Christopher Ellis*, Haohuang Wen, Zhiqiang Lin, and Anish Arora

Replay (Far) Away: Exploiting and Fixing Google/Apple Exposure Notification Contact Tracing

Abstract: Digital contact tracing offers significant promise to help reduce the spread of SARS-CoV-2 and other viruses. Google and Apple joined together in 2020 to create the Google/Apple Exposure Notification (GAEN) framework to determine encounters with anonymous users later diagnosed COVID-19 positive. However, as GAEN lacks geospatial awareness, it is susceptible to geographically distributed replay attacks. Anonymous, low-cost, crowd-sourced replay attack networks deployed by malicious actors (or far away nation-state attackers) who utilize malicious (or innocent) users' smartphones to capture and replay GAEN advertisements can drastically increase false-positive rates even in areas that otherwise exhibit low positivity rates. In response to this powerful replay attack, we introduce GAEN$^+$, a solution that enhances GAEN with geospatial awareness while maintaining user privacy, and demonstrate its ability to effectively prevent geographically distributed replay attacks.

Keywords: digital contact tracing, Google Apple Exposure Notification framework, Bluetooth Low Energy, geospatial index, H3, replay attack, Android, iOS, COVID-19

DOI 10.56553/popets-2022-0130
Received 2022-02-28; revised 2022-06-15; accepted 2022-06-16.

1 Introduction

The COVID-19 pandemic continues to evolve and spread two years since its outbreak in early 2020, claiming over six million lives worldwide by May 2022 [1]. Contact tracing remains as one of the early identified countermeasures to reduce the spread and overall impact of viruses such as SARS-CoV-2. To reduce time and cost intensive resources required by contact tracing solely performed by humans, numerous digital contact tracing (DCT) protocols and smartphone apps have been developed [2]. These protocols commonly utilize native smartphone features, such as Bluetooth, WiFi (e.g., WiFiTrace [3]), GPS (e.g., SafePaths [4]), acoustic signals (e.g., ATurf [5] and NOVID [6]), or QR code scanning (e.g., CrowdNotifier [7]) to provide the underlying mechanisms for smartphone apps to determine encounters with other individuals and ultimately notify of potential exposure to positive diagnosed users.

Compared to other wireless technologies, Bluetooth Low Energy (BLE) predominately enables numerous DCT protocols, such as BlueTrace [8], Temporary Contact Numbers (TCN) Protocol [9], Pan-European Privacy Preserving Proximity Tracing (PEPP-PT) [10], PPContactTracing [11], and Decentralized Privacy-Preserving Proximity Tracing (DP-3T) [12]. BLE is an attractive enabling technology due to its relatively low power consumption and its ability to facilitate proximity awareness without using additional sensors or location data. DCT protocols relying solely on BLE are characterized as more decentralized and privacy preserving because they detect proximity between devices to determine encounters as opposed to uploading precise location coordinates to a central server. However, the design and implementation is also a key factor for BLE-based privacy preserving decentralized protocols; otherwise, privacy information can still be leaked [13].

Google and Apple joined forces to create the Google/Apple Exposure Notification (GAEN) framework [14, 15] based heavily on DP-3T, and provided SDKs for approved public health authorities to develop and publish smartphone contact tracing apps. As a result, GAEN-powered apps are widely available, operate more efficiently in the background, and offer a unified protocol to enable communication between the two typically competing, widely adopted platforms with

***Corresponding Author: Christopher Ellis:** The Ohio State University, E-mail: ellis.729@osu.edu
Haohuang Wen: The Ohio State University, E-mail: wen.423@osu.edu
Zhiqiang Lin: The Ohio State University, E-mail: zlin@cse.ohio-state.edu
Anish Arora: The Ohio State University, E-mail: anish@cse.ohio-state.edu

a combined near 99% of the mobile OS global market share [16].

At a high level, GAEN securely generates pseudorandom, ephemeral identifiers and broadcasts them over BLE for nearby participating smartphones to capture. Later, these identifiers are rederived from a key shared anonymously by a positively diagnosed user and matched with previously captured identifiers. These mechanisms for encounter determination and significance occurring on users' devices, with a central server only to relay pseudorandom keys, make GAEN a largely decentralized protocol.

While GAEN attempts to balance the tradespace between data-utility and data-privacy to preserve user privacy and increase adoption rates, this approach also introduces weaknesses. In addition to other attacks, numerous researchers have shown GAEN is susceptible to distributed relay and replay attacks [17–21]. This is particularly due to the wireless broadcasting nature of BLE advertisements and the lack of adequate context, such as geospatial awareness, in the protocol itself. Effectively, a malicious actor can capture a legitimate GAEN advertisement from an honest user and replay it anywhere in the world. If another honest user captures replayed advertisements originating from a user who is later diagnosed positive for COVID-19, they may receive a false-positive exposure notification even though the two were not in close proximity.

More critically, the geographically distributed replay attacks can be launched by "far away" nation-state actors with advanced capabilities for deploying botnets [22] through compromised smartphones and BLE IoT devices. If adapted as zero-click malware, innocent users across the entire world may unknowingly contribute or fall victim to a larger attack aimed to destroy public trust and confidence in DCT protocols, and bring negative consequences to individuals' and collective society's daily life.

Previous works have specifically addressed the replay vulnerability in GAEN [23–25]. For example, SpreadMeNot [26] offers a public-private key distribution solution that diverges significantly from GAEN. Other researchers offer modifications that arguably fit within the GAEN framework. In particular, Raskar et al. present a recommendation for adding global location to GAEN [27], suggesting GPS context to be included as an encrypted payload in broadcast GAEN advertisements. While this recommendation appears to be a strong candidate against replay attacks, among other issues that we discuss later, it risks decreased adoption from public perception around GPS locations being continuously broadcast to others, even if encrypted. Another recent work, *ACTGuard* [28] recommends a third-party app and server to ultimately verify two users' identifiers are broadcast at the same time and location through relaying one-way hashes upon encounter and calculation of encounter to a remote server. However, this solution seems to add unnecessary complexity to an otherwise simple protocol.

Contributions. The main contribution of this paper is our proposal of GAEN$^+$ (§4), an elegant variant of the GAEN framework that increases its resiliency against a geographically distributed (far away) replay attack by introducing geospatial awareness. After recognizing the weaknesses of the existing GAEN protocol and the proposed solutions from Raskar et al. [27] and *ACTGuard* [28], we propose a solution that does not require substantial modifications, add any sensitive location data to the transmitted protocol fields, or introduce additional infrastructure. Instead, GAEN$^+$ slightly modifies the existing key derivation to include location context provided by a hierarchical geospatial index while still preserving bi-directional anonymity. The source code of our defense has been released at https://github.com/OSUSecLab/GAENPlus.

2 Background

GAEN Specification. GAEN is heavily based on the decentralized, privacy-preserving proximity tracing system, namely, DP-3T [12]. The primary enabling technology is the use of smartphones' BLE chipsets to continuously broadcast and capture pseudorandom ephemeral tokens or identifiers. Captured identifiers are later compared against anonymous identifiers derived from keys provided by others who are diagnosed positive for COVID-19. The occurrence of an encounter is determined on a user's smartphone, anonymous keys are shared with other users through a central server that does not store personally identifiable information (PII), and the significance of an encounter is again determined on each individual's device. These qualities make GAEN largely a decentralized protocol.

While many other DCT protocols are implemented entirely in a standalone smartphone app, GAEN is provided as an SDK framework for Android and iOS app developers at approved public health authorities to utilize, allowing the core cryptographic and BLE functionality implemented at the OS level. The OS is considered trusted and therefore reasonably assumed capable of secure storage of derived keys to maintain privacy.

In the following, we summarize the GAEN framework cryptography and BLE specifications [29, 30] to provide sufficient context for the attack analysis and countermeasure recommendations.

Pseudorandom Key Derivations. Once a user has downloaded and confirmed the installation of a GAEN-powered app from Google Play Store or Apple App Store, a Temporary Exposure Key (TEK) is generated daily using a cryptographic random number generator function (CRNG):

$$\mathsf{TEK_i} \leftarrow \mathsf{CRNG}(16) \tag{1}$$

and stored securely, along with its creation time interval i, for 14 days on the user's device. The current day's TEK is then combined with a NULL salt value and the static string "EN-RPIK" to generate a 16-byte Rolling Proximity Identifier Key (RPIK) through an HMAC Key Derivation Function (HKDF):

$$\mathsf{RPIK_i} \leftarrow \mathsf{HKDF}(\mathsf{TEK_i},\ \mathsf{NULL},\ \text{“EN}-\text{RPIK”},\ 16) \tag{2}$$

The current RPIK is combined with 16-bytes of padded data, consisting of the static string "EN-RPI" and a discrete time interval value, as input to a symmetric encryption algorithm, AES128. A 32-bit Exposure Notification Interval Number (ENIN) is derived from 10 minute windows starting from the Unix Epoch and therefore allow all participants to use the same values for key derivation:

$$\mathsf{ENIN_x} \leftarrow \mathsf{ENIN}(\mathsf{UnixTimeStamp}) \leftarrow \frac{\mathsf{UnixTimeStamp}}{60 \times 10} \tag{3}$$

to form TEK and ENIN pairs. Additionally, with six NULL bytes to form padding:

$$\mathsf{PadData_j} \leftarrow (\text{“EN}-\text{RPI”},\ \mathsf{NULL},\ \mathsf{ENIN_j}) \tag{4}$$

the framework generates the ephemeral Rolling Proximity Identifier (RPI) for a given time interval, deriving 144 RPIs per TEK and day:

$$\mathsf{RPI_{i,j}} \leftarrow \mathsf{AES128}(\mathsf{RPIK_i},\ \mathsf{PadData_j}) \tag{5}$$

Ephemeral Payloads. A new 16-byte RPI is generated every 10–15 minutes, coinciding with the rotation of the broadcasting BLE MAC address by the OS to reduce the potential for identification and long-term tracking. A 4-byte Associated Encrypted Metadata (AEM) field is generated containing version information and transmission power to assist in distance calculation, as well as reserved bytes. An RPI and AEM is combined to form the service data portion of a BLE payload. The BLE advertisement includes a 0xFD6F service UUID that allows applications and chipset interfaces to apply filters. The complete BLE advertising payload is broadcasted several times per second.

Scanning. Every 2–5 minutes, the GAEN-powered app opportunistically enters a scanning mode for approximately 10 seconds. Each captured RPI is paired with its BLE Received Signal Strength Indicator (RSSI) value and timestamp before being securely stored in a database on the user's smartphone for 14 days. These captured RPIs never leave the smartphone.

Exposure Notification. Upon positive diagnosis, a user is provided a submission key from the app's governing public health authority and is given the option to upload Diagnosis Keys (DKs) to a centralized server managed by a public health authority. The set of DKs consists of a range of TEK and respective creation ENIN_i pairs, (TEK_i, i), stored on the device for up to 14 days. If a user is not positively diagnosed, the pairs do not leave their device. DKs from all positively diagnosed users who choose to share are aggregated on the server and sent periodically to other app users.

The app now uses the aggregated DKs to reproduce the RPIKs and subsequent RPIs. The RPIs derived from the anonymous TEKs are then compared to the RPIs captured and stored in the user device's database. Upon a successful match, the proximity and significance of an encounter is determined through an RSSI calculation [31]. If an encounter is considered significant, the user receives an exposure notification from the app with healthcare guidance.

Scenario. Figure 1 depicts a typical scenario between honest users Alice, Bob, and Charlie, where Alice and Bob exchange RPIs as they are within BLE range. Upon Alice's positive diagnosis, she anonymously uploads (TEK_i, i) pairs to the public health authority which combines them with other pairs to periodically distribute to Bob and Charlie to calculate potential exposures. Since Bob determines he was in close proximity anonymously with Alice, he receives an exposure notification.

While some GAEN-based apps ask users to optionally provide their phone number during initial signup to assist human contract tracing efforts, no other PII is required by or uploaded to the server. Therefore, the identity of the individual(s) for whom a user was exposed remains private.

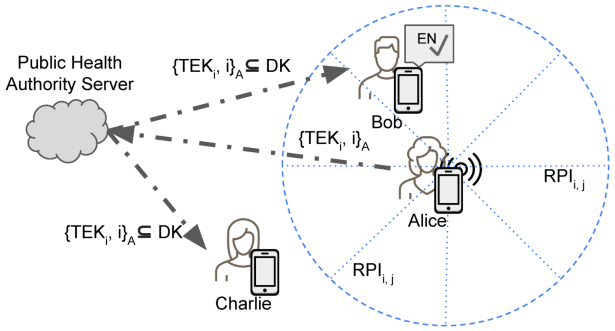


Fig. 1. Current GAEN framework operations among honest users

3 Attacks Against GAEN

DCT protocols such as GAEN are subject to various attacks [17, 20, 23, 32, 33], including (I) sniffing, (II) tracking, and (III) replay particularly due to their reliance on BLE for connectionless, wireless data transmission via broadcasting. In the following, we review these attacks and discuss countermeasures incorporated thus far into GAEN.

(I) Sniffing. As a wireless communication protocol, BLE is inherently susceptible to sniffing attacks. BLE utilizes adaptive frequency hopping to reduce collisions [34] for connection-oriented data transmission, which makes following stateful transmissions between two devices more difficult. However, the connectionless nature of advertisements in GAEN removes this barrier, requiring simply a BLE enabled device that follows the protocol specification to receive BLE advertisement payloads.

Therefore, there is no mitigation that GAEN can employ to address sniffing BLE advertisements. Doing so would logically contradict the very purpose of advertising presence or services. Consequently, the intended ease of BLE advertisement capture paves the way for other attacks, such as tracking and relay/replay.

(II) Tracking. Mobile apps that advertise an identifier are prone to tracking attacks [35] that aim to observe data trends and record an individual's location over time. Essentially, if an advertised attribute (e.g., the BLE MAC address or a payload value) is observed in one location at one time and again at another time, one can infer the device's trajectory, average speed, and location. This attack becomes more accurate the longer low entropy, identifiable attributes are broadcast.

GAEN attempts to counter tracking attacks by deriving a new RPI with each change in BLE Media Access Control (MAC) address already in place by the Android and iOS operating systems. Specifically in BLE parlance, the advertiser's BLE MAC address is set as Random, Private, and Non-resolvable [36]. This occurs approximately every 10–15 minutes to mitigate timing attacks that attempt to observe a strict change frequency to further support tracking. In effect, this type of ephemeral address hides the true, static, private MAC address of the BLE interface. This reduces the potential for tracking a repeatedly observed BLE MAC address or RPI. However, despite this mitigation, Corona-Sniffer [37] has demonstrated the feasibility for tracking a smartphone using a GAEN-powered app through a deployed network of geographically dispersed BLE receivers.

(III) Replay Attacks. Generally, a replay attack involves the capture and repeat transmission of data by a third-party. This is typically carried out by a malicious actor who intends to exploit a protocol's weakness to gain access to a system, poison a data set, or cause undesirable system effects.

Replay attacks may be mitigated at different levels of the network communication stack. For example, by using session IDs that cryptographically confirm the originating source, packet sequence numbers, or two-factor authentication. Effectively, these mitigations aim to provide connection state, timing, or other contexts to verify the validity of the transmission. However, mitigating replay attacks with broadcasted, connectionless protocols, such as BLE advertisements, becomes more difficult due to their nature of transmission and application.

As such, since GAEN transmits its RPIs within BLE advertisements, it is a prime target for replay attacks. While the developers of GAEN are aware of replay attacks, the ephemeral nature of RPIs only provides a weak temporal defense, leaving the geospatial vector open to attack.

- **Temporal Context.** GAEN limits the replay window of an RPI to approximately 2 hours. While RPIs are both originally created and later derived using established Unix timestamp intervals, GAEN adds a $\pm$ 2 hour buffer to increase validity likely to account for the fuzzyness of RPI rotation and time disparities among devices. RPIs are stored by interval when captured by a GAEN app, rendering RPIs replayed outside the interval $\pm$ 2 hours invalid.

  However, an RPI and timestamp interval pair is 20 bytes total, and therefore can be quickly transmitted across the Internet with typical smartphone data rates at high volume. For example, with a modest 5

Megabit/sec download rate, a smartphone can theoretically download 200 kilobytes (KB), or 10,000 captured (RPI, interval) pairs, in just 320 milliseconds. Consequently, distributed RPIs can be replayed by a malicious actor and captured by honest users within this wide window.

- **Geospatial Context.** GAEN relies on the limited range of BLE and the RSSI values at time of capture to calculate proximity to nearby smartphones. However, since the payload lacks geospatial awareness, an RPI is valid when broadcasted in any location. For example, an RPI generated and broadcasted in Columbus, OH is equally valid in Brooklyn, NY. This lack of geospatial context makes GAEN vulnerable to geographically distributed replay attacks that exploits its assumption of accurate proximity.

The GAEN Distributed Replay Attack. While it is well-known that GAEN is fundamentally subject to replay attacks [17–19, 21, 32], it is noteworthy to consider the ease of implementation, deployment, cost, scalability, and potential severity of their impact. We discuss these characteristics in this section and provide the attack scenario in Appendix A and its impact in Appendix B for readers desiring further background. Later, we evaluate our solution with GAEN$^+$ (§6) in terms of its effectiveness as a countermeasure against this distributed reply attack.

Baumgärtner et al. demonstrate in real-world scenarios that GAEN is susceptible to replay attacks, implementing a proof-of-concept with both smartphones and Raspberry Pis that capture, relay, and replay RPIs [19]. Beskorovajnov et al. observe DCTs in general that use broadcasting mechanisms that do not consider time, location, or other session information are subject to relay and replay attacks [38], further supported by Gvili [21] and Sun et al. [33].

An anonymous, crowd-sourced, geographically distributed replay attack network easily scales through smartphone apps and cloud services, without requiring specialized hardware. Like-minded malicious actors download a replay app designed to continually capture, transmit, receive, and replay crowd-sourced RPIs running in the background on their BLE enabled and Internet connected smartphones. After initial setup, they simply carry their smartphone on their person throughout their normal daily routine that now acts as a node in the distributed replay attack network.

Such a network can easily be deployed by a tech-savvy user with basic web and app development experience. To substantiate this claim, we developed a simple web application built with Python and hosted it on a basic, $5 USD/month, single-CPU cloud server with default configurations. This trivial setup easily accommodates approximately 100 concurrent connections to serve over 220,000 requests for 500 crowd-sourced RPIs over 10 minute interval windows.

Meanwhile, motivated nation-state actors may integrate the concepts into malware that develops a botnet, compromising smartphones and IoT devices connected to the Internet without user awareness. This hidden malware could utilize the same capture/replay functionality as a distributed app, having compromised the device with access to the underlying libraries at run-time. This effort is feasible given the numerous zero-click attacks against innocent users' smartphones from nation-state attackers (e.g. [39–41]).

4 Countermeasures

The absence of geospatial awareness exposes GAEN to geographically distributed replay attacks. If a replay attack network, as discussed in §3, is deployed and adopted by malicious actors, false-positive rates in otherwise low positivity rate areas would substantially increase. This can decrease public trust and adoption rates of GAEN and DCT applications in general. Therefore, GAEN must maintain public trust in order to increase adoption rates and provide its maximum utility for notifying users of exposure to positively diagnosed individuals.

To this end, we establish the following research goals and requirements to enhance GAEN:

- G_1**:** Retrieval of current location data without compromising user privacy (§4.1).
- G_2**:** Enhancement of GAEN with geospatial awareness to reduce the effect of replay attacks (§4.2).
- G_3**:** Incorporation of modifications without requiring significant rework of the protocol or significantly changing its performance or user experience (§4.3).

4.1 G_1: Retrieving Current Location

When a concept requires location context, GPS is typically the first solution considered. Fortunately, modern smartphones commonly feature GPS receivers that utilize satellite constellations for its current latitude and longitude. Bettinger demonstrates accuracy of GPS data from smartphones varies wildly depending on conditions, however, determines an average horizontal position accuracy of an iPhone 6 to be within 7–13m (23–43ft) and frequently under 2m (7ft), consistent

with other autonomous GPS receivers [42]. While this average distance exceeds the Center for Disease Control and Prevention's recommendation of maintaining $1.8m$ $(6ft)$ distance between individuals [43], it falls well within BLE range of approximately $91m$ $(300ft)$. However, this level of GPS accuracy may appear too pervasive to user privacy and plainly violates GAEN's Additional Terms that states apps cannot utilize precise locations [44]. Further, traditional GPS solutions typically consume significant amounts of power when active, resulting in noticeable battery drain and degraded user experience.

There are numerous alternative sources for location, including WiFi, Bluetooth, cellular networks, and the use of infrared and ultrasound frequencies [45]. Fortunately, both Android and iOS offer WiFi and cellular network positioning through coarse location services [46], which benefits our use case from the perspectives of adoption and adhering to GAEN's terms for developers. Additionally, these alternatives consume less power and are generally faster to retrieve results since the data is received from ground systems versus satellite constellations with GPS. Given these characteristics, WiFi and cellular positioning appear to be viable solutions for a smartphone to retrieve its current, approximate location. However, the challenge still remains how to use these coordinates in a meaningful way to provide geospatial awareness.

4.2 G_2: Geospatial Awareness

Now with an appropriate solution to retrieve location, the next goal involves applying this context to the current GAEN framework to reduce the impact of geographically-distributed replay attacks. Adding geospatial awareness to the protocol simply translates to having some concept of location to complement its temporal context. The most obvious way is to include GPS coordinates in the AEM field as part of the advertised payload, as recommended by Raskar et al. [27]. However, broadcasting precise or coarse GPS coordinates, even if encrypted, may leave GAEN susceptible to more public scrutiny. Further, GPS coordinates are point values and do not intrinsically provide any surrounding bounds to reduce the space to which an RPI is considered valid and still requires a calculation on the receiving smartphone. This inspires the use of predetermined area bounds, restricting RPI validity from the entire Earth to a more reasonable area.

Using Political Boundaries. Naturally, we first considered established and adopted conventions for geographic partitions, such as political boundaries, to include states, counties, and zip codes. In the United States, development of GAEN-powered applications are left up to state-level health authorities and governments. If each state uses their own central server to upload and exchange keys, this would provide up to 50 distinct geographically distinct zones.

However, the Association of Public Health Laboratories hosts a national key server that U.S. states can utilize to increase effectiveness during travel across state lines. As of May 2022, nearly half of the U.S. states are participating [47]. While a national server aims to ensure users in all states receive DKs despite their originating state, it also increases the space for which a geographically, wide-reaching replay attack is valid. However, this seems like a reasonable trade-off that can simply be addressed through a creative solution in the protocol itself. Further, given their non-uniformity of area and representation of population densities, state boundaries make for seemingly arbitrary divisions from protocol and statistical perspectives. Finally, using states as geographic bounds would simply prevent a malicious subscriber from receiving RPIs from other states with the simple counter of focusing malicious adoption on more concentrated areas of cases within state lines. These issues do not support using state boundaries as a geospatial component to mitigate replay attacks.

Counties and ZIP codes are the next intuitive alternatives that provide more granular geographic bounds compared to states. However, these still suffer from the same population density and size issues discussed above. Further analysis reveals an issue around encounters near borders where two users are only feet apart but technically in different zones. The reasonable solution is to include neighboring zones during RPI derivation from received TEKs. But accurate calculation becomes intuitively very difficult when considering the number of edges a county or ZIP may have. If we simply include neighboring counties, non-uniform size and population densities again make the results of this calculation vary wildly. A relatively small county may border a significantly larger neighbor which may drastically and inaccurately increase the geographic search space.

Using a Hierarchical Geospatial Index. The need to address zone size, population density, and cross-border calculation issues leads us to consider more uniform, hierarchical geographic data structures. The underlying recursive algorithms take latitude and longitude coordinates and output index values at various resolutions that are contained within each other, provid-

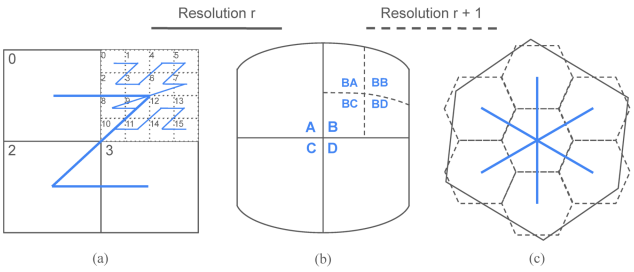


Fig. 2. Hierarchical geospatial indexing: (a) GeoHash Z-Curve encoding, (b) S2 geodesic squares, (c) H3 hexagons

ing their hierarchical characteristic. We next systematically examine a number of open-source geospatial index implementations, which include Geohash [48], Google's S2 Geometry library [49], and Uber's H3 [50]. The key differences between these solutions largely involve the shape of the geofenced area represented by the output value from provided coordinates, as illustrated in Figure 2, in addition to other defining characteristics.

GeoHash. GeoHash is a hierarchical, geospatial system that encodes latitude and longitude coordinates into alphanumeric strings. Using Z-curves [51], it effectively subdivides maximum resolutions of longitudinal coordinates from -180 to +180 degrees and latitudinal coordinates from -90 to +90 degrees for each successive bit pair. The higher or lower interval is chosen with bits 1 and 0, respectively. For example, a latitude value starting with 1 will make the current latitude interval 0 to +90, producing an effective error of ±45. This produces a sub-divisible collection of non-overlapping rectangles as precise as the number of bit-pairs.

While GeoHash is relatively straight forward, its simplicity introduces undesirable flaws. Generally, shared proximity between two points is easily determined by matching common prefixes. However, this is not true for points on either side of the 180 meridian due to its Mercator projection [52], where they would have different prefixes on each side. While this is an edge case, it represents an unnecessary limitation that is addressed by other available solutions. Further, GeoHash's geometrical representation is rectangular with varying areas depending on the latitude, due to its two-dimensional projection on the Earth's spherical shape. This results in varied error rates for proximity detection that again introduce an unnecessary limitation given alternatives.

Google's S2 Geometry. Google's S2 Geometry library overcomes GeoHash's geometric limitations by providing data overlaid to a three-dimensional sphere that better represents the Earth. Latitude and longitude are converted to indices representing geodesic squares of consistent area depending on the resolution. The spherical geometry approach also removes the discontinuity limitation around the 180 meridian. While S2 is an improvement over GeoHash, it still provides a less optimal solution compared to others when considering neighboring areas. This is due to its geodesic square zone representation, requiring consideration of both edge-to-edge and diagonal neighbors, eight in total. Further, distance calculation from a zone's center to its neighbors is non-uniform, as edge-to-edge neighbors have smaller distances than diagonal neighbors.

Uber's H3. Developed by Uber, the H3 hierarchical geospatial indexing library overcomes S2's geometric limitation from using a geodesic square representation by utilizing hexagonal areas instead. Hexagons provide the useful characteristic that all neighboring cells' centers are equidistant, removing S2's non-uniform distance calculations. Additionally, the number of nearest-neighbor cells to process is reduced from eight to six. Hexagons also impact the hierarchical overlap; rather than one resolution perfectly containing its child rectangles from a higher resolution, hexagons will have small overlaps which may provide interesting optimizations for neighbor calculation. H3 provides 16 levels of resolution that range from providing 122 unique indexes at approximately 4.2×10^6 km^2 at resolution 0 to 1.1×10^{13} unique indexes at approximately 4.3×10^{-5} km^2 [50].

Our analysis leads us to choose H3 as the most suitable geospatial index solution. While GeoHash and S2 perform well in typical cases, H3 is able to overcome more edge cases and provides a more uniform grid overlay of the Earth, thereby reducing the false-positive and false-negative rates from neighboring zone calculations.

4.3 G_3: Minimal GAEN Modifications

The final goal is to incorporate geospatial awareness in a manner that does not require drastic changes to the existing GAEN protocol, reduce privacy, or require additional third-party software, hardware, or services. Gvili remarks on the disadvantage and issues of unnecessary modifications [21], which further inspires an elegant solution for GAEN$^+$. Once again, including the fuzzy, bounded area in the AEM field presents an obvious first consideration. However, the property that two different smartphones are able to independently determine a shared, bounded area index prompts us to consider GAEN's key derivation algorithms (§2).

As shown in the specification background, GAEN creates three keys during its derivation scheme. TEKs are sent to the public health authority by users who opt-in upon positive diagnosis and derived RPIs are broadcasted for others to capture. This makes the RPIK particularly interesting because its placement between the TEK and RPI make it the only key that truly remains on the smartphone and is not exposed to other parties. *The RPIK is thus an ideal candidate to include a geospatial component to achieve the desired property.*

In effect, *using the smartphone's calculated geospatial index as the salt value — instead of just using NULL bytes during RPIK derivation* — adds geospatial awareness through a bi-layer indirection. No location context is directly included in the RPI payload nor in the TEKs uploaded to the central server. As a result, this key differentiator from other solutions maintains GAEN's current level of user privacy.

By using the geospatial index as the salt, the static string "EN-RPIK" is left intact and therefore still adhering to the GAEN specification and its ability to create an Associated Encrypted Metadata Key (AEMK). Therefore, our enhanced protocol GAEN$^+$ introduces a Location-based Rolling Proximity Identifier Key (LRPIK) that includes the geospatial index, Loc_x, as its salt argument instead of NULL bytes as in the RPIK derivation in the original GAEN:

$$\mathsf{LRPIK_x} \leftarrow \mathsf{HKDF}(\mathsf{TEK_i},\ \mathsf{Loc_x},\ \text{“EN}-\text{RPIK”},\ 16); \quad (6)$$

GAEN$^+$ generates a single daily TEK as before. Now, the smartphone will periodically retrieve its coarse location coordinates and query a geospatial index to detect movement into a new cell and with each new RPI computation. A new LRPIK will be derived whenever the geospatial index query returns a new value, implying the user is in a new location. When the returned geospatial index matches the previous query return value, the LRPIK does not need to be derived again, implying the user is still in the same zone, reducing unnecessary calculations. Finally, the RPI is then derived from the current LRPIK_x and broadcasted as before with GAEN.

We now consider new scanning and RPI derivation behavior. As before, smartphones within BLE range will scan and receive nearby GAEN advertisements containing an RPI. Now, after a scan which results in at least one successful GAEN advertisement capture, the current geospatial index value, Loc_x, is stored on the smartphone's local, secure storage L. The smartphone will periodically download the DKs from the central server as before. Similar to advertising, each Loc_x in L will be used to derive a new LRPIK_x for each TEK. For each derived LRPIK_x, the set of RPIs is derived and compared with previously captured RPIs for a given ENIntervalNumber, as before. Further, since the pseudorandom TEK is still used as a HKDF parameter, the risk for RPI collision from two different locations remains consistent with GAEN at any resolution value.

Effectively, our enhancement defines geospatial bounds that a generated RPI is valid. If the RPI is broadcasted and received in the same bounds as it was generated, its derivation from a TEK provided by a positively diagnosed user will result in a match. Otherwise, a match will not occur. As a result, GAEN$^+$ can drastically reduce false-positives introduced by a distributed replay attack scenario reviewed in §3.

5 GAEN$^+$ Implementation

(I) Enabling Coarse Location Services. We utilize coarse location services for its numerous benefits over standard GPS for our particular use case. Compared to precise locations offered by standard GPS, coarse location services adheres to GAEN's Additional Terms, helps preserve privacy, and consumes less power by using location data from WiFi access points and cellular towers. Android provides resolution approximate to a city block [53] or 311 ft. [54]. For iOS, the *Significant-change Location Service* provides updates approximately every 1,600 ft. While this represents a 5x difference, we discuss future options for iOS later in §7.

The first modification requires users to grant app permissions for coarse location services. On Android, this includes adding the ACCESS_COARSE_LOCATION and ACCESS_BACKGROUND_LOCATION permissions to allow access to data while the app is in the background [53]. On iOS, this includes the *Significant-change Location Service* and *Always* authorization [46]. If a user disables location services or their smartphone has no cell service and is not connected to a WiFi network, a notification should appear that the contact tracing application is unable to operate, similar to the notification that currently appears when a user disables Bluetooth with a GAEN-powered app installed.

(II) Calculating Hierarchical Geospatial Index w/ H3. GAEN$^+$ features H3 as the hierarchical geospatial index to provide geospatial awareness. H3 is written in C and compiled as a shared library with bindings available in Java and other languages [55], making it easily portable to both Android and iOS. The library size is approximately 200 kilobytes and therefore, when

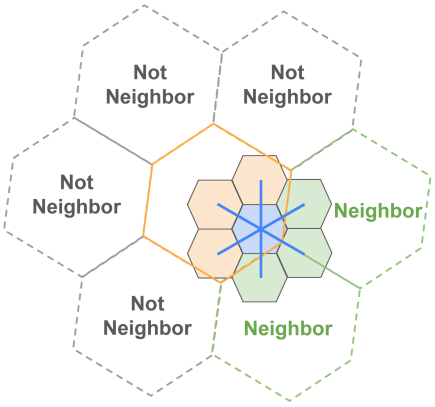


Fig. 3. Neighboring cells of current location at resolution 11, where orange cells share the same parent cell and green cells belong to a different parent and therefore used in neighboring cell LRPIK derivation.

compared to the average size of smartphone apps in the tens of megabytes [56], the additional size is negligible.

H3 overlays a hexagon grid to provide coverage for the entire Earth with 16 resolutions [57]. The coarse latitude and longitude coordinates provided by the location services will be fed into an H3 function to calculate an index, given a resolution: `H3Index geoToH3(double lat, double lng, int res)`.

Neighboring Cells. Calculating LRPIKs based on a cell with discrete boundaries introduces the scenario where two users are within BLE range, but in two separate cells. We account for this edge case by adding an optimized neighboring cell check (Figure 3). By logging the visited cell index at one resolution higher than what is used to derive LRPIKs, GAEN$^+$ is able to determine which neighbors to check. *This optimization reduces the number of neighboring cell checks from six down to two.* Retrieving a child's parent cell is made trivial with the provided H3 function: `H3Index h3ToParent(H3Index h, int parentRes)`. Therefore, we recommend using resolution 11 for active location indexing and resolution 10 for LRPIK calculation. Resolution 10 provides an average hexagon edge length of approximately 66 meters (216 ft.) and provides over 33 billion hexagons with approximately 15 m^2 (162,000 $ft.^2$) of area. We discuss future work for dynamic resolutions in §7.

(III) Modification of GAEN Algorithms. Our proposed solution requires minimal changes to the GAEN cryptography and BLE specifications. Specifically, (1) RPIK derivation must include the parent cell of the current H3 index cell as a salt value, now referred to as an LRPIK. This includes deriving the LRPIK to both create RPIs to broadcast and match for exposure notification.

Algorithm 1: GAEN$^+$ RPI Matching

```
1  LOCS ← GetLocFromLocalSecureStorage();
2  RPIS ← GetRPIFromLocalSecureStorage();
3  DKS ← GetLatestDKFromPublicServer();
4  for dk in DKS do
5      checkedLocs ← Set();
6      for locVisited in LOCS[dk.i] do
7          locParent ← h3ToParent(locVisited, 10);
8          if locParent in checkedLocs then continue;
9          nearLocs[] ← nearestNeighbors(locVisited);
10         dkLocs ← Set(locParent);
11         for neighborLoc in nearLocs do
12             parent ← h3ToParent(neighborLoc, 10);
13             dkLocs.add(parent);
14         end
15         for loc in dkLocs do
16             if loc in checkedLocs then continue;
17             checkedLocs.add(loc);
18             lrpik ←
                 HKDF(k.TEK, loc, "EN − RPIK", 16);
19             for int in intervals[0..144] do
20                 rpi ← AES128(lrpik, dk.i + int);
21                 if rpi in rpis[dk.i] then
22                     // Significance Calculation
23                 end
24             end
25         end
26     end
27 end
```

(2) Storing the current geospatial index whenever an RPI is received to later use for matching. This value is stored in a daily set, kept for 14 days, and not paired with any additional information; therefore indices are only temporally precise to a day.

Algorithm 1 shows the addition of neighboring cells as candidates for LRPIK derivation at lines 9–14. The function nearestNeighbors takes an H3 index and returns an array of all neighbors of the same resolution (11) that are one cell distance away. Next, the parent at resolution 10 of each neighbor cell is added to a candidate list. The loop starting at line 15 drives the location component and matching, with an optimization to not calculate any already-checked locations for a particular DK.

Next, the framework must periodically retrieve the smartphone's coarse GPS coordinates to retrieve the current geospatial index. If the index value changes from the last query, it must then check to see if the parent index has also changed. If so, the framework generates a new LRPIK and RPI, changes the advertising BLE MAC address, and continues broadcasting the new GAEN advertisement, as presented in Algorithm 2.

Since the framework's SDK will only compile if provided a license granted to an approved public health authority [14], we use the reference code provided for Android [58] and implement algorithms for critical operations for deriving, broadcasting, and matching RPIs in Java and C++, facilitated by Java Native Interface (JNI).

Algorithm 2: GAEN$^+$ Broadcasts

```
while true do
    lat, lng ← GetCoarseLocation();
    latestGeoIndex ← geoToH3(lat, lng, 11);
    if curGeoIndex != latestGeoIndex then
        curGeoIndex ← latestGeoIndex;
        curParent ← h3ToParent(curGeoIndex, 10);
        if curParent != latestParent then
            curParent ← latestParent;
            SetRandomBLEAddress();
            rpik ← HKDF(TEK_i, curParent, "EN − RPIK", 16);
        end
    end
    if 10–15 minutes has elapsed then
        SetRandomBLEAddress();
        enin ← ENIntervalNumber(now);
        rpi ← AES128(rpik, ENIN);
    end
    BroadcastGAENPayload(rpi);
end
```

Replay Attack Mitigated by GAEN$^+$. The mitigated attack scenario involves honest users Alice, Bob, and Charlie who have a GAEN$^+$ powered app installed on their smartphone. As illustrated in Figure 4, Mallory and Malaki are malicious actors who have subscribed to the distributed replay attack network with the replay app installed on their smartphone.

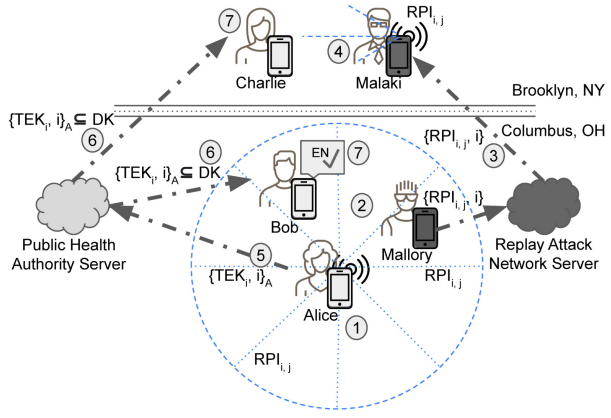


Fig. 4. Our proposed GAEN$^+$ adds a geospatial component to decrease impact of distributed, crowd-sourced replay attacks

1. Alice generates a TEK daily and derives a LRPIK for her current location in Columbus, OH. She creates a new RPI every 10–15 minutes and broadcasts over BLE, periodically checking if her location has changed to warrant new LRPIK and RPI derivation.
2. Bob and Mallory are within Alice's BLE advertising range, capture Alice's GAEN advertisement, and extract the RPI. Bob stores his current geospatial index and the RPI for 14 days. Mallory calculates the current ENIntervalNumber i and sends the captured (RPI, i) pair to the replay attack network.
3. Malaki, in Brooklyn, NY, periodically receives (RPI, i) pairs, including Alice's. He continuously replays all RPIs generated in the current interval.
4. Charlie is nearby Malaki and receives Alice's replayed RPI. He stores his current geospatial index and Alice's RPI for 14 days.
5. A few days later, Alice receives a positive diagnosis for COVID-19 and elects to submit her (TEK, i) pairs to the public health authority's server.
6. Bob and Charlie receive DKs from the public health authority server. Using each stored geospatial index, they derive the LRPIKs and RPIs for each TEK. Bob finds a match between Alice's RPI and a derived RPI from an anonymous TEK. Charlie does not, as he was not in Columbus, OH.
7. Bob is correctly notified of exposure and Charlie is correctly not notified of exposure.

6 Evaluation

We now evaluate GAEN$^+$ across three different aspects: effectiveness of preventing distributed replay attacks (§6.1), impact of its resolution parameter (§6.2), and overhead including battery consumption, network and storage requirements, and computation (§6.3). We consider various location and user activity profiles during our evaluation. Specifically, location profiles indicate the amount of individuals in a given cell at sparse, medium, and dense collectives. Low, medium, and high activity profiles represent the amount of locations an individual visits in a particular day. Experiments were conducted on a Google Pixel XL running Android 10.0.

6.1 GAEN$^+$ Effectiveness

Our evaluation of GAEN$^+$ effectiveness began with simulating a set of experiments and comparing the results with the original GAEN framework. The experiment setup and results are presented in Table 1.

We designed three locations and three activity profiles to combine and generate various simulations of user density and movement scenarios for a particular day. Specifically, Sparse, Medium, and Dense location profiles account for 100, 200, and 400 users respectively. Low, Medium, and High activity profiles translate to a user visiting 5, 10, and 20 locations respectively. A set of latitude and longitude coordinates for 50 locations spread across the globe serve as the potential locations.

RPI Generation				RPI Receiving & Matching							
Location Profile	# User	# TEK	# RPI Gen.	Activity Profile	# Loc. Visited	# RPI Recv.	# Matched RPI (GAEN$^+$)	Time (s) (GAEN$^+$)	# Matched RPI (GAEN)	Time (s) (GAEN)	% False Pos. (GAEN)
Sparse	100	1,400	201,600	*Low*	5	6,048	620	1.238	6,048	0.374	89.75%
Medium	200	2,800	403,200	*Low*	5	12,096	1,264	2.468	12,096	0.727	89.55%
Dense	400	5,600	806,400	*Low*	5	24,192	2,162	6.333	24,192	1.505	91.06%
Sparse	100	1,400	201,600	*Medium*	10	6,048	1,200	2.177	6,048	0.371	80.16%
Medium	200	2,800	403,200	*Medium*	10	12,096	1,930	4.578	12,096	0.753	84.04%
Dense	400	5,600	806,400	*Medium*	10	24,192	4,621	12.742	24,192	1.519	80.90%
Sparse	100	1,400	201,600	*High*	20	6,048	1,960	3.811	6,048	0.379	67.59%
Medium	200	2,800	403,200	*High*	20	12,096	3,896	8.007	12,096	0.733	67.79%
Dense	400	5,600	806,400	*High*	20	24,192	8,973	21.428	24,192	1.529	62.91%

Table 1. Effectiveness evaluation of GAEN$^+$ compared with original GAEN in RPI matching across Low, Medium, and High activity profiles, and Sparse, Medium, Dense user populations.

We derive 1,400, 2,800, and 5,600 TEKs for Sparse, Medium, and Dense profile respectively over a 14-day period, leading to 201,600, 403,200, and 806,400 total RPIs generated in a single day. These RPIs are generated at the 50 locations at different time intervals, which simulates an attack scenario where a malicious subscriber replays RPIs captured from other locations.

Next, for each activity profile, we assume 3% of the generated RPIs are received by an honest user (i.e., a single replay attack victim). We then randomly generate the locations this particular user has visited, based on the activity profile, out of the 50 locations. The user's smartphone then receives the generated TEKs/DKs, derives LRPIKs with its visited locations, and attempts to match derived RPIs with its captured RPIs. The H3 library is used to calculate the geospatial index of the locations at resolution 11, and derive LRPIKs at resolution 10 as introduced in §5.

The results of RPI matching are shown in the last five columns of Table 1, with GAEN$^+$ accurately confirming the approximate ratio of visited locations versus received RPIs for a given profile. For example, 620 matched RPIs out of 6,048 received RPIs for a Sparse/Low scenario having visited 5 out of 50 locations in total. Further supporting, 8,793 matched RPIs out of 24,192 received RPIs for a Dense/High scenario having visited 20 out of 50 locations in total.

In contrast, GAEN inaccurately matched all RPIs received from every profile combination, which includes RPIs from locations the user did not visit. The false positive rate exhibited by GAEN effectively equals the percentage of remaining RPIs from locations the user did not visit, or in other words, received from a replay attack. Our results demonstrate the ability for GAEN$^+$ to accurately determine valid captured RPIs, further indicating its general effectiveness against geographically distributed replay attacks.

6.2 GAEN$^+$ Resolution Parameter

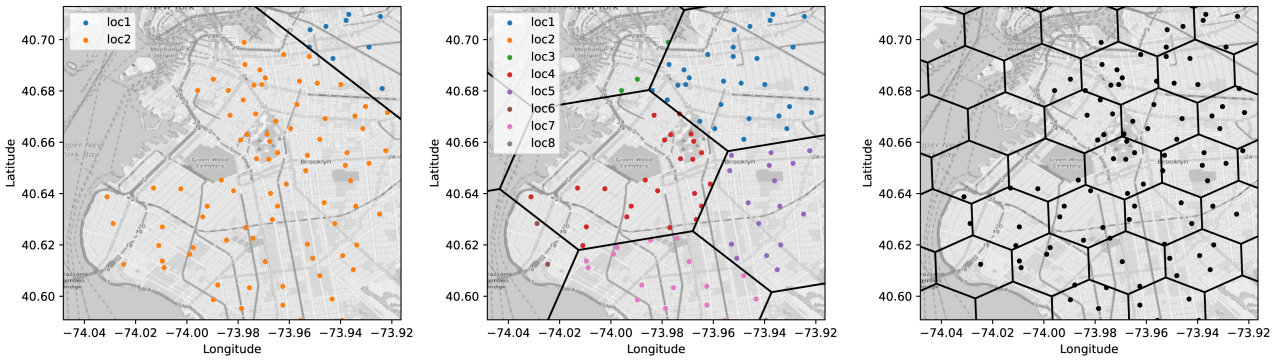
To show how resolution affects geospatial indexing, we randomly selected 100 geopoints in Brooklyn, NY, and show their distribution within H3 cells at resolution 4, 6, and 7, respectively in Figure 5a, Figure 5b, and Figure 5c. As shown, the 100 geopoints are respectively mapped to 2, 8, and 26 H3 hexagon-shaped cells.

To further evaluate resolution's impact on computation time and accuracy for matching in GAEN$^+$, we selected the Medium location profile and Medium activity profile as an average case, using the geopoints from Brooklyn. A set of 10 out of 100 visited locations (for Medium activity) was first randomly generated to be consistent across each experiment, for resolutions 4 through 11 as presented in Table 2.

Res.	# H3 Cell	# RPI Gen.	RPI Receiving & Matching		
			# Recv	# Matched	Time (s)
4	2	403,200	12,096	12,096	0.939
5	3	403,200	12,096	12,096	0.946
6	8	403,200	12,096	12,096	2.194
7	26	403,200	12,096	10,901	3.858
8	86	403,200	12,096	3,487	4.753
9	97	403,200	12,096	1,559	4.688
10	99	403,200	12,096	1,454	4.256
11	100	403,200	12,096	1,454	4.547

Table 2. Impact of resolution parameter on GAEN$^+$

For each experiment, we count the number of RPIs matched and observe the computation time. As shown in Table 2, the number of matched RPIs at relatively low resolutions 4, 5, and 6 equals the number received. Under the right conditions (i.e., visited locations that can be contained within a single cell and neighbors) these low resolutions effectively mimic the original GAEN framework, where the entire Earth can be considered one cell. Indeed, the number of cells containing the geopoints is 2, 3, and 8, respectively. The high matching rate is also due to the high frequency of H3 index colli-



(a) Resolution 4. **(b)** Resolution 6. **(c)** Resolution 7.

Fig. 5. Distribution of 100 H3-indexed geo-points under different resolutions.

sions of two different locations. For instance, under resolution 4 (Figure 5a), geopoints that are relatively far from each other will still be mapped to the same H3 index, given the hexagon area is significantly larger. However, the chance of collision between these same points at higher resolutions is significantly reduced and thus the number of matches converges to 1,454 at resolution 10. Note, this result is approximately 12% of the received RPIs, where the additional 2% is due to the realistically uneven distribution of geopoints and inclusion of neighboring cells which allows some variable amount of false-positives, discussed further in §7.

As shown in Table 2, the computation time gradually increases with some variability for each resolution and tapers off at approximately 4.5 seconds, which aligns with observations of the Medium/Medium scenario in Table 1. In summary, higher resolution parameters in GAEN$^+$ offer finer-grain protection against replay attack to eliminate more false positives.

6.3 GAEN$^+$ Overhead

Battery Consumption. We evaluate GAEN$^+$ for power requirements compared to original GAEN. Figure 6 depicts RPI derivation and advertisement broadcasting battery consumption for two hours, updating location and LRPIK every 60 seconds. Results reveal GAEN$^+$ is within 1% on average of GAEN even with a relatively high-activity movement profile. The remaining battery percentage decreases linearly, as expected given the consistent operations. While the precision of battery percentage is limited to integer values, this level of granularity still shows the minor difference between GAEN$^+$ and GAEN. If more precise introspection was available, this difference might actually prove to be less than 1%. Further, a less active movement profile would exhibit battery consumption even more closer to GAEN.

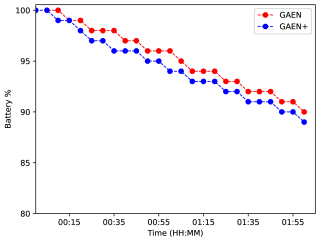
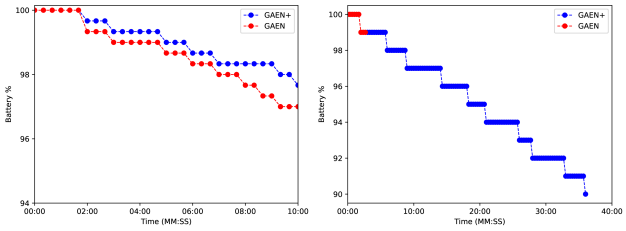


Fig. 6. GAEN$^+$ vs. GAEN: Battery Consumption for RPI Generation and Broadcasting

Figure 7 measures battery consumption of RPI matching for both GAEN and GAEN$^+$ over time. To clearly show the difference, we conducted the experiments under two stressful environment settings: (1) we repeated the matching process for 10 minutes, and (2) we ran the RPI matching process for 500 iterations. The corresponding results are respectively shown in Figure 7a and Figure 7b. Note that the experiment was performed under Medium location profile and Medium activity profile as an average case. Figure 7a shows the difference of battery percentage between GAEN and GAEN$^+$ mostly stays within 1% within 10 minutes and the consumption of GAEN$^+$ is surprisingly lower than the original GAEN. This is because GAEN actually runs larger number of matching iterations than GAEN$^+$ within a fixed time window, and thus GAEN has to invoke more JNI binding calls that cause additional overhead. Figure 7b indicates that GAEN and GAEN$^+$ have the same amount of battery consumption for the first 2 minutes, but GAEN$^+$ needs additional 36 minutes to complete the remaining iterations. Overall, GAEN and GAEN$^+$ have almost the same battery consumption overhead when given the same amount of time.



(a) 10 Minutes Matching **(b)** 500 Matching Iterations

Fig. 7. GAEN$^+$ vs. GAEN: Battery Consumption for RPI Matching

Network & Storage. GAEN$^+$ does not modify the original GAEN upload or download process of DKs and therefore network traffic remains unchanged and without additional burden. GAEN$^+$ requires storing geospatial index values for which an RPI was successfully captured over a 14-day period. For example, Brooklyn, NY and its surrounding areas has 823,543 cells at resolution 10. At 8 bytes per cell, this results in only 6.5MB of storage if every cell was visited and captured an RPI within a 14-day period. In reality, the number of visited cells will be substantially lower, resulting in minimal storage requirements.

Computation Time. The tables and figures in this section demonstrate the effect various movement profiles, location user densities, and the resolution parameter has on the GAEN$^+$ computation time for RPI matching. This aligns with expectations given the increased algorithm complexity introduced by the LRPIK derivation loop. This is a direct result of the resolution parameter, where a lower resolution produces larger cells and therefore less locations to derive LRPIKs, and vice versa for higher resolutions. Further, the introduction of neighboring cells to the matching algorithm (1) also adds to computation time a theoretical maximum of two times the number of locations a user has visited and received an RPI. Observing concrete results in Table 1, we first note GAEN remains consistent for each scenario type and correlates to the number of received TEKs and RPIs. GAEN$^+$ however shows additional computation time approximately linear with number of locations visited for each activity profile. Finally, we observe the matching time for GAEN$^+$ increases non-linearly with larger (Figure 7) versus smaller (Table 1) TEK quantities, due to complexity and potentially to caching, hardware, or operating system scheduling limitations.

7 Discussion

Replay Attack Effectiveness. GAEN$^+$ does not fully remove the ability for replay attacks to target the same cell a legitimate RPI was captured. For example, a node could capture RPIs in a high traffic area and immediately replay them in the same position. Alternatively, the replay network could reference the same geospatial index database and nodes could subscribe to only download RPIs from their current location's index. The countermeasures to this smart variant of a distributed replay attack include more dynamic use of higher H3 resolutions to reduce an area's geospatial validity and more precise smartphone locations.

Using a hierarchical geospatial index library like H3 also has trade-offs. Since the geometry forms strict boundaries, we must consider neighboring cells and do so in an optimized way as discussed in §5. While this addresses the near-but-cross-boundary case, it also opens up a larger area for otherwise invalid RPIs to be replayed. This is true in the likely case with 6/7 probability where a user is not in the center cell of a parent, but one of its six neighboring cells. However, it remains as shown in our evaluation, that even using a 10/16 resolution significantly decreases incorrectly matched RPIs.

Resource Requirements. As with most systems modifications that enhance accuracy, GAEN$^+$ requires more resources to operate than original GAEN without a geospatial component. Our evaluation shows negligible additional power consumption during broadcasts, which accounts for periodically accessing the smartphone's coarse location, calling into H3 for a cell index, and deriving new LRPIKs.

Noticeable battery consumption is observed during matching, plainly attributable to the additional algorithm complexity of deriving and matching RPIs for three times the number of locations for a given day. However, it is further noted this measurement is non-linear with computation time. As the test is conducted on a real-world device, this observation is potentially due to OS scheduling or other background system processes contending for resources. Still, we believe the computation time is reasonably acceptable.

Storing H3 Indices. To derive LRPIKs while matching captured RPIs, GAEN$^+$ requires a smartphone to store a 14-day, or the determined epidemiological relevant period for a particular virus, window of H3 cell index history. Indices are stored in a daily set to allow for eventual expiration when a day falls outside of the

window. In effect, each set has a 24-hour resolution. Further, the cell locations are derived using coarse location services to meet the requirements from Google/Apple that precise locations cannot be utilized. Despite common navigation apps often storing varying degrees of location history, we note this new feature may raise privacy concerns and now review the original GAEN assumptions and functionality provided to users.

The H3 cell index history is never meant to leave the device, be shown or exposed to any user through interfaces or logs, or remain in storage longer than 14 days. This aligns with the confidentiality of TEKs and derived critical keys featured in the GAEN algorithm. Ahmed et al. [59] investigated COVIDWISE, the GAEN app for the state of Virginia, U.S., and found no TEKs revealed to users in logs, or other means, and note the safeguards in place between signatures and API restrictions to safeguard access from other apps. Effectively, a malicious actor would need to severely compromise a smartphone at the operating system level, the trusted component, to successfully retrieve the index history. Unfortunately at that point, the device is compromised to the point H3 cell index history is just a small fraction of private information, real-time GPS coordinates or call history for example, that an attacker may exfiltrate.

User Exposure Location Discovery. The concern that location history on one device can be used to reveal the location of a confirmed exposure to another user also exists. While technically feasible with additional framework modifications, *this is not an implemented feature of GAEN$^+$*. Should modification occur by a tech-savvy third-party, they are equally capable of creating their own app that captures original GAEN framework RPIs and records its location with far more precise GPS coordinates. Given GAEN advertisements are broadcasted over-the-air with BLE and therefore easily sniffed, there is no safeguard to prevent others from creating an app that receives DKs from a GAEN-powered app on a jailbroken smartphone (to circumvent any OS or SDK restrictions if necessary) and matches an identity and location through data complementing with other media. This now provides basis for our privacy preserving assertion.

Privacy Preserving Assertion. Despite adding geospatial awareness, GAEN$^+$ does not diminish the privacy preserving aspects and requirements of GAEN. Introducing a geospatial component during RPIK derivation, GAEN$^+$ maintains a bi-layer of indirection because the location context is effectively hidden from both those nearby receiving RPIs and the central server that relays DKs to other users. Note that in order to derive a location from a DK's TEK, one must also have an RPI derived from it, and vice versa.

From the central server's perspective, GAEN$^+$ does not include additional data in uploads, such as location or RPIs, as this behavior is unmodified from GAEN (a user still only generates one TEK per day and optionally uploads a limited selection upon diagnosis). Since the server does not have RPIs by regular means, there is no way to determine the location a TEK was generated.

There are two recipients to consider for captured RPIs: the honest user and the malicious actor. It is important to note that, *by virtue of being able to capture an RPI from a nearby user, one can also simply record their own location which is inherently shared by the nearby user, just as easily as they can record the current time.* The honest user does not attempt to derive location and therefore is not a threat. Suppose the malicious actor has captured RPIs, receives DKs, and attempts to bruteforce the RPI derivation to determine RPI derivation location. For each TEK, this requires computing LRPIKs for each geospatial index in a chosen resolution, then comparing each of the 144 derived RPIs to the set of captured RPIs. Considering resolution 10 from H3, this requires approximately 2^{42} comparisons per random TEK. Despite this being a reasonable safeguard, as noted, bruteforcing is a superfluous activity – the malicious actor can already query their own location, thus the nearby user's.

Therefore, adding a location component to LRPIK derivation does not expose location context that a malicious actor couldn't already attain with easier, direct methods.

Retrieving User Location. While GAEN$^+$ adds a geospatial context to mitigate replay attacks, it also introduces potential for other location-based attacks and shortcomings. GAEN$^+$ uses coarse locations that come from WiFi and cell data (discussed in §5), versus precise data which can augment the coarse data with actual GPS signals. Not only does this adhere to stricter privacy and GAEN's Additional Terms, it also eliminates the risk to GPS spoofing attacks and other targeted GPS receiver attacks. However, it also relies on the accuracy of the crowd-source data provided by other cell service subscribers which may vary between services, maliciously poisoned, or otherwise tampered. IP geolocation accuracy and similarity among nearby networks is another factor for WiFi, particularly when reliable

cell service may be intermittent or unavailable, such as on a plane, subway train, or cruise ship.

The GPS spoofing attack would need to be considered if GAEN$^+$ was modified to include precise locations and therefore likely using GPS. With GPS spoofing, an attacker provides a valid GPS signal at a signal strength stronger than the valid GPS signals in the area are able to currently provide, which the GPS receiver dutifully accepts. In the context of a GAEN$^+$ variant using precise locations, an RPI may be consequently generated for a location other than where the smartphone is located.

Remaining Susceptible Attacks. While we have shown GAEN$^+$ largely addresses the geographically distributed replay attack through minor modifications to the algorithms, it is still susceptible to other known attacks which target other weaknesses. For example, if a malicious actor has the ability to know TEKs that will eventually be confirmed positive and uploaded to the public health authority server, the requirement to first crowd-source other RPIs to then replay is removed from the attack. The compromise vectors are numerous, including colluding with a health care provider and server or database compromise. Using this advantage, the malicious app can simply create location-valid RPIs from a to-be confirmed-positive TEK and broadcast to nearby users. This vulnerability exists in GAEN due to the fundamental step of anonymously uploading TEKs.

Responsible Disclosure. As our research inspired us to implement a low-cost proof-of-concept for a distributed, crowd-sourced replay attack network, we reported our findings to Apple via product-security @apple.com and Google via https://issuetracker.google.com/issues/new on 10/14/2021, which has so far prompted Google to award an Honorable Mention.

Future Work & Recommendations. While GAEN$^+$ demonstrates the ability to reduce the impact of distributed replay attacks while preserving user privacy, opportunities exist for future improvement.

- *Location Precision.* GAEN$^+$ utilizes coarse locations to adhere to GAEN's Additional Terms that forbids using precise location data. However, the two platforms offer various levels of precision that may vary substantially in practice (§5). In keeping with the approach of GAEN, it is recommended that Google and Apple collaborate to define a consistent radius coarse location service specifically for contact tracing, which allows for optimized use with H3.
- *H3 Resolutions.* Given the range of precision provided by the 16 resolutions of H3, future work can explore creative applications of dynamic resolutions that adapt to population density and events. For example, if a smartphone detects it is inside a stadium, the resolution increases to provide more precision.
- *Algorithm Optimization.* While the C++ implementations of GAEN$^+$ show negligible or acceptable tradeoff overhead, future work may focus on reducing the algorithmic complexity. One approach is to annotate received RPIs with the current H3 index to reduce the number of RPI comparisons during matching. Another is to decrease the cell index history window size, only requiring the additional location loop to execute over locations from a given interval, perhaps 2 hours versus 24 hours. However, we note these approaches add precision to time and location for both the user and others, which must be taken in account in the privacy-performance tradeoff.
- *Obscured H3 Indices.* Storing the H3 cell index history in an obscured or oblivious way can increase the privacy preserving characteristics of GAEN$^+$, further strengthening resilience against device compromise or seizure. Future research may adapt methods presented by DP-3T, specifically the cuckoo filter [12] in the unlinkable design. Though we leave this for future research, a re-imagined approach could potentially utilize a new local filter per day storing locations with an optimized way to check if a location was included.
- *Applicability.* While GAEN$^+$ is aptly named as an enhancement to GAEN, the core concept and key contribution of adding a geospatial context may be applicable to other DCT protocols that are generally susceptible to replay attacks through their agnostic key derivation and beacon broadcast mechanisms [21, 33, 38]. Effectively, GAEN$^+$ leverages a validation method in which each party can determine the "secret" by independently observing their current context such as locations for DCT protocols.

8 Related Work

Having recognized the absence of geolocation context in GAEN, *ACTGuard* [28] recommends thwarting replay attacks by installing a third-party app that creates a hash of captured RPIs with respective time, location, and its own RPI. This hash is uploaded to a third-party central server to later distribute to other users. However, the installation of an additional app for *ACT-*

Protocol	GAEN Compatible	Location Context	No Payload Mod.	Location Indirection	Location-Free Payload	Replay Attack Defense	No Third-party Server	No Third-party App	No Additional Uploads
GAEN [14, 60]	-	✗	-	✗	✓	✗	✓	✓	✓
ACTGuard [28]	✓	✓	✓	✗	✓	✓	✗	✗	✗
Hashomer [25]	✗	✓	-	✗	✓	✓	✓	✓	✓
Raskar et al [27]	✓	✓	✗	✗	✗	✓	✓	✓	✓
GAEN$^+$	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3. Comparison of GAEN$^+$ with other related works.

Guard introduces an unnecessary adoption barrier and complexity. For each captured RPI at a new time and location, the user's smartphone must compute, store, and upload a one-way hash. Additionally, to download a very large set of hashes and compare during matching against positive users' DKs can create massive amounts of hashes and network transfer burden.

Raskar et al. [27] propose recommendations for adding global context to GAEN in an effort to address a broader set of inherent limitations of its decentralized approach, including reducing false-positives and — in contrast to the GAEN principle — to allow a user to recall the environment in which an exposure occurred. They propose inserting GPS/location data into the AEM field, which is subsequently encrypted with the AEMK, as part of the advertisement payload.

While their recommendation begins to approach similar resiliency against geographically distributed replay attacks as GAEN$^+$, it requires adding location context to the payload. This direct inclusion thus risks alarming typical users who may only focus on the idea their GPS coordinates are broadcast for anyone to capture. This approach risks inhibiting adoption, whereas GAEN$^+$ maintains a location-free payload. Also, while both solutions recommend GPS, neither concretely provides a solution for representing user's geospatial location. Given the limitations of their approaches, as illustrated in Table 3, we motivated the development of GAEN$^+$, whose payload location-free protocol is completely compatible with GAEN.

Pinkas et al. introduce *Hashomer* [25], utilizing coarse location and GeoHash in their own key derivation that varies significantly from GAEN. However, this protocol is subject to limitations described in (§4.2), and further remains vulnerable to replay attacks given location context is optional.

Other proposed solutions to mitigate replay attacks on DCT protocols in general deviate even further from our stated research goals (§4). Vaudenay proposes bidirectional communication between smartphones for challenge-response [23], while Pietrzak introduces a non-interactive "delayed authentication" that uses a weak commitment scheme and a message authentication code to require a two step authentication process [24].

9 Conclusion

Digital contact tracing offers significant potential to help reduce the spread of SARS-CoV-2 and other viruses. Low-cost, crowd-sourced, geographically distributed replay attack networks threaten the GAEN framework if deployed by malicious citizen or nation-state actors, replaying GAEN advertisements throughout the world. Such a scaled attack would greatly increase false-positive notifications and introduce cascading inequitable economic and social effects, undermining the intent of GAEN as a public good. By adding geospatial awareness during an intermediate key derivation, enabled by coarse location coordinates and the H3 hierarchical geospatial index library, our GAEN$^+$ is a solution that demonstrates its ability to mitigate cross-region replay attacks while preserving user privacy. Our experiment results show that GAEN$^+$ is able to effectively prevent distributed replay attacks with acceptable additional overhead compared with the original GAEN framework, supporting its feasibility as a real-world enhancement.

Acknowledgement

We thank Moti Yung and the anonymous reviewers for their invaluable feedback. This research was partially supported by National Science Foundation (NSF) Awards 2112471 and 2118491. Any opinions, findings, and conclusions or recommendations in this paper are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] "COVID-19 map - Johns Hopkins Coronavirus Resource Center," https://coronavirus.jhu.edu/map.html, (Accessed on 09/26/2021).

[2] T. Li, C. Cobb, J. J. Yang, S. Baviskar, Y. Agarwal, B. Li, L. Bauer, and J. I. Hong, "What makes people install a covid-19 contact-tracing app? understanding the influence of app design and individual difference on contact-tracing app adoption intention," *Pervasive and Mobile Computing*, vol. 75, p. 101439, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119221000833

[3] A. Trivedi, C. Zakaria, R. Balan, and P. J. Shenoy, "Wifitrace: Network-based contact tracing for infectious diseases using passive wifi sensing," *CoRR*, vol. abs/2005.12045, 2020. [Online]. Available: https://arxiv.org/abs/2005.12045

[4] "Overview ‹ safe paths — mit media lab," https://www.media.mit.edu/projects/safepaths/overview/, (Accessed on 11/30/2021).

[5] Y. Luo, C. Zhang, Y. Zhang, C. Zuo, D. Xuan, Z. Lin, A. C. Champion, and N. Shroff, "Acoustic-turf: acoustic-based privacy-preserving covid-19 contact tracing," *arXiv preprint arXiv:2006.13362*, 2020.

[6] P.-S. Loh, "Flipping the perspective in contact tracing," *arXiv preprint arXiv:2010.03806*, 2020.

[7] W. Lueks, S. Gürses, M. Veale, E. Bugnion, M. Salathé, K. G. Paterson, and C. Troncoso, "Crowdnotifier: Decentralized privacy-preserving presence tracing," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 350–368, 2021. [Online]. Available: https://doi.org/10.2478/popets-2021-0074

[8] "BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders," https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf, (Accessed on 09/26/2021).

[9] S. Niyogi, J. Petrie, S. Leibrand, J. Gallagher, M. E. Hamish, Z. Szabo, G. Danezis, I. Miers, H. de Valence, and D. Reusche, "TCNcoalition/TCN: Specification and reference implementation of the TCN protocol for decentralized, privacy-preserving contact tracing." https://github.com/TCNCoalition/TCN, April 2020, (Accessed on 04/23/2020).

[10] "Documentation for pan-european privacy-preserving proximity tracing (PEPP-PT)," https://github.com/pepp-pt/pepp-pt-documentation, (Accessed on 10/01/2021).

[11] P. Singh, A. Singh, G. Cojocaru, P. Vepakomma, and R. Raskar, "PPContactTracing: A privacy-preserving contact tracing protocol for covid-19 pandemic," *arXiv preprint arXiv:2008.06648*, 2020.

[12] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli *et al.*, "Decentralized privacy-preserving proximity tracing," *arXiv preprint arXiv:2005.12273*, 2020.

[13] H. Wen, Q. Zhao, Z. Lin, D. Xuan, and N. Shroff, "A study of the privacy of covid-19 contact tracing apps," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2020, pp. 297–317.

[14] "Exposure notifications: Helping fight COVID-19 - google," https://www.google.com/covid19/exposurenotifications/, (Accessed on 10/02/2021).

[15] "Privacy-preserving contact tracing - Apple and Google," https://covid19.apple.com/contacttracing, (Accessed on 09/27/2021).

[16] "Mobile OS market share 2021," https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/, (Accessed on 09/27/2021).

[17] "Privacy and security attacks on digital proximity tracing systems," https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf, (Accessed on 10/01/2021).

[18] S. Farrell and D. J. Leith, "A coronavirus contact tracing app replay attack with estimated amplification factors," 2020.

[19] L. Baumgärtner, A. Dmitrienko, B. Freisleben, A. Gruler, J. Höchst, J. Kühlberg, M. Mezini, R. Mitev, M. Miettinen, A. Muhamedagic *et al.*, "Mind the gap: Security & privacy risks of contact tracing apps," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 458–467.

[20] N. Ahmed, R. A. Michelin, W. Xue, S. Ruj, R. Malaney, S. S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, and S. K. Jha, "A survey of covid-19 contact tracing apps," *IEEE access*, vol. 8, pp. 134 577–134 601, 2020.

[21] Y. Gvili, "Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc." *Cryptology ePrint Archive*, 2020.

[22] "Doj announces seizure of domain behind Russian-backed botnet," https://www.cnn.com/2018/05/24/politics/doj-botnet-takedown-malware-russia-hackers/, (Accessed on 10/12/2021).

[23] S. Vaudenay, "Analysis of dp3t-between scylla and charybdis," Tech. Rep., 2020.

[24] K. Pietrzak, "Delayed authentication: Preventing replay and relay attacks in private contact tracing," Cryptology ePrint Archive, Report 2020/418, 2020, https://ia.cr/2020/418.

[25] B. Pinkas and E. Ronen, "Hashomer–privacy-preserving bluetooth based contact tracing scheme for hamagen," in *Innovative Secure IT Technologies against COVID-19 (CoronaDef) Workshop*, 2021.

[26] P. Tedeschi, S. Bakiras, and R. Di Pietro, "SpreadMeNot: A provably secure and privacy-preserving contact tracing protocol," *arXiv preprint arXiv:2011.07306*, 2020.

[27] R. Raskar, A. Singh, S. Zimmerman, and S. Kanaparti, "Adding location and global context to the google/apple exposure notification bluetooth api," *arXiv preprint arXiv:2007.02317*, 2020.

[28] M. Casagrande, M. Conti, and E. Losiouk, "Contact tracing made un-relay-able," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 221–232.

[29] "Exposure notification - cryptography specification.pages," https://blog.google/documents/69/Exposure_Notification_-_Cryptography_Specification_v1.2.1.pdf/, (Accessed on 09/26/2021).

[30] "Exposure notification - Bluetooth specification.pages," https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf/, (Accessed on 09/26/2021).

[31] Q. Zhao, H. Wen, Z. Lin, D. Xuan, and N. Shroff, "On the accuracy of measured proximity of bluetooth-based contact tracing apps," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2020, pp. 49–60.

[32] J.-H. Hoepman, "A critique of the google apple exposure notification (GAEN) framework," *arXiv preprint arXiv:2012.05097*, 2020.

[33] R. Sun, W. Wang, M. Xue, G. Tyson, S. Camtepe, and D. C. Ranasinghe, "An empirical assessment of global covid-19 contact tracing applications," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1085–1097.

[34] "How Bluetooth technology uses adaptive frequency hopping to overcome packet interference | bluetooth® technology website," https://www.bluetooth.com/blog/how-bluetooth-technology-uses-adaptive-frequency-hopping-to-overcome-packet-interference/, (Accessed on 10/12/2021).

[35] Q. Zhao, C. Zuo, J. Blasco, and Z. Lin, "Periscope: Comprehensive vulnerability analysis of mobile app-defined bluetooth peripherals," in *ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*. ACM, 2022, pp. 521–533.

[36] "Bluetooth addresses & privacy in bluetooth low energy - novel bits," https://www.novelbits.io/bluetooth-address-privacy-ble/, (Accessed on 10/12/2021).

[37] "Corona-sniffer: Contact tracing BLE sniffer PoC," https://github.com/oseiskar/corona-sniffer, (Accessed on 09/30/2021).

[38] 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, https://ia.cr/2020/505.

[39] "The pernicious invisibility of zero-click mobile attacks," https://www.forbes.com/sites/forbestechcouncil/2020/12/14/the-pernicious-invisibility-of-zero-click-mobile-attacks/?sh=5dfe48433079, (Accessed on 10/13/2021).

[40] "A new NSO zero-click attack evades Apple's iPhone security protections, says citizen lab," https://techcrunch.com/2021/08/24/nso-pegasus-bahrain-iphone-security/, (Accessed on 10/13/2021).

[41] "Sneaky zero-click attacks are a hidden menace," https://www.wired.com/story/sneaky-zero-click-attacks-hidden-menace/, (Accessed on 10/13/2021).

[42] "Smartphone GPS accuracy study in an urban environment," https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0219890, (Accessed on 10/10/2021).

[43] "How to protect yourself & others," https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/prevention.html, (Accessed on 10/10/2021).

[44] "Exposure notifications service additional terms," https://blog.google/documents/72/Exposure_Notifications_Service_Additional_Terms.pdf/, (Accessed on 10/09/2021).

[45] "Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning," https://www.globe.gov/documents/2631933/6965868/42534991.pdf, (Accessed on 10/10/2021).

[46] "Using the significant-change location service | Apple developer documentation," https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_significant-change_location_service, (Accessed on 10/10/2021).

[47] "Exposure notifications," https://www.aphl.org/programs/preparedness/Crisis-Management/COVID-19-Response/Pages/exposure-notifications.aspx, (Accessed on 09/26/2021).

[48] "Geohashes - IBM documentation," https://www.ibm.com/docs/en/streams/4.3.0?topic=334-geohashes, (Accessed on 10/03/2021).

[49] "S2 geometry," https://s2geometry.io/, (Accessed on 10/02/2021).

[50] "H3," https://h3geo.org/, (Accessed on 10/02/2021).

[51] C. Böhm, "Space-filling curves for high-performance data mining," *arXiv preprint arXiv:2008.01684*, 2020.

[52] "Mercator projection | definition, uses, & limitations | britannica," https://www.britannica.com/science/Mercator-projection, (Accessed on 10/12/2021).

[53] "Location data maps SDK for android google developers," https://developers.google.com/maps/documentation/android-sdk/location, (Accessed on 09/28/2021).

[54] "What is the distance of one city block?" https://www.reference.com/science/distance-one-city-block-532a6843f5039b79, (Accessed on 10/10/2021).

[55] "Bindings | H3," https://h3geo.org/docs/community/bindings, (Accessed on 10/03/2021).

[56] "Average app file size: Data for Android and iOS mobile apps," https://sweetpricing.com/blog/2017/02/average-app-file-size/, (Accessed on 10/03/2021).

[57] "Table of cell areas for H3 resolutions," https://h3geo.org/docs/core-library/restable/, (Accessed on 09/26/2021).

[58] "Exposure notifications API," https://github.com/google/exposure-notifications-internals, (Accessed on 10/12/2021).

[59] S. Ahmed, Y. Xiao, T. Chung, C. Fung, M. Yung, and D. Yao, "Privacy guarantees of bluetooth low energy contact tracing: A case study on covidwise," *Computer*, vol. 55, no. 02, pp. 54–62, feb 2022.

[60] "Privacy-preserving contact tracing - Apple and Google," https://covid19.apple.com/contacttracing, (Accessed on 10/02/2021).

[61] "Drugstores struggle to keep covid at-home tests in stock amid omicron crisis," https://www.cnbc.com/2022/01/05/drugstores-struggle-to-keep-covid-at-home-tests-in-stock-as-omicron-rages-across-us.html, (Accessed on 03/14/2022).

[62] "In africa at-home covid tests are scarce and expensive, help may not come until next year | pbs newshour," https://www.pbs.org/newshour/world/in-africa-at-home-covid-tests-are-scarce-and-expensive-help-may-not-come-until-next-year, (Accessed on 03/14/2022).

[63] "Tracking the covid-19 economy's effects on food, housing, and employment hardships | center on budget and policy priorities," https://www.cbpp.org/research/poverty-and-inequality/tracking-the-covid-19-economys-effects-on-food-housing-and, (Accessed on 03/14/2022).

A Distributed Replay Attack Scenario

Assume there are three honest users Alice, Bob, and Charlie who have a GAEN-powered app installed on their smartphone, and two malicious users Mallory and Malaki who do not know each other but have individually subscribed to the replay attack network and have the replay app installed on their smartphone. The replay attack works as follows:

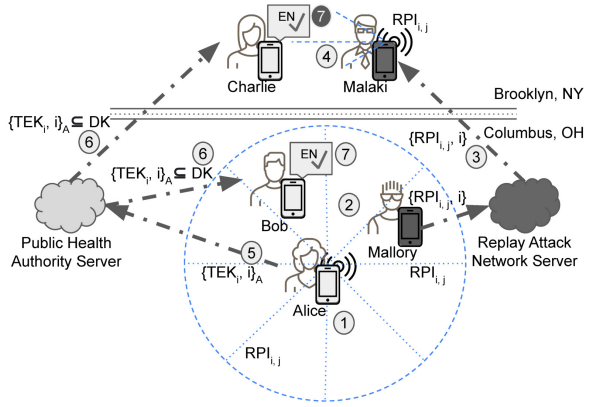


Fig. 8. Step-by-step attack scenario of GAEN's vulnerability to crowd-sourced, geographically distributed replay attacks

(1) Alice generates a TEK and RPIK daily, every 10-15 minutes broadcasts a new RPI over BLE.
(2) Bob and Mallory are within Alice's BLE advertising range, capture her advertisement, and extract the RPI. Bob stores the RPI for 14 days. Mallory calculates the current ENIntervalNumber i and sends the captured (RPI, i) pair to the replay attack network from his location in Columbus, OH.
(3) Malaki, in Brooklyn, NY, receives (RPI, i) pairs, including Alice's, and continuously replays the broadcast of received RPIs in the interval window i.
(4) Charlie receives Alice's RPI, replayed by Malaki.
(5) A few days later, Alice is positively diagnosed for COVID-19. She elects to submit her (TEK, i) pairs to the public health authority's server.
(6) Bob and Charlie receive the set of anonymous DKs from the public health authority server and derive the RPIKs and subsequent RPIs. Bob and Charlie individually find a match between Alice's RPI and an RPI derived from a TEK.
(7) Bob and Charlie are both notified of exposure (due to proximity to Alice, though this is anonymized). Bob's notification is valid whereas Charlie's is a false-positive.

B GAEN Replay Attack Impact

A crowd-sourced, geographically distributed replay attack network can cause severe consequences and disruptions to daily lives, communities, and economies. Depending on government, public health authority, employer, or academic institution guidance and policies, an exposure notification may require someone to self-quarantine and lead to wide-spread cascading effects to others. This lost-cost, feasible attack may be inviting to nearby or far away malicious actors wishing to erode public trust and further amplify these effects or simply desire notoriety. Note, we have never deployed our own concrete proof-of-concept in practice.

As shown recently with the Omicron variant, the availability of accurate, rapid, at-home tests fluctuates with positivity rates throughout the pandemic [61]. With false-positive notifications, a test may be used that otherwise could have been saved for someone with a legitimate exposure. This scarcity is even more consistently drastic in developing countries or hot spot regions [62], and further prohibitively expensive. Given viruses do not financially discriminate, daily routines become a larger risk and burden to the disadvantaged or low-income populations, who may rely more on exposure notifications. An individual may avoid contact with friends, family members, or work places, after receiving numerous notifications if it means reducing the risk of exposing those who are more vulnerable, do not have testing readily accessible, or can not afford to miss work for a smaller paycheck.

The impacts to individuals also broadly effects communities. Teachers or students may miss classes requiring a decreasing supply of substitutes to cover, resulting in continual disruption of education. Food suppliers may become excessively delayed in shipping causing grocery store stocks to dwindle. Replay attacks continuously and successfully executed in high numbers would become widely publicized, likely resulting in a loss of user confidence and a severe negative impact on GAEN adoption rates. Daily personal activities or long-planned vacations may be cancelled or postponed, exacerbating the cascading economic and mental health effects [63].