They can add targeted location data to check the infection data of a particular location or user. They can coerce a victim to reveal their location data [45]. These risks and attacks can be avoided if the location data is stored securely, without access to the attacker or even the app user. If the local data storage is encrypted and authenticated, the security risks can be avoided. Vaudenay [45] suggests using Trusted Platform Module(TPM) to prevent coercion threats and trolling attacks, which is also suggested in the DP-3T proposal [9]. Trusted Computing Group (TCG) has platform-specification for implementing TPM in mobile platforms for secure storage and execution [26]. If mobile phones are equipped with secure storage, a trusted exposure notification app may use this secure storage to store local location data.

5 Towards Secure and Privacy Friendly Architectures

The state-of-the-art proposals summarized in Sect. 2 are by no means a comprehensive list. And not all the 'privacy-protecting' proposals actually protect all user privacy [46]. We list a few challenges that persists in many proposals.

First, the security of location data is completely ignored or mentioned in passing as the proposals mainly focus on user privacy. Second, the user privacy is not protected irrespective of infection status and is mostly not protected from malicious central authority. Third, multi-hop transmission of the SARS-CoV-2 virus is failed to be traced by Bluetooth-based systems as these systems only account for human-to-human transmission. And finally, there is lack of details on implementing and maintaining honest-but-curious servers in both centralized and decentralized models.

The existence of these challenges may be attributed to the lack of architectural support to achieve security and privacy in mobile phones. The presence of secure storage and its access to trustworthy apps can help solve the first challenge. The second and challenges are addressed in a few proposals [20,37,43,44], but their practicality may be hindered by the overhead incurred by their privacy protection solution. These solutions use garbled circuits and homomorphic encryption in MPC, PSI and PIR based protocols to both protect infected user privacy and protect geolocation data. They may be viable in the presence of hardware acceleration for their cryptographic primitives. The availability of homomorphic hardware acceleration will not only be useful for exposure notification, as homomorphic encryption and computation is considered post-quantum secure. In the age of mobile phones with multiple lens and advanced software for night vision photography, why are they not equipped with acceleration to support secure and privacy friendly applications?

Acknowledgements. This work was supported in part by NSF grant 1704176 and NSF grant 2028190.

References

1. Aislelabs. https://www.aislelabs.com/
2. Apple iPhone 5S: technical specification. https://www.gsmarena.com/
3. iBeacon and Battery Drain on Phones: a technical report (2014). https://www.aislelabs.com/reports/ibeacon-battery-phones/
4. iBeacon Battery Drain on Apple vs Android: a technical report (2014). https://www.aislelabs.com/reports/ibeacon-battery-drain-iphones/
5. The Hitchhikers Guide to iBeacon Hardware: a comprehensive report by Aislelabs (2015). https://www.aislelabs.com/reports/beacon-guide/
6. CoEpi: Community epidemiology in action (2020). https://github.com/Co-Epi/
7. Covid Watch: slowing the spread of infectious diseases using crowdsourced data (2020). https://github.com/covid19risk
8. CovidSafe (2020). https://github.com/CovidSafe
9. Decentralized privacy-preserving proximity tracing (2020). https://github.com/DP-3T/documents
10. MIT PrivateKit (2020). https://privatekit.mit.edu/
11. PACT: private automated contact tracing (2020). https://pact.mit.edu/
12. PrivateKit: SafePaths (2020). https://privatekit.mit.edu/
13. ROBust and privacy-presERving proximity tracing protocol (2020). https://github.com/ROBERT-proximity-tracing/documents

14. Safetrace API: Privacy-first contact tracing (2020). https://safetraceapi.org/
15. WeTrace, a privacy focused mobile COVID-19 tracing App (2020). https://github.com/WeTrace-ch/WeTrace
16. Alderson, E.: Aarogya Setu: The story of a failure (2020). https://medium.com/@fs0c131y/aarogya-setu-the-story-of-a-failure-3a190a18e34
17. Altuwaiyan, T., Hadian, M., Liang, X.: EPIC: efficient privacy-preserving contact tracing for infection detection. In: 2018 IEEE International Conference on Communications, pp. 1–6. IEEE (2018). https://doi.org/10.1109/ICC.2018.8422886
18. Bay, J., et al.: BlueTrace: a privacy-preserving protocol for community-driven contact tracing across borders (2020). https://bluetrace.io/
19. Becker, J.K., Li, D., Starobinski, D.: Tracking anonymized bluetooth devices. PoPETs **2019**(3), 50–65 (2019). https://doi.org/10.2478/popets-2019-0036
20. Bell, J., Butler, D., Hicks, C., Crowcroft, J.: TraceSecure: towards privacy preserving contact tracing. CoRR abs/2004.04059 (2020). https://arxiv.org/abs/2004.04059
21. Berke, A., Bakker, M., Vepakomma, P., Larson, K., Pentland, A.S.: Assessing disease exposure risk with location data: a proposal for cryptographic preservation of privacy (2020)
22. Cao, X., Moore, C., O'Neill, M., Hanley, N., O'Sullivan, E.: High-speed fully homomorphic encryption over the integers. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 169–180. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44774-1_14
23. Chan, J., et al.: PACT: privacy sensitive protocols and mechanisms for mobile contact tracing. CoRR abs/2004.03544 (2020). https://arxiv.org/abs/2004.03544
24. Google, Apple: privacy-preserving contact tracing (2020). https://www.apple.com/covid19/contacttracing/
25. Government, S.: TraceTogether (2020). https://www.tracetogether.gov.sg/
26. Group, T.C.: TPM2.0 mobile reference architecture, December 2014. https://trustedcomputinggroup.org/resource/tpm-2-0-mobile-reference-architecture-specification/
27. Gvili, Y.: Security analysis of the COVID-19 contact tracing specifications by Apple Inc. and Google Inc., Cryptology ePrint Archive, Report 2020/428 (2020). https://eprint.iacr.org/2020/428
28. Israel Ministry of Health, I.M.: Hamagen: COVID-19 exposure prevention app (2020). https://github.com/MohGovIL/hamagen-react-native
29. North Macedonia Ministry of Health, N.M.M.: StopKorona! (2020)
30. Järvinen, K., Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Garbled circuits for leakage-resilience: hardware implementation and evaluation of one-time programs. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 383–397. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_26
31. Jayet-Griffon, C., Cornelie, M., Maistri, P., Elbaz-Vincent, P., Leveugle, R.: Polynomial multipliers for fully homomorphic encryption on FPGA. In: International Conference on ReConFigurable Computing and FPGAs, ReConFig, pp. 1–6. IEEE (2015). https://doi.org/10.1109/ReConFig.2015.7393335
32. National Informatics Centre, I.: Arogya Setu (2020). https://www.mygov.in/aarogya-setu-app/
33. Organization, W.H.: What is contact tracing and why is it important? (May 2017)
34. Ministry of Digital Affairs of Polland, M.: ProteGo Safe (2020)
35. Norwegian Institute of Public Health, N.I.: Smittestopp app (2020)

36. Reed, H.: Digital contact tracing and alerting vs exposure alerting (22 Apr 2020). https://harper.blog/2020/04/22/digital-contact-tracing-and-alerting-vs-exposure-alerting/
37. Reichert, L., Brack, S., Scheuermann, B.: Privacy-preserving contact tracing of COVID-19 patients. Cryptology ePrint Archive, Report 2020/375 (2020). https://eprint.iacr.org/2020/375
38. Riazi, M.S., Laine, K., Pelton, B., Dai, W.: HEAX: an architecture for computing on encrypted data. In: ASPLOS 2020: Architectural Support for Programming Languages and Operating Systems, pp. 1295–1309. ACM (2020). https://doi.org/10.1145/3373376.3378523
39. Saputro, N., Akkaya, K.: Performance evaluation of smart grid data aggregation via homomorphic encryption. In: 2012 IEEE Wireless Communications and Networking Conference, pp. 2945–2950. https://doi.org/10.1109/WCNC.2012.6214307
40. Seri, B., Vishnepolsky, G.: BlueBorne (2017)
41. Seri, B., Vishnepolsky, G., Zusman, D.: BleedingBit: the hidden attack surface against BLE chips (2019). https://info.armis.com/rs/645-PDC-047/images/Armis-BLEEDINGBIT-Technical-White-Paper-WP.pdf
42. Tang, Q.: Privacy-preserving contact tracing: current solutions and open questions. Cryptology ePrint Archive, Report 2020/426 (2020). https://eprint.iacr.org/2020/426
43. Tjell, K., Gundersen, J.S., Wisniewski, R.: Privacy preservation in epidemic data collection. CoRR abs/2004.14759 (2020). https://arxiv.org/abs/2004.14759
44. Trieu, N., Shehata, K., Saxena, P., Shokri, R., Song, D.: Epione: lightweight contact tracing with strong privacy. CoRR abs/2004.13293 (2020). https://arxiv.org/abs/2004.13293
45. Vaudenay, S.: Analysis of DP3T. Cryptology ePrint Archive, Report 2020/399 (2020). https://eprint.iacr.org/2020/399
46. Vaudenay, S.: Centralized or decentralized? the contact tracing dilemma. Cryptology ePrint Archive, Report 2020/531 (2020). https://eprint.iacr.org/2020/531
47. Wang, W., Huang, X.: FPGA implementation of a large-number multiplier for fully homomorphic encryption. In: 2013 IEEE International Symposium on Circuits and Systems, pp. 2589–2592. https://doi.org/10.1109/ISCAS.2013.6572408