# PrivGeo: Privacy-Preserving Geofencing using Paillier Encryption

Oliver Timothy[1] and Hai-Van Dang[1]

University of Plymouth, United Kingdom

**Abstract.** Geofencing is a widely adopted technology that enables virtual boundary enforcement for location-based applications. However, conventional geofencing methods pose significant privacy risks due to continuous location tracking. This paper presents PrivGeo, a privacy-preserving geofencing system leveraging Paillier homomorphic encryption to enable secure geofence containment checks while protecting user location privacy. The proposed scheme integrates the Haversine formula with homomorphic encryption to compute distance securely without exposing users' precise locations. Compared to the existing secure distance computation based on Paillier encryption, our method reduces encryption overhead by minimising the number of encryption at the user side. Experimental evaluations demonstrate that the proposed PrivGeo reduces encryption overhead by an average 46%, and maintains low latency, making it suitable for low-powered devices and near real-time geofencing applications. The propose scheme is also proved to be correct and IND-CPA (indistinguishability against chosen-plaintext attacks) secure.

**Keywords:** Homomorphic encryption · Paillier encryption · Geofencing · Haversine.

## 1 Introduction

Geofencing is a technology that defines a virtual boundary around a physical location, allowing systems to detect when a user's device enters or exits this area. The boundary can be defined as a circle, polygon or polyline [26,30,9]. It has seen widespread adoption across multiple industries, including marketing and retail [5,23,20], fleet management and transportation [18,19], public health and patient monitoring [28,29,21], behavioural research [25], and public safety [27]. This broad application underscores its growing significance.

Despite its utility, geofencing introduces significant privacy risks due to continuous location tracking. A primary concern is identity inference, where anonymised location data can be cross-referenced with external datasets to re-identify individuals. De Montjoye et al. [8] demonstrated that 95% of individuals in a dataset of 1.5 million could be uniquely identified using just four spatio-temporal points, exposing the vulnerability of geofencing applications to de-anonymisation. Similarly, Liu et al. [15] highlight that location data can link disparate datasets,

enabling re-identification and unauthorised tracking. Another critical issue is profiling completeness, where service providers or adversaries construct detailed user profiles based on geolocation data. Wiedemann et al. [32] demonstrate how machine learning-based semantic attacks can extract behavioural patterns, even when obfuscation techniques are applied. Liu et al. [15] further emphasise that Points of Interest (POIs), such as hospitals or restaurants, can reveal sensitive behavioural insights and predict user activities. These risks raise concerns over consumer surveillance, targeted advertising, and data exploitation, underscoring the need for privacy-preserving geofencing solutions.

To address these privacy risks, there exists a variety of privacy preserving solutions for the geofencing and other location-based services. The existing methods can be classified into four categories: privacy policy–based mechanisms, obfuscation-based mechanisms, cryptography-based mechanisms, cooperation and caching-based mechanism [13]. The privacy policy-based approach depends on the privacy management rules to constrain the service provider and the third party to process and store the location securely [1]. It does not prevent the provider from knowing the users' location. The cooperation and caching-based approach utilizes the user's local cache and a neighbor's cache to minimize the communication with the untrusted server [3]. However, when the data cannot be found in the cache, the communication with the server is necessary, and this approach still discloses the user's location to the server. The obsfucation-based approach implements different techniques such as location generalization (e.g. cloaking), perturbation (e.g. differential privacy), spoofing (e.g. dummy addition), anonymization to conceal the precise users' location [22,2]. Although this approach does not require complex computation, it introduces the noise into the result and reduces the accuracy. The cryptography-based approach uses the cryptographic techniques such as space transformation, multiparty computation, and private information retrieval to protect the users' location confidentiality whilst enabling the location-based queries and responses [11,10,12,34,17]. This approach has computation overhead but generates more accurate query results than the obsfucation-based one. In this paper, our approach is cryptography-based due to the higher accuracy compared to the obsfucation-based approach.

## 2 Related Work

There exist different methods to identify if a user location is inside or outside a geofence. With a circular geofence defined by a center $C$ and a radius $r$, the user is identified as being outside the boundary, if their distance to $C$ is larger than $r$. With a polygon geofence, the boundary violation can be detected using Ray Casting or Triangle Weight Characterization (TWC) algorithms. Ray Casting projects an infinite ray from a user's location and counting the number of the intersection with the boundary edges to decide if the location is inside or outside the boundary. The second method, Triangle Weight Characterization (TWC), divides the geofence domain into a finite number of triangles, then iterates over each triangle to determine if the given location is inside any of the triangle [26].

All these methods require the geofence service provider to know the user's location. In the scope of this paper, we focus on preventing the service provider to learn anything about the user's location whilst still providing the service. Although there is a paramount of research in protecting the privacy in location-based services such as point of interest, the research in privacy-preserving geofencing is still limited. In this section, we will summarize and analyse the related works in (i) calculating the distance between two locations securely which can be used for geofencing with the circular boundary, and (ii) protecting privacy in geofencing.

Šeděnka and Gasti [24] proposed PP-UTM, PP-ECEF, and PP-HS, three protocols for privacy-preserving distance computation. These allow two parties to compute their mutual distance without revealing precise locations. Originally designed for secure distance calculations, they can be adapted for privacy-preserving geofencing. PP-UTM utilises a projection-based method within Universal Transverse Mercator (UTM) zones, enabling efficient Euclidean distance computations with runtimes under 60 ms on smartphones. However, its reliance on zone-specific coordinates limits its applicability to a single UTM zone, making it unsuitable for global distance computations. Additionally, sharing projection parameters poses a privacy risk, as mismanagement of zone boundaries could inadvertently expose sensitive location information. PP-ECEF, based on an Earth-centred Cartesian coordinate system, achieves sub-1% error for distances up to 14,000 km. However, errors increase beyond this range due to limitations in the spherical Earth approximation. PP-HS, based on the haversine formula, offers the highest accuracy (error $< 0.1\%$ globally) but has the highest runtime among the three protocols. Despite this, it remains suitable for real-time applications, with runtimes of 41.3 ms on desktops and 78 ms on smartphones. It also requires 896 bytes of bandwidth, making it feasible for resource-constrained devices. All three protocols utilise discretisation, converting real-valued inputs into integers to support the DGK cryptosystem, introducing minor approximation errors in distance computations.

NEXUS, proposed by Guldner et al. [10], is a privacy-preserving geofencing protocol that utilises Paillier homomorphic encryption. It enhances privacy by offloading encrypted computations to a geofencing service, while a separate key authority manages decryption and notifications, ensuring no single entity has full access to sensitive user data. A significant limitation of NEXUS is its restriction to rectangular geofences, defined in a 2D Cartesian plane ($\mathbb{R}^2$). Containment is computed using dot products and perpendicular projections, reducing computational overhead but sacrificing geographic accuracy over large distances. Unlike protocols using geodetic models (e.g., WGS84) to account for Earth's curvature, NEXUS prioritises efficiency over precision, making it better suited for localised applications. NEXUS achieves an average evaluation time of 323 ms with a 2048-bit Paillier key, making it practical for real-world applications. However, its restriction to rectangular geofences and reliance on planar geometry limit its adaptability for global geofencing scenarios.

Lee [14] proposed a privacy-preserving proof-of-location system using the winding number algorithm and BFV homomorphic encryption, implemented with Microsoft SEAL. The client encrypts its location, while the server processes ciphertext to verify geofence containment without exposing sensitive data. Masking techniques protect intermediate results, preventing either party from inferring private information. Notably, the protocol eliminates the need for a trusted third party, enhancing user control over location privacy. The protocol benefits from efficient parallelisation of the winding number algorithm, reducing redundant computations and improving performance. However, it assumes small geofences, approximating Earth as flat, which introduces inaccuracies near the poles where longitude lines converge. Additionally, boundary conditions for polygon edges remain undefined, potentially leading to inconsistencies in edge cases. The protocol also requires multiple client-server interactions (e.g., masking, decryption, result retrieval), which may introduce latency, impacting near real-time feasibility. Despite these limitations, it demonstrates strong privacy guarantees and computational efficiency.

Bosch [4] considered a geofence as a list of GPS coordinates. The author utilised symmetric encryption to protect the location and geofence coordinates securely, and Bloom Filter to identify if a location is inside or outside the geofence. At first, all the coordinates of the geofence is encrypted than hashed to a Bloom Filter. This leads to the pre-processing cost to convert a geofence into a Bloom Filter, but optimizes the storage cost because a geofence is represented by a Bloom Filter vector. Besides, thanks to the good performance of symmetric encryption and Bloom Filter operations, the computation cost for geofence containment check is low. However, this apporach has false positive due to the Bloom Filter.

Among the reviewed protocols, PP-HS offers the best overall performance, achieving error $< 0.1\%$, global applicability, and strong privacy preservation. Its efficient runtime and low bandwidth requirements make it practical for near real-time applications on resource-constrained devices. In contrast, PP-ECEF provides strong accuracy for moderate distances but suffers from errors beyond 14,000 km. PP-UTM, while computationally efficient, is restricted to single-zone applications due to zone boundary constraints [24]. NEXUS prioritises privacy but sacrifices geographic accuracy due to planar assumptions [10] and [4] endures the false positive the cost for preprocessing the geofences. Similarly, the Privacy-Preserving Proof-of-Location protocol ensures strong privacy but faces inefficiencies and flat-Earth approximations, limiting its suitability for global applications [14]. Due to its balance of accuracy, global applicability, privacy, and computational efficiency, PP-HS was selected as the reference protocol. To ensure a fair comparison with the proposed solution, we adapt it to operate within the same cryptosystem.

## 3 Preliminaries

**Definition 1 (Haversine formula [6]).** *The Haversine formula is widely used to compute the great-circle distance between two points on a sphere. Given two locations* $\mathsf{U} = (\mathsf{lat}_A, \mathsf{lon}_A), \mathsf{G} = (\mathsf{lat}_B, \mathsf{lon}_B, r)$*, where* $r$ *represents the radius of the geofence and* $R$ *is the Earth's radius (approximately 6,371km). The distance* $d$ *between* $\mathsf{U}$ *and* $\mathsf{G}$ *is calculated as:*

$$a = \sin^2((\mathsf{lat}_B - \mathsf{lat}_A)/2) + \cos \mathsf{lat}_A \cos \mathsf{lat}_B \sin^2((\mathsf{lon}_B - \mathsf{lon}_A)/2)$$
$$d = 2R \arcsin(\sqrt{a})$$

Based on Definition 1, we apply the trigonometric identities $\sin^2 \frac{x}{2} = \frac{1-\cos(x)}{2}$ to transform $a$ into:

$$\begin{aligned} a &= \sin^2((\mathsf{lat}_B - \mathsf{lat}_A)/2) + \cos \mathsf{lat}_A \cos \mathsf{lat}_B \sin^2((\mathsf{lon}_B - \mathsf{lon}_A)/2) \\ &= \frac{1 - \cos(\mathsf{lat}_B - \mathsf{lat}_A)}{2} + \cos \mathsf{lat}_A \cos \mathsf{lat}_B \frac{1 - \cos(\mathsf{lon}_B - \mathsf{lon}_A)}{2} \\ &= \frac{1 - \cos(\mathsf{lat}_B - \mathsf{lat}_A) + \cos \mathsf{lat}_A \cos \mathsf{lat}_B \, (1 - \cos(\mathsf{lon}_B - \mathsf{lon}_A))}{2} \end{aligned}$$

**Definition 2 (Paillier cryptosystem homomorphic properties).** *Given a Paillier cryptosystem* $(,,\mathsf{pub},\mathsf{priv})$ *where* $,,\mathsf{pub},\mathsf{priv}$ *are encryption function, decryption function, public key, private key respectively and* $k$ *be a constant. The Paillier cryptosystem supports the following homomorphic properties:*

- ***Homomorphic Addition:*** $\mathsf{Enc}_{\mathsf{pub}}(m_1 + m_2) = \mathsf{Enc}_{\mathsf{pub}}(m_1) \cdot \mathsf{Enc}_{\mathsf{pub}}(m_2)$
- ***Homomorphic Scalar Multiplication:*** $\mathsf{Enc}_{\mathsf{pub}}(m \cdot k) = \mathsf{Enc}_{\mathsf{pub}}(m)^k$

The details of Paillier cryptosystem can be found at [16].

**Theorem 1 (Paillier cryptosystem security [16]).** *Paillier cryptosystem is IND-CPA (indistinguishability against chosen-plaintext attacks) secure under the Decisional Composite Residuosity Assumption (DCRA).*

**Definition 3 (PrivGeo scheme).** *Let* $\mathsf{lat}_B, \mathsf{lon}_B, r$ *be a circular geofence with the centre* $(\mathsf{lat}_B, \mathsf{lon}_B)$ *and the radius* $r$*. Let* $\mathsf{lat}_A, \mathsf{lon}_A$ *be a given location. We define a privacy-preserving circular geofencing scheme* $\mathsf{PrivGeo}$ *as a tuple of* 4 *algorithms* $\mathsf{PrivGeo} = \{\mathsf{Init}, \mathsf{Encrypt}, \mathsf{Compute}, \mathsf{Evaluate}\}$*.*

- $\{\mathsf{pub}, \mathsf{priv}\} \leftarrow \mathsf{Init}(1^\lambda)$*: given the security parameter* $\lambda$ *as input, it outputs a pair of public key and private key*
- $c \leftarrow \mathsf{Encrypt}(\mathsf{lat}_A, \mathsf{lon}_A, \mathsf{pub})$*: given a location* $(\mathsf{lon}_A, \mathsf{lat}_A)$ *and the public key* $\mathsf{pub}$*, it outputs the ciphertexts computed from the given location.*
- $f \leftarrow \mathsf{Compute}(c, \mathsf{lat}_B, \mathsf{lon}_B)$*: given the ciphertexts output* $c$ *of* $\mathsf{Encrypt}$ *algorithm, the centre of the circular geofence* $(\mathsf{lon}_B, \mathsf{lat}_B)$*, it outputs the ciphertext of the distance between a location and the geofence centre.*
- $\{0, 1\} \leftarrow \mathsf{Evaluate}(f, r, \mathsf{priv})$*: given the ciphertext output* $f$ *of* $\mathsf{Compute}$ *algorithm, the radius of the circular geofence* $r$ *and the private key* $\mathsf{priv}$*, it outputs* 1 *if the given location is inside the geofence,* 0 *otherwise.*

**Definition 4 (IND-CPA security).** *Let $\lambda$ be the security parameter and* $\mathsf{negl}()$ *be the negligible function. For a probabilistic polynomial time (PPT) adversary $\mathcal{A}$, and a simulator* $\mathsf{Sim} = \{\mathsf{Sim}_1, \mathsf{Sim}_2\}$*, consider the following two experiments:*

$\mathsf{Real}_{\mathcal{A}}(1^\lambda)$*:*

1: $\{\mathsf{pub}, \mathsf{priv}\} \leftarrow \mathsf{Init}(1^\lambda)$
2: $S \leftarrow \{\}$
3: ***for*** *each* $i \in \{1, \ldots, \ell\}$ ***do***
4: $\quad \mathcal{A}$ *outputs* $(\mathsf{lat}, \mathsf{lon})$
5: $\quad$ *Add* $(\mathsf{lat}, \mathsf{lon})$ *to* $S$
6: $\quad c \leftarrow \mathsf{Encrypt}(\mathsf{lat}, \mathsf{lon}, \mathsf{pub})$
7: $\quad$ *Send* $c$ *to* $\mathcal{A}$
8: $\mathcal{A}$ *chooses* $\{\mathsf{lat}_0, \mathsf{lon}_0, \mathsf{lat}_1, \mathsf{lon}_1\}$ *where* $(\mathsf{lat}_0, \mathsf{lon}_0) \notin S$ *and* $(\mathsf{lat}_1, \mathsf{lon}_1) \notin S$
9: $b \leftarrow \{0, 1\}$
10: $c' \leftarrow \mathsf{Encrypt}(\mathsf{lat}_b, \mathsf{lon}_b, \mathsf{pub})$
11: *Send* $c'$ *to* $\mathcal{A}$
12: $\mathcal{A}$ *outputs a bit* $b'$

$\mathsf{Ideal}_{\mathcal{A}}(1^\lambda)$*:*

1: $\{\mathsf{pub}, \mathsf{priv}\} \leftarrow \mathsf{Sim}_1(1^\lambda)$
2: $S \leftarrow \{\}$
3: ***for*** *each* $i \in \{1, \ldots, \ell\}$ ***do***
4: $\quad \mathcal{A}$ *outputs* $(\mathsf{lat}, \mathsf{lon})$
5: $\quad$ *Add* $(\mathsf{lat}, \mathsf{lon})$ *to* $S$
6: $\quad c \leftarrow \mathsf{Sim}_2(\mathsf{pub})$
7: $\quad$ *Send* $c$ *to* $\mathcal{A}$
8: $\mathcal{A}$ *chooses* $\{\mathsf{lat}_0, \mathsf{lon}_0, \mathsf{lat}_1, \mathsf{lon}_1\}$ *where* $(\mathsf{lat}_0, \mathsf{lon}_0) \notin S$ *and* $(\mathsf{lat}_1, \mathsf{lon}_1) \notin S$
9: $c' \leftarrow \mathsf{Sim}_2(\mathsf{pub})$
10: *Sends* $c'$ *to* $\mathcal{A}$
11: $\mathcal{A}$ *outputs a bit* $b'$

*PrivGeo scheme is IND-CPA (indistinguishability against chosen-plaintext attacks) secure iff for any PPT adversary $\mathcal{A}$, there exists a simulator* $\mathsf{Sim}$ *so that*

$$|\Pr[\mathsf{Real}_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}_{\mathcal{A}}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

## 4 Methodology

### 4.1 System model, adversary model and main idea

**System model.** We consider the system with the following entities:

- Carer Device $\mathsf{C}$: Acts as the carer for the user. The carer would need to know if the user is inside the geofence or not.
- User Device $\mathsf{U}$: Represents the user. The user is willing to let the carer know if they are inside the geofence or not, but does not want the carer or the geofencing service to know their exact location.
- Geofencing Service $\mathsf{G}$: Provides the service to calculate the distance between the user and the geofence centre without knowing the actual user location.

**Adversary model.** We assume that the user device $\mathsf{U}$ is trusted to record its location correctly. The geofencing service $\mathsf{G}$ and the carer device $\mathsf{C}$ are honest-but-curious. They follow the protocol properly and do not collude with each other. However, they are curious about the user's actual location. Finally, we consider the adversary $\mathcal{A}$ to be computationally bounded.

**Main Idea.** We define a circular geofence as $(\mathsf{lat}_B, \mathsf{lon}_B, r)$, where $\mathsf{lat}_B$ and $\mathsf{lon}_B$ represent the geofence centre's latitude and longitude in radians. The geofence radius $r$ is given in meters. The system components uses PrivGeo scheme described in Defintion 3 to protect the user location as follows.

1. C calls $\mathsf{Init}(1^\lambda)$ to generate a pair of keys $\mathsf{pub}, \mathsf{priv}$. C shares the public key $\mathsf{pub}$ with U.
2. U records their location and encrypts it using $\mathsf{Encrypt}(\mathsf{lat}_A, \mathsf{lon}_A, \mathsf{pub})$. U sends the ciphertext $c$ to G
3. G calls $\mathsf{Compute}(c, \mathsf{lat}_B, \mathsf{lon}_B)$ to calculate the intermediate result $f$ and sends it to C
4. C calls $\mathsf{Evaluate}(f, r, \mathsf{priv})$ to identify if U is inside or outside the geofence.

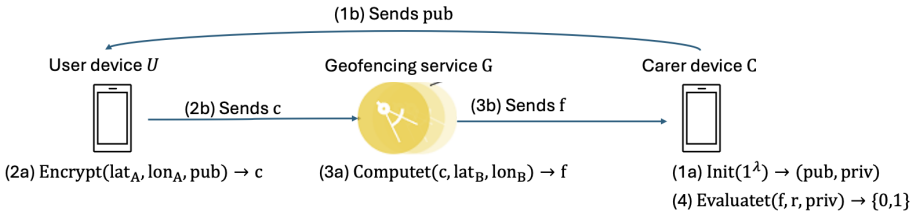



Fig. 1: System model of PrivGeo

### 4.2 PP-HS-Geo: Adapted PP-HS with Paillier encryption for geofencing application

PP-HS, originally designed for privacy-preserving distance computation by Šeděnka and Gasti [24], can be adapted for geofencing applications. As discussed in Section 2, PP-HS provides high accuracy but was implemented using the DGK cryptosystem, which differs from our Paillier-based approach.

To ensure a fair and consistent evaluation, we adapt PP-HS to operate within PrivGeo scheme to create a privacy-preserving solution called PP-HS-Geo. In its original implementation, PP-HS is a protocol to securely calculate the distance $d$ between two given locations based on the following formulation of the Haversine formula:

$$a = \sin^2((\mathsf{lat}_B - \mathsf{lat}_A)/2) + \cos \mathsf{lat}_A \cos \mathsf{lat}_B \sin^2((\mathsf{lon}_B - \mathsf{lon}_A)/2),$$
$$c = 2 \arctan\left(\sqrt{a/(1-a)}\right), d = Rc$$

In [24], $a$ can be decomposed into trigonometric components as follows:

$$a = \alpha^2\beta^2 - 2\alpha\beta\gamma\delta + \gamma^2\delta^2 + \zeta\eta\theta^2\lambda^2 - 2\zeta\eta\theta\lambda\mu\nu + \zeta\eta\mu^2\nu^2$$

where: $\alpha = \cos(\mathsf{lat}_A/2)$ $\beta = \sin(\mathsf{lat}_B/2)$ $\gamma = \sin(\mathsf{lat}_A/2)$ $\delta = \cos(\mathsf{lat}_B/2)$
$\zeta = \cos(\mathsf{lat}_A)$ $\eta = \cos(\mathsf{lat}_B)$ $\theta = \sin(\mathsf{lon}_A/2)$ $\lambda = \cos(\mathsf{lon}_B/2)$
$\mu = \cos(\mathsf{lon}_A/2)$ $\nu = \sin(\mathsf{lon}_B/2)$

Paillier encryption can be applied to encrypt $a$ as follows. This is then used to construct PP-HS-Geo in Fig. 2. Please refer to [24] for the correctness proof of the approach.

$$f = \mathsf{Enc}_{\mathsf{pub}}(a) = \mathsf{Enc}_{\mathsf{pub}}(\alpha^2)^{\beta^2} \cdot \left(-2\mathsf{Enc}_{\mathsf{pub}}(\alpha\gamma)^{\beta\delta}\right) \cdot \mathsf{Enc}_{\mathsf{pub}}(\gamma^2)^{\delta^2}$$
$$\cdot \mathsf{Enc}_{\mathsf{pub}}(\zeta\theta^2)^{\eta\lambda^2} \cdot \left(-2\mathsf{Enc}_{\mathsf{pub}}(\zeta\theta\mu)^{\eta\lambda\nu}\right) \cdot \mathsf{Enc}_{\mathsf{pub}}(\zeta\mu^2)^{\eta\nu^2}$$

$\underline{\mathsf{pp} \leftarrow \mathsf{Init}(1^\lambda)}$ ▷ Computed at carer device $C$

1: Generate Paillier pair of keys: $\{\mathsf{pub}, \mathsf{priv}\} \leftarrow \mathsf{KeyGen}\{1^\lambda\}$
2: Send $\mathsf{pub}$ key to $U$

---

$\underline{c \leftarrow \mathsf{Encrypt}(\mathsf{lat}_A, \mathsf{lon}_A, \mathsf{pub})}$ ▷ At user device $U$

1: $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\alpha^2)$
2: $c_2 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\gamma^2)$
3: $c_3 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\alpha\gamma)$
4: $c_4 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\zeta\theta^2)$
5: $c_5 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\zeta\theta\mu)$
6: $c_6 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\zeta\mu^2)$
7: **return** $c = \{c_1, c_2, c_3, c_4, c_5, c_6\}$

---

$\underline{f \leftarrow \mathsf{Compute}(c, \mathsf{lat}_B, \mathsf{lon}_B)}$ ▷ At geofencing service $G$

1: $f = c_1^{\beta^2} \cdot \left(-2c_3^{\beta\delta}\right) \cdot c_2^{\delta^2} \cdot c_4^{\eta\lambda^2} \cdot \left(-2c_5^{\eta\lambda\nu}\right) \cdot c_6^{\eta\nu^2}$
2: **return** $f$

---

$\underline{\{0,1\} \leftarrow \mathsf{Evaluate}(f, r, R, \mathsf{priv})}$ ▷ Computed at carer device $C$

1: $a \leftarrow \mathsf{Dec}_{\mathsf{priv}}(f)$
2: $c = 2\arctan\left(\sqrt{a/(1-a)}\right)$
3: $d = Rc$
4: **if** $d <= r$ **then**
5: **return** 1
6: **else**
7: **return** 0

Fig. 2: PP-HS-Geo scheme

### 4.3 Proposed scheme PrivGeo

Our proposed scheme PrivGeo utilises a transformed formulation of the Haversine formula in Definition 1, given by:

$$a = 1 - \cos(\mathsf{lat}_B - \mathsf{lat}_A) + \cos\mathsf{lat}_A \cos\mathsf{lat}_B(1 - \cos(\mathsf{lon}_B - \mathsf{lon}_A)),$$
$$d = 2R\arcsin\left(\sqrt{\frac{a}{2}}\right)$$

To avoid the complexity of the notations, we use the same notation $a$ here, but please note this is a bit different from $a$ defined in Definition 1. In the rest of the paper, this is the formula of $a$.

To avoid direct comparisons e.g. $(\mathsf{lat}_B - \mathsf{lat}_A)$ which the Paillier cryptosystem does not support, the angle difference identity $\cos(\phi - \psi) = \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\psi)$ can be used to expand the haversine formula as follows:

$$\begin{aligned}
a &= 1 - \cos(\mathsf{lat}_B - \mathsf{lat}_A) + \cos\mathsf{lat}_A \cos\mathsf{lat}_B(1 - \cos(\mathsf{lon}_B - \mathsf{lon}_A)) \\
&= 1 - (\cos\mathsf{lat}_B \cos\mathsf{lat}_A + \sin\mathsf{lat}_B \sin\mathsf{lat}_A) + \cos\mathsf{lat}_A \cos\mathsf{lat}_B(1 - (\cos\mathsf{lon}_B \cos\mathsf{lon}_A + \sin\mathsf{lon}_B \sin\mathsf{lon}_A) \\
&= 1 - \cos\mathsf{lat}_B \cos\mathsf{lat}_A - \sin\mathsf{lat}_B \sin\mathsf{lat}_A + \cos lat_A \cos\mathsf{lat}_B(1 - \cos\mathsf{lon}_B \cos\mathsf{lon}_A - \sin\mathsf{lon}_B \sin lon_A) \\
&= 1 - \cos\mathsf{lat}_B \cos\mathsf{lat}_A - \sin\mathsf{lat}_B \sin\mathsf{lat}_A + \cos lat_A \cos\mathsf{lat}_B - \cos\mathsf{lat}_A \cos\mathsf{lat}_B \cos\mathsf{lon}_B \cos\mathsf{lon}_A \\
&\quad - \cos\mathsf{lat}_A \cos\mathsf{lat}_B \sin\mathsf{lon}_B \sin\mathsf{lon}_A \\
&= 1 - \sin\mathsf{lat}_B \sin\mathsf{lat}_A - \cos\mathsf{lat}_A \cos\mathsf{lat}_B \cos\mathsf{lon}_B \cos\mathsf{lon}_A - \cos\mathsf{lat}_A \cos\mathsf{lat}_B \sin\mathsf{lon}_B \sin\mathsf{lon}_A
\end{aligned}$$

Let $a' = 1 - a = \sin\mathsf{lat}_B \sin\mathsf{lat}_A + \cos\mathsf{lat}_A \cos\mathsf{lat}_B \cos\mathsf{lon}_B \cos\mathsf{lon}_A + \cos\mathsf{lat}_A \cos\mathsf{lat}_B \sin\mathsf{lon}_B \sin\mathsf{lon}_A$
When applying the Paillier encryption with the public key $\mathsf{pub}$ to encrypt $a'$, we

have the below result thanks to the additive homomorphic encryption. This is then utilized to construct our PrivGeo in Fig. 3.

$$f = \mathsf{Enc}_{\mathsf{pub}}(a') = \mathsf{Enc}_{\mathsf{pub}}(\sin \mathsf{lat}_A)^{\sin \mathsf{lat}_B} \cdot \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \cos \mathsf{lon}_A)^{\cos \mathsf{lat}_B \cos \mathsf{lon}_B} \cdot \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \sin \mathsf{lon}_A)^{\cos \mathsf{lat}_B \sin \mathsf{lon}_B}$$

$\underline{\mathsf{pp} \leftarrow \mathsf{Init}(1^\lambda)}$ ▷ At carer device C

1: Generate Paillier pair of keys $\{\mathsf{pub}, \mathsf{priv}\} \leftarrow \mathsf{KeyGen}\{1^\lambda\}$
2: Send pub key to $U$

---

$\underline{c \leftarrow \mathsf{Encrypt}(\mathsf{lat}_A, \mathsf{lon}_A, \mathsf{pub})}$ ▷ At user device U

1: $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\sin \mathsf{lat}_A)$
2: $c_2 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \cos \mathsf{lon}_A)$
3: $c_3 \leftarrow \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \sin \mathsf{lon}_A)$
4: **return** $c = \{c_1, c_2, c_3\}$

---

$\underline{f \leftarrow \mathsf{Compute}(c, \mathsf{lat}_B, \mathsf{lon}_B)}$ ▷ At geofencing service G

1: $f = c_1^{\sin \mathsf{lat}_B} \cdot c_2^{\cos \mathsf{lat}_B \cos \mathsf{lon}_B} \cdot c_3^{\cos \mathsf{lat}_B \sin \mathsf{lon}_B}$ ▷ $\{c_1, c_2, c_3\} \leftarrow c$
2: **return** $f$

---

$\underline{\{0, 1\} \leftarrow \mathsf{Evaluate}(f, r, \mathsf{priv})}$ ▷ At carer device C

1: $a \leftarrow 1 - \mathsf{Dec}_{\mathsf{priv}}(f)$
2: $d = 2R \arcsin\left(\sqrt{\frac{a}{2}}\right)$
3: **if** $d <= r$ **then**
4: **return** 1
5: **else**
6: **return** 0

Fig. 3: Proposed scheme PrivGeo

## 5 Security and performance analysis

### 5.1 Security analysis

We will prove the correctness and IND-CPA security (indistinguishability against chosen-plaintext attacks) of the proposed scheme PrivGeo.

**Theorem 1.** *PrivGeo constitutes a privacy-preserving circular geofencing scheme with correctness and IND-CPA security given the correctness and IND-CPA security of Paillier cryptosystem [16].*

**Proof of correctness.** In Evaluate() algorithm, line 1, we have
$f = c_1^{\sin \mathsf{lat}_B} \cdot c_2^{\cos \mathsf{lat}_B \cos \mathsf{lon}_B} \cdot c_3^{\cos \mathsf{lat}_B \sin \mathsf{lon}_B}$
$= \mathsf{Enc}_{\mathsf{pub}}(\sin \mathsf{lat}_A)^{\sin \mathsf{lat}_B} \cdot \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \cos \mathsf{lon}_A)^{\cos \mathsf{lat}_B \cos \mathsf{lon}_B} \cdot \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \sin \mathsf{lon}_A)^{\cos_B \sin \mathsf{lon}_B}$
$= \mathsf{Enc}_{\mathsf{pub}}(\sin \mathsf{lat}_A \sin \mathsf{lat}_B) \cdot \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \cos \mathsf{lon}_A \cos \mathsf{lat}_B \cos \mathsf{lon}_B) \cdot \mathsf{Enc}_{\mathsf{pub}}(\cos \mathsf{lat}_A \sin \mathsf{lon}_A \cos_B \sin \mathsf{lon}_B)$
(homomorphic scalar multiplication)
$= \mathsf{Enc}_{\mathsf{pub}}(\sin \mathsf{lat}_A \sin \mathsf{lat}_B + \cos \mathsf{lat}_A \cos \mathsf{lon}_A \cos \mathsf{lat}_B \cos \mathsf{lon}_B + \cos \mathsf{lat}_A \sin \mathsf{lon}_A \cos_B \sin \mathsf{lon}_B)$
(homomorphic addition)

Therefore, $\mathsf{Dec}_{\mathsf{priv}}(f) = \sin \mathsf{lat}_A \sin \mathsf{lat}_B + \cos \mathsf{lat}_A \cos \mathsf{lon}_A \cos \mathsf{lat}_B \cos \mathsf{lon}_B + \cos \mathsf{lat}_A \sin \mathsf{lon}_A \cos_B \sin \mathsf{lon}_B = a'$ and $1 - \mathsf{Dec}_{\mathsf{priv}}(f) = a$. We deduce that $d$ in Fig. 3 is the correct distance between the user location and the geofence centre based on Haversine formula. This proves the correctness of the scheme.

**Proof of IND-CPA security.** Given $\mathsf{D}_{\text{Paillier}}$ be the ciphertext domain of Paillier cryptosystem. Let denote $c \xleftarrow{\$} \mathsf{D}_{\text{Paillier}}$ be sampling $c$ from the domain $\mathsf{D}$ randomly. We define the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ as follows

$\underline{\mathsf{Ideal}_{\mathcal{A}}(1^\lambda)}$:

1: $\mathsf{Sim}_1$ runs $\{\mathsf{pub}, \mathsf{priv}\} \leftarrow \mathsf{PrivGeo.Init}(1^\lambda)$
2: $S \leftarrow \{\}$
3: **for** each $i \in \{1, \ldots, \ell\}$ **do**
4: $\quad \mathcal{A}$ outputs $(\mathsf{lat}, \mathsf{lon})$
5: $\quad$ Add $(\mathsf{lat}, \mathsf{lon})$ to $S$
6: $\quad \mathsf{Sim}_2$ runs $c \xleftarrow{\$} \mathsf{D}_{\text{Paillier}}$
7: $\quad$ Send $c$ to $\mathcal{A}$
8: $\mathcal{A}$ chooses $\{\mathsf{lat}_0, \mathsf{lon}_0, \mathsf{lat}_1, \mathsf{lon}_1\}$ where $(\mathsf{lat}_0, \mathsf{lon}_0), (\mathsf{lat}_1, \mathsf{lon}_1) \notin S$
9: $\mathsf{Sim}_2$ runs $c' \xleftarrow{\$} \mathsf{D}_{\text{Paillier}}$
10: Sends $c'$ to $\mathcal{A}$
11: $\mathcal{A}$ outputs a bit $b'$

Because Paillier cryptosystem is IND-CPA secure [16], the outputs $c$ and $c'$ generated by $\mathsf{Sim}_2$ in $\mathsf{Ideal}_{\mathcal{A}}(1^\lambda)$ (line 6, 9) is indistinguishable from the actual ciphertexts created by $\mathsf{Encrypt}()$ in $\mathsf{Real}_{\mathcal{A}}(1^\lambda)$ (line 6,10 in definition of $\mathsf{Real}_{\mathcal{A}}(1^\lambda)$ in section 3, definition 4). It means that $\mathcal{A}$ in the real world (e.g. $\mathsf{Real}_{\mathcal{A}}(1^\lambda)$ where $\mathcal{A}$ is given the ciphertexts) can learn approximately the same amount of information as $\mathcal{A}$ in the ideal world (e.g. $\mathsf{Ideal}_{\mathcal{A}}(1^\lambda)$ where $\mathcal{A}$ is given a randomly generated value). In other words, the probability that $\mathcal{A}$ outputs the correct result is the same in both the real and ideal worlds.

$$|\Pr[\mathsf{Real}_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}_{\mathcal{A}}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

Therefore, PrivGeo is IND-CPA secure.

### 5.2 Performance analysis

We compare the computation and communication cost of PP-HS-Geo and the proposed PrivGeo based on counting the number of costly operations in their definition in Fig. 2 and Fig.3. It is seen that our PrivGeo reduces nearly 50%

Table 1: Comparing PP-HS-Geo and PrivGeo schemes

| | Encrypt() | Compute() | Evaluate() | $\lvert c \rvert$ | $\lvert f \rvert$ |
|---|---|---|---|---|---|
| PP-HS-Geo | 6 Enc | 6 exp + 5 mul | 1 Dec | 6 $\lvert G \rvert$ | $\lvert G \rvert$ |
| PrivGeo | 3 Enc | 3 exp + 2 mul | 1 Dec | 3 $\lvert G \rvert$ | $\lvert G \rvert$ |

Enc: Paillier encryption, **exp**: exponentiation, **mul**: group multiplication, **Dec**: Paillier decryption, $|G|$: size of an elemenet in the group $G$ defined in Paillier cryptosystem, $|c|$: size of the output of Encrypt, $|f|$: size of the output of Compute

the computation and communication cost of PP-HS-Geo.

# 6 Experiments and Results

## 6.1 Experimental Setup

All experiments were conducted on a host machine with an AMD Ryzen 5 3600 6-Core Processor @ 4.2 GHz, 16GB DDR4 RAM, and Windows 11 Home.

For runtime performance and scalability experiments, a distributed architecture was employed, comprising Dockerised Flask APIs for the carer device C and geofencing service G, along with a Python user script as the user device U. This models a deployment where system components communicate over a network. Meanwhile, security overhead and accuracy experiments were run as a stand-alone Python script, with the former isolating security computation from network effects and the latter enabling controlled location testing.

Flask APIs were served via Gunicorn with 4 worker processes per service. With a total of 8 worker processes running concurrently on a 6-core host machine, this corresponds to a worker-to-core ratio of 8:6 (1.33). This configuration was selected to prevent worker timeouts due to CPU contention, while maintaining efficient handling of simultaneous geofencing queries. Note that the worker configuration was only leveraged during the scalability experiment, where multiple concurrent requests were issued using multithreading.

The system was implemented in Python and employs the PyPHE library [7] for Paillier homomorphic encryption. Geofence locations were obtained from OpenStreetMap's Overpass API [33] by querying cafe locations in the UK. The data was retrieved in GeoJSON format using the WGS84 coordinate system, where each geofence is represented as a latitude/longitude coordinate pair.

The PyPHE library [7] represents real numbers using fixed-point encoding with exponent tracking, a necessary adaptation due to the Paillier cryptosystem's restriction to integer arithmetic. While this allows encrypted computations over approximate floating-point values, it introduces minor precision loss. In edge cases where the user's location $U$ is equal to the geofence centre $G$, this can lead to mathematical domain errors during decrypted evaluation, as values may fall marginally outside the valid input range of functions such as arcsin or $\sqrt{}$.

To avoid this, the user's location is rounded to five decimal places, while the geofence centre is rounded to six. Additionally, if both latitude and longitude components of the geofence centre end in trailing zeroes, a small offset of $10^{-6}$ (approximately 11.1 cm) is applied to ensure the user and geofence locations are numerically distinct. This modification slightly shifts the geofence centre but has no practical impact on geofencing accuracy.

## 6.2 Evaluation Methodology

We conduct 5 experiments: (1) accuracy, (2) runtime performance, (3) scalability, (4) security overhead and (5) communication overhead. To ensure accurate performance measurement, we exclude the following overhead:

– **Network Communication Overhead:** Time incurred during data transmission between the user device, geofencing service, and carer device (e.g., HTTP requests).

- **Data Serialisation/Deserialisation Overhead:** Processing time for converting data to and from JSON format for network transmission.
- **Key Generation Overhead:** Time spent generating cryptographic keys (e.g., Paillier public/private keys) before computation.
- **Geofence Setup Overhead:** Time spent fetching geofence coordinates before runtime measurements.

In each experiment 1-5, we run 30 times and calculate the mean and standard deviation using the SciPy library [31].

**Experiment 1 (accuracy).** This experiment evaluates the correctness of the systems in classifying user locations as inside or outside a geofence. Locations are generated using random angles and distances from the geofence center including 30 inside, outside, and edge points, plus one at the center. Each result is compared to the plaintext ground truth to determine accuracy, which is computed as: Accuracy = (Correct Predictions/Total Predictions) $\times$ 100%.

**Experiment 2 (runtime performance).** This experiment evaluates the computational performance of the system by measuring runtime for Encrypt(), Compute(), Evaluate() algorithms and the total_runtime which is the sum of the mentioned algorithms. The experiment is conducted using test cases with varying numbers of geofences: $1, 10, 100, 200, 300$.

**Experiment 3 (scalability).** This experiment evaluates the system's ability to handle a high volume of geofencing queries. The experiment simulates multiple user devices sending geofencing queries to the geofencing service. Query volume varies across $1, 10, 50$ and 100 simultaneous requests, with each test case consisting of 10 geofences.

To focus on the system's query processing performance, user device data is precomputed and encrypted before execution, ensuring that encryption overhead does not influence the results. Multithreading is used to simulate multiple user devices, where each thread issues a single geofencing query to the geofencing service. After processing all queries, system performance is evaluated using the following metrics:

- **Throughput:** The number of queries processed per second, computed as: Throughput = Number of Queries/total_runtime
- **Latency:** The average processing time per query, computed as: Latency = total_runtime/Number of Queries

**Experiment 4 (security overhead).** This experiment evaluates the runtime overhead of PP-HS-Geo and PrivGeo against a baseline geofencing protocol that uses the Haversine formula to calculate the distance without encryption. The security overhead is computed as: Overhead = ((total_runtime − Baseline Runtime)/Baseline Runtime) $\times$ 100

**Experiment 5 (communication overhead).** This experiment quantifies communication overhead by measuring the size of incoming request payloads received by the geofencing service and carer device. Only data received via network requests (e.g., encrypted coordinates or computation results) is included; response sizes and internal data retrieval are excluded.

Table 2: Runtime performance for one query with different number of geofences

| # of geofences | Run time (seconds) | PP-HS-Geo | Proposed PrivGeo |
|---|---|---|---|
| 1 | Encrypt | 1.528 ± 0.027 | 0.778 ± 0.026 |
| | Compute | 0.051 ± 0.006 | 0.049 ± 0.001 |
| | Evaluate | 0.068 ± 0.001 | 0.069 ± 0.002 |
| | total_runtime | 1.647 ± 0.026 | 0.897 ± 0.026 |
| 10 | Encrypt | 1.533 ± 0.034 | 0.775 ± 0.03 |
| | Compute | 0.47 ± 0.026 | 0.485 ± 0.027 |
| | Evaluate | 0.694 ± 0.026 | 0.697 ± 0.033 |
| | total_runtime | 2.698 ± 0.052 | 1.957 ± 0.048 |
| 100 | Encrypt | 1.56 ± 0.035 | 0.755 ± 0.015 |
| | Compute | 4.755 ± 0.056 | 4.979 ± 0.084 |
| | Evaluate | 6.948 ± 0.078 | 6.926 ± 0.063 |
| | total_runtime | 13.263 ± 0.117 | 12.66 ± 0.102 |
| 200 | Encrypt | 1.566 ± 0.036 | 0.753 ± 0.023 |
| | Compute | 9.444 ± 0.069 | 9.837 ± 0.051 |
| | Evaluate | 13.783 ± 0.082 | 13.815 ± 0.067 |
| | total_runtime | 24.793 ± 0.117 | 24.405 ± 0.086 |
| 300 | Encrypt | 1.554 ± 0.043 | 0.757 ± 0.017 |
| | Compute | 14.117 ± 0.087 | 14.723 ± 0.104 |
| | Evaluate | 20.627 ± 0.074 | 20.636 ± 0.074 |
| | total_runtime | 36.299 ± 0.13 | 36.116 ± 0.112 |

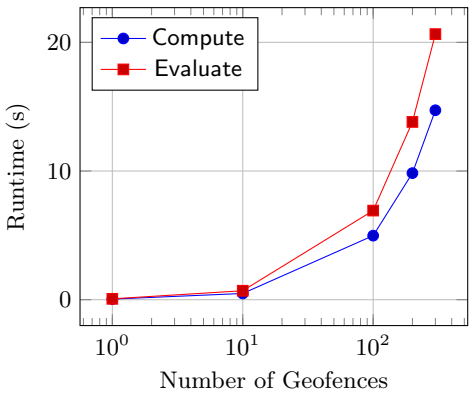
### 6.3 Results

**Experiment 1 (accuracy).** Accuracy results show that PrivGeo achieved 100% accuracy compared to PP-HS-Geo at 99.963% ± 0.201, likely due to fewer cryptographic operations using the PyPHE library [7].

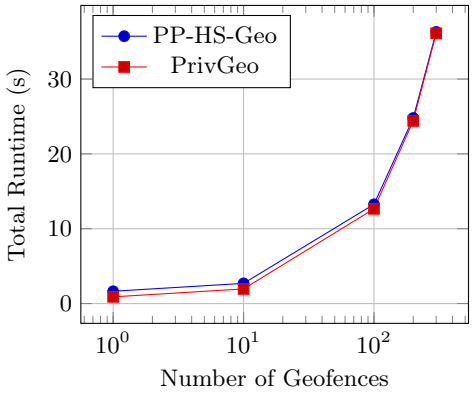
**Experiment 2 (runtime performance).** Table 2 and Figure 4a illustrate the breakdown of runtime components for the PrivGeo, while Figure 4b compares the total_runtime between the PP-HS-Geo and the proposed PrivGeo.

As shown in Table 2, Encrypt time remains effectively constant across all test cases, as each test involves only a single geofencing query. Compute and Evaluate time, however, increase as the number of geofences grows. Figure 4a visualises this trend for the proposed PrivGeo, confirming that both Compute and Evaluate scale with geofence count. Notably, Evaluate emerges as the primary bottleneck when scaling to larger numbers of geofences.

Figure 4b presents a total_runtime comparison between the PP-HS-Geo and proposed PrivGeo. The proposed PrivGeo exhibits a slightly lower runtime across the test cases, averaging 0.533 seconds faster due to encryption optimisations. However, this difference appears to diminish as the number of geofences increases, which explains similar total_runtime when the number of geofences is large.




(a) Compute and Evaluate runtime of PrivGeo




(b) Comparing total_runtime between PP-HS-Geo and PrivGeo

Fig. 4: Experiment 2 - Running time

**Experiment 3 (scalability).** Table 3 presents the total_runtime, throughput, and latency of the system for varying numbers of queries, while Figure 5 visualises the impact of increasing query volume on runtime for the proposed PrivGeo. As expected, system total_runtime increases with query volume, demonstrating a near-linear trend. The throughput of the proposed PrivGeo averages 0.745 queries per second across all test sizes, while the PP-HS-Geo achieves an average of 0.748. Notably, throughput improves with higher query volumes, reaching 0.96 q/s at 100 queries. Latency per query also decreases from over 3.6 seconds with a single query to just over 1 second at 100 queries, likely due to reduced per-query overhead when executing larger sets of queries in a single test run. These results demonstrate the system's suitability for near real-time geofencing applications.

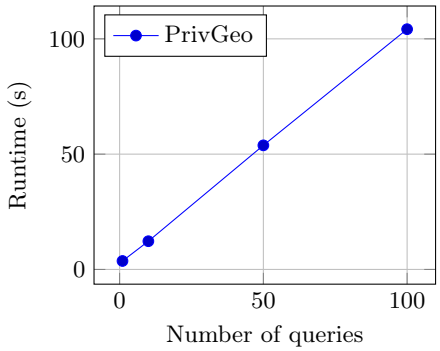



Fig. 5: Runtime vs. Queries

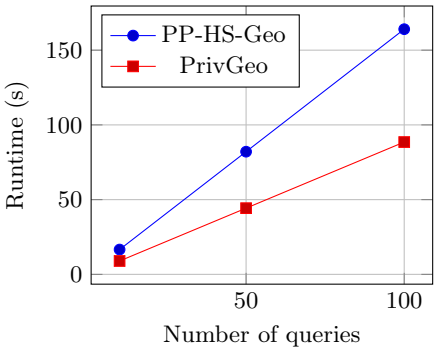



Fig. 6: Runtime vs. Queries

**Experiment 4 (security overhead).** Table 4 and Figure 6 present the overhead in total_runtime of the PP-HS-Geo and proposed PrivGeo to provide the privacy protection. The proposed PrivGeo demonstrates a significant reduction in overhead cost compared to PP-HS-Geo. Across all test cases, it achieves an average runtime reduction of approximately 46%, confirming the efficiency of

Table 3: Running time, throughput and latency for different number of queries

| # of queries | Metric | PP-HS-Geo | Proposed PrivGeo |
|---|---|---|---|
| 1 | total_runtime (s) | 3.658 ± 0.073 | 3.662 ± 0.061 |
| | Throughput (q/s) | 0.273 ± 0.005 | 0.273 ± 0.005 |
| | Latency (s/q) | 3.658 ± 0.073 | 3.662 ± 0.061 |
| 10 | total_runtime (s) | 12.205 ± 0.058 | 12.237 ± 0.082 |
| | Throughput (q/s) | 0.819 ± 0.004 | 0.817 ± 0.005 |
| | Latency (s/q) | 1.221 ± 0.006 | 1.224 ± 0.008 |
| 50 | total_runtime (s) | 53.51 ± 0.16 | 53.813 ± 0.11 |
| | Throughput (q/s) | 0.934 ± 0.003 | 0.929 ± 0.002 |
| | Latency (s/q) | 1.07 ± 0.003 | 1.076 ± 0.002 |
| 100 | total_runtime (s) | 103.647 ± 0.735 | 104.187 ± 0.329 |
| | Throughput (q/s) | 0.965 ± 0.007 | 0.96 ± 0.003 |
| | Latency (s/q) | 1.036 ± 0.007 | 1.042 ± 0.003 |

Table 4: Runtime overhead due to privacy protection

| # of queries | Baseline (s) | Metric | PP-HS-Geo | PrivGeo |
|---|---|---|---|---|
| 10 | 2.042e−5 | total_runtime (s) | 16.597 ± 0.184 | 8.947 ± 0.115 |
| | | Overhead (%) | 8.130e7 ± 8.994e5 | 4.383e7 ± 5.615e5 |
| 50 | 9.620e−5 | total_runtime (s) | 82.071 ± 0.115 | 44.288 ± 0.12 |
| | | Overhead (%) | 8.531e7 ± 1.195e5 | 4.604e7 ± 1.250e5 |
| 100 | 1.983e−4 | total_runtime (s) | 164.03 ± 0.165 | 88.521 ± 0.232 |
| | | Overhead (%) | 8.273e7 ± 8.341e4 | 4.465e7 ± 1.171e5 |

its optimisations. Despite this improvement, encryption remains the dominant factor affecting overall system performance. These results suggest that the proposed PrivGeo is better suited to low-powered devices and resource-constrained applications, as it reduces computational overhead and minimizes processing demands on the user device.

**Experiment 5 (communication overhead).** Table 5 presents the communication overhead for both the PP-HS-Geo and proposed PrivGeo, measured by the size of incoming payloads received by the geofencing service G and the carer device C. In our implementation, the payload G receives includes $c$ from Encrypt() plus the public key while C receives only the intermediate result $f$ from Compute(). As expected, C receives the same amount of data in both algorithms, since it only receives the final encrypted result, and this value increases with the number of geofences. However, the proposed PrivGeo significantly reduces the data received by the geofencing service G.

For a single geofencing query, the PP-HS-Geo sends approximately 12.14 KB to G , whereas the proposed PrivGeo requires only 6.47 KB — representing a 46.7% reduction. These results suggest that the proposed PrivGeo is more suitable for low-bandwidth or mobile network environments, as it reduces upstream

data transmission and may help lower energy consumption on constrained devices.

Table 5: Communication

| Geofences | Metric | PP-HS-Geo | Proposed PrivGeo |
|---|---|---|---|
| 1 | Geofence service G received (KB) | 12.142 | 6.47 |
| | Carer device C received (KB) | 2.784 | 2.784 |
| 10 | Geofence service G received (KB) | 12.143 | 6.47 |
| | Carer device C received (KB) | 19.347 | 19.347 |

## 7 Conclusion

This paper introduced PrivGeo, a privacy-preserving geofencing system that integrates Paillier homomorphic encryption with the Haversine formula to securely determine geofence containment without exposing users' precise location details. Conventional geofencing methods pose significant privacy threats, such as identity inference and behavioural profiling, which PrivGeo mitigates by employing encrypted location data and separation of trust. Unlike anonymisation, which can still reveal movement patterns, encryption renders location data completely unreadable, preventing identity inference. Besides, the separation of trust is enabled by minimizing the information the geofencing service $G$ and carer device $C$ hold. As a result, neither $G$ nor $C$ can reconstruct the user's movement or location history on their own.

Experimental evaluations demonstrated that the proposed algorithm reduces encryption overhead by 46% compared to existing methods, making it well-suited for low-powered devices and resource-constrained applications, such as IoT-based geofencing and mobile location tracking. Performance analysis further confirmed linear computational scaling, ensuring scalability for larger deployments. Furthermore, low encryption overhead and efficient processing contribute to low latency, supporting near real-time geofencing applications.

Despite its advantages, the system currently supports only circular geofences. Future research will focus on extending support for arbitrary geofence shapes, including polygons, evaluating performance on low-powered devices to ensure feasibility in resource-constrained environments, and exploring efficient encryption schemes to minimise overhead.

## References

1. Geographic location/privacy (georriv) working group. https://datatracker.ietf.org/doc/charter-ietf-geopriv/, accessed: 2024-04-10
2. Al-Balasmeh, H., Singh, M., Singh, R.: Framework of geofence service using dummy location privacy preservation in vehicular cloud network. Int. Arab J. Inf. Technol. **20**(1), 66–77 (2023)

3. Amini, S., Lindqvist, J., Hong, J.I., Mou, M., Raheja, R., Lin, J., Sadeh, N., Tochb, E.: Caché: caching location-enhanced content to improve user privacy. SIGMOBILE Mob. Comput. Commun. Rev. **14**(3), 19–21 (Dec 2011). https://doi.org/10.1145/1923641.1923649, https://doi.org/10.1145/1923641.1923649
4. Bösch, C.: An efficient privacy-preserving outsourced geofencing service using bloom filter. In: 2018 IEEE Vehicular Networking Conference (VNC). pp. 1–8. IEEE (2018)
5. Cheruiyot, K., Moenyane, K.: Exploring the potential of adopting geofence mobile technology in the south african retail sector. Cogent Business & Management **11**(1), 2327126 (2024)
6. Chopde, N.R., Nichat, M.: Landmark based shortest path detection by using a* and haversine formula. International Journal of Innovative Research in Computer and Communication Engineering **1**(2), 298–302 (2013)
7. Data61, C.: Python paillier library. https://github.com/data61/python-paillier (2013)
8. De Montjoye, Y.A., Hidalgo, C.A., Verleysen, M., Blondel, V.D.: Unique in the crowd: The privacy bounds of human mobility. Scientific reports **3**(1), 1–5 (2013)
9. Deshmukh, P., Bhajibhakre, A., Gambhire, S., Channe, A., Deshpande, N.: Survey of geofencing algorithms. International Journal of Computer science engineering Techniques **3**(2), 1–5 (2018)
10. Guldner, M., Spieldenner, T., Schubotz, R.: Nexus: Using geo-fencing services without revealing your location. In: 2018 Global Internet of Things Summit (GIoTS). pp. 1–6. IEEE (2018)
11. Hallgren, P., Ochoa, M., Sabelfeld, A.: Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In: 2015 13th Annual Conference on Privacy, Security and Trust (PST). pp. 1–6. IEEE (2015)
12. Hallgren, P., Orlandi, C., Sabelfeld, A.: Privatepool: Privacy-preserving ridesharing. In: 2017 IEEE 30th Computer Security Foundations Symposium (CSF). pp. 276–291. IEEE (2017)
13. Jiang, H., Li, J., Zhao, P., Zeng, F., Xiao, Z., Iyengar, A.: Location privacy-preserving mechanisms in location-based services: A comprehensive survey. ACM Computing Surveys (CSUR) **54**(1), 1–36 (2021)
14. Lee, C.: Privacy-preserving proof-of-location using homomorphic encryption (2020)
15. Liu, B., Zhou, W., Zhu, T., Gao, L., Xiang, Y.: Location privacy and its applications: A systematic study. IEEE access **6**, 17606–17624 (2018)
16. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)
17. Paulet, R., Kaosar, M.G., Yi, X., Bertino, E.: Privacy-preserving and content-protecting location based queries. IEEE transactions on knowledge and data engineering **26**(5), 1200–1210 (2013)
18. Raikar, M.M., Angadi, K., Reddy, S., Naik, N., Doddwada, A.: Fleet tracking and geofencing using the internet of things (iot). In: 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC). pp. 463–468 (2023). https://doi.org/10.1109/ICSCCC58608.2023.10176660
19. Reclus, F., Drouard, K.: Geofencing for fleet & freight management. In: 2009 9th International Conference on Intelligent Transport Systems Telecommunications, (ITST). pp. 353–356 (2009). https://doi.org/10.1109/ITST.2009.5399328
20. Rodriguez Garzon, S., Deva, B.: Geofencing 2.0: taking location-based notifications to the next level. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 921–932 (2014)

21. Salehi, W., Gupta, G., Bhatia, S., Koundal, D., Mashat, A., Belay, A.: Iot-based wearable devices for patients suffering from alzheimer disease. Contrast Media & Molecular Imaging **2022**(1), 3224939 (2022)
22. Saravanan, P.S., Ramani, S., Reddy, V.R., Farhaoui, Y.: A novel approach of privacy protection of mobile users while using location-based services applications. Ad Hoc Networks **149**, 103253 (2023)
23. Sasaki, I., Arikawa, M., Lu, M., Utsumi, T., Sato, R.: Data-driven geofencing design for point-of-interest notifiers utilizing genetic algorithm. ISPRS International Journal of Geo-Information **13**(6), 174 (2024)
24. Šeděnka, J., Gasti, P.: Privacy-preserving distance computation and proximity testing on earth, done right. In: Proceedings of the 9th ACM symposium on Information, computer and communications security. pp. 99–110 (2014)
25. Shevchenko, Y., Reips, U.D.: Geofencing in location-based behavioral research: Methodology, challenges, and implementation. Behavior Research Methods **56**(7), 6411–6439 (2024)
26. Stevens, M.N., Rastgoftar, H., Atkins, E.M.: Specification and evaluation of geofence boundary violation detection algorithms. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 1588–1596. IEEE (2017)
27. Suyama, A., Inoue, U.: Using geofencing for a disaster information system. In: 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS). pp. 1–5 (2016). https://doi.org/10.1109/ICIS.2016.7550849
28. Tobin, K., Heidari, O., Volpi, C., Sodder, S., Duncan, D.: Use of geofencing interventions in population health research: a scoping review. BMJ open **13**(8), e069374 (2023)
29. Ullah, F., Haq, H.U., Khan, J., Safeer, A.A., Asif, U., Lee, S.: Wearable iots and geo-fencing based framework for covid-19 remote patient health monitoring and quarantine management to control the pandemic. Electronics **10**(16), 2035 (2021)
30. Vagal, V., Markantonakis, K., Shepherd, C.: A new approach to complex dynamic geofencing for unmanned aerial vehicles. In: 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC). pp. 1–7. IEEE (2021)
31. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
32. Wiedemann, N., Janowicz, K., Raubal, M., Kounadi, O.: Where you go is who you are: a study on machine learning based semantic privacy attacks. Journal of Big Data **11**(1), 39 (2024)
33. Wiki, O.: Overpass api — openstreetmap wiki, (2025), https://wiki.openstreetmap.org/w/index.php?title=Overpass_API&oldid=2809610, [Online; accessed 16-February-2025]
34. Yan, Q., Lou, J., Vuran, M.C., Irmak, S.: Scalable privacy-preserving geo-distance evaluation for precision agriculture iot systems. ACM Transactions on Sensor Networks (TOSN) **17**(4), 1–30 (2021)