# Privacy-Preserving Distance Computation and Proximity Testing on Earth, Done Right

Jaroslav Šeděnka*
Dept. of Mathematics and Statistics
Masaryk University
Kotlářská 2, 61137 Brno, Czech Republic
sedenka@mail.muni.cz

Paolo Gasti
School of Engineering and Computing Sciences
New York Institute of Technology
1855 Broadway, New York, NY 10023, USA
pgasti@nyit.edu

## ABSTRACT

In recent years, the availability of GPS-enabled smartphones have made location-based services extremely popular. A multitude of applications rely on location information to provide a wide range of services. Location information is, however, extremely sensitive and can be easily abused. In this paper, we introduce the first protocols for secure *computation of distance* and for *proximity testing* over a sphere. Our *secure distance* protocols allow two parties, Alice and Bob, to determine their mutual distance without disclosing any additional information about their location. Through our *secure proximity testing* protocols, Alice only learns if Bob is in close proximity, i.e., within some arbitrary distance.

Our techniques rely on three different representations of Earth, which provide different trade-offs between accuracy and performance. We show, via experiments on a prototype implementation, that our protocols are practical on resource-constrained smartphone devices. Our distance computation protocols runs, in fact, in 54 to 78 ms on a commodity Android smartphone. Similarly, our proximity tests require between 1.2 s and 2.8 s on the same platform. The imprecision introduced by our protocols is very small, i.e., between 0.1% and 3% on average, depending on the distance.

## Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: Public Policy Issues—*Privacy*; E.3 [**DATA ENCRYPTION**]

## 1. INTRODUCTION

Since the introduction of the first popular GPS-enabled smartphones, location-aware services have been growing in popularity. Users have now access to a wide array of location-based applications, including social networks [21], directories of restaurants and hotels [49, 44], taxi pick-up schedulers [5], on-line publishing of geo-tagged photos and videos [11], local deals [17], lost/stolen phone locators [20], and "friend finders" [16, 1, 4, 34].

These services often raise severe privacy concerns. By disclosing location data to third parties (e.g., application developers, cellphone manufacturers, friends), user can be easily tracked, usually against their will. Although most users might be comfortable with (or unaware of) being tracked by cellphone carriers, they may be more reluctant to share location information with third parties of unknown reputation. The dangers of incautious location sharing have been shown to be very real with documented cases of location data being actively used by stalkers [47]. Privacy is not only an issue for the party disclosing its location: service providers may, in fact, be unwilling to learn the whereabouts of their users, for fear of bad press and lawsuits [46].

As of today, revealing location information is, in practice, an all-or-nothing proposition. Mobile devices usually implement coarse-grained "privacy settings", that allow users to turn off location services for specific applications – often preventing location-based applications from working at all.

In recent years, researchers have investigated a suitable middle ground between *full disclosure* and *no disclosure* of location information. This effort has resulted in the introduction of various techniques such as location cloaking [30], noisy distance computation [18], and privacy-preserving distance [36] and proximity protocols [50]. However, existing approaches either provide privacy at the expense of accuracy, or rely on assumptions that restricts their usefulness. In particular, cloaking and noisy techniques introduce artificial "measurement errors" in the original location, perturbing the result of the distance computation. Existing protocols for computing distances in the encrypted domain, on the other hand, offer accurate results only over small distances, since they work on either coarse approximations of Earth, or on small projections of Earth on Euclidean planes. When multiple small projections are used, parties must privately determine if their respective coordinates fall within the same projection. If they do not, and the protocols do not account for it, distance computation will return meaningless results. Finally, existing work does not address the problem of defining a suitable projection for distance computation. This makes comparison of accuracy between existing techniques difficult, since the choice of the projection affects the performance of the protocols.

**Contributions.** In this paper we introduce novel privacy-preserving protocols for reliably computing distances and

*Part of this work was done while visiting the New York Institute of Technology

*ASIA CCS'14,* June 4–6, 2014, Kyoto, Japan.
Copyright 2014 ACM 978-1-4503-2800-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2590296.2590307.

proximity between two arbitrary points on Earth. With our protocols, the two parties – henceforth, "Alice" and "Bob" – are not required to disclose their location. However, at the end of our distance protocols, Alice learns how far she is located from Bob, while Bob learns nothing.

Using our *secure proximity testing* protocols, Alice only learns whether Bob is within a certain distance. Moreover, Bob can return an *unconditional positive* (i.e., the distance between Alice and Bob always appear to be smaller than the threshold) or *unconditional negative* (the distance between Alice and Bob is always larger than the threshold) response to proximity queries from Alice, in order to provide him with an extra layer of privacy.

An important difference between two of our protocols and the current state of the art is the ability of our protocols to accurately compute distance between two parties located in arbitrary locations on earth. To achieve this, the two protocols do not make use of projections of small parts of Earth's surface.

**Our Protocols.** The first protocol, **PP-UTM**, is a concrete instantiation of Euclidean distance over a projection of Earth. The projection we use, called UTM, maps Earth over set of planes. As discussed in Section 3, this technique provides accurate results if the two parties are located within the same UTM zone. For this reason, we consider this protocol as a *baseline* for assessing the performance of our contribution. To the best of our knowledge, this is the first instantiation of an Euclidean-distance-based privacy-preserving protocol that considers a real projection. This, in turn, allows us to evaluate, for the first time, the real-world error introduced by a privacy-preserving distance computation protocol and secure proximity testing protocol.

The second protocol, **PP-ECEF**, allows Alice to calculate distances in the Earth-Centered, Earth-Fixed (ECEF, also known as Earth Centered Rotational, or ECR) coordinate system. This protocol provides very accurate results (less than 0.1% error) when the two parties are within 14,000 km (roughly 8,700 miles), and reasonably accurate results (less than 1% error) for greater distances.

Our third protocol, **PP-HS**, is based on the haversine formula [42], which is a trigonometric formula used to compute distances on a sphere. Although slightly less efficient than the previous two protocols, PP-HS is very accurate regardless of the position of Alice and Bob. The protocol introduces very small error (below 0.1%) when the two parties are more than 14,000 km apart.

**Organization.** The rest of the paper is organized as follows. In Section 2 we review the related literature. Section 3 discusses several techniques for computing distances on various approximations of Earth. In Section 4 we introduce our security model and the cryptographic tools used in our protocols. Section 5 presents our protocols for privacy-preserving distance computation. We provide a security analysis of the protocols in Section 6, and evaluate accuracy and performance in Section 7. We conclude in Section 8.

## 2. RELATED WORK

Related work on location privacy can be divided in three classes: (1) work that aims at quantifying location privacy (or lack thereof); (2) techniques that anonymize user location information, and possibly add location obfuscation; and (3) privacy-preserving protocols based on secure multiparty computation (SMC).

**Quantifying Location Privacy.** In [40] Shokri et al. develop a formal framework for the analysis of location-privacy protection mechanisms. This framework captures adversary's prior information, and models various attacks. They introduce metrics for attacker's performance, such as *accuracy*, *certainty* and *success*.

Shokri et al. [41] classify location exposure into *continuous* (i.e., the adversary can track users over time and space) and *sporadic* (the adversary's focus is on localizing users data specific point in time). Their work addresses the case of sporadic location exposure, formalizing location privacy. In their framework, the authors also perform localization attacks using Bayesian inference for Hidden Markov Processes on anonymized traces.

**Location Anonymization and Obfuscation.** Gruteser et al. [18] design a middleware architecture, used by a location broker service to anonymize user locations and still allow them to receive the intended service. This technique provide a median resolution of 125 meters, and does not protect users' privacy against the location broker. Several papers have followed the work of Grueteser et al. (see, e.g., [2, 24, 32, 39]).

Duckham et at. [9] introduced a formal model for location obfuscation techniques. The authors argue that location inaccuracy (lack of correspondence between information and reality), imprecision (lack of specificity of information), and vagueness (existence of boundary cases in information) provide an feasible way for implementing location privacy. In particular, they argue that their model provides a generic mechanism for balancing quality of information with privacy.

Krumm's work [30] shows that spatial cloaking, Gaussian noise and reduced location resolution can degrade the identification success of the adversary. Gruteser et al. [18] use $k$-anonymity, implemented via spatial and temporal cloaking, to increase the adversary's uncertainty.

In general, techniques belonging to this class provide only limited accuracy. Moreover, as has been shown by Golle et at. [15], Beresford et al. [2], Hoh et al. [19] and many others, the highly identifying nature of location information often makes proper anonymization difficult, if not impossible. This makes location anonymization not suitable for a large array of location-based services, including social networks.

**Privacy-Preserving Protocols.** More related to our work, there has been a large amount of research on SMC protocols for privacy-preserving proximity testing and privacy-preserving distance computation between two points or a point and a curve.

Zhong et al. [50] introduced Louis, Lester and Pierre, three privacy-preserving protocols for proximity testing. The Louis protocol requires a semi-trusted third party that does not learn any location information. Lester, the second protocol, does not require a third party, but one of the participants might learn the location of the other even if they are no longer close. Finally, the Pierre protocol provides better security, at the cost of reduced accuracy.

Narayanan et al. [36] show how to reduce proximity testing to equality testing. Their approach is based on dividing the Euclidean space into a grid system; the position of each user is defined as a set of adjacent triangles on the grid. If two users are within a certain range, they must share at least one grid component. Proximity is therefore computed through a simple and efficient privacy-preserving protocol. The authors extend the protocol to work on Earth by slicing it into thin one-degree strips. This instantiation of their technique introduces three classes of error: (1) even though the strips are relatively thin, the curvature of Earth still leads to errors for points at different latitudes; (2) grids from different strips do not align at the strip boundary, and therefore the parties must belong to the same strip for best accuracy; and (3) since the basic unit for the tessellation is a triangle, the perimeter of an area which represent the user location is not a circle, and therefore there may be false positives or false negatives across the boundaries of a user's zone. Classes (1) and (2) limit the applicability of this approach to relatively small distances.

Mascetti et al. [33] present *Longitude*, a privacy-aware centralized technique for determining the distance of two points. The proposed protocol implements Euclidean distance, and the result is obliviously compared with a threshold. This way, one of the parties only learns whether the other protocol participant's input is within a certain distance.

Li et al. [26] design a suite of privacy-preserving protocols that allows two parties to share information about their location with fine-grained access control. Users can specify a condition and match all users that satisfy such condition. Alternatively, their protocols rely on attribute-based encryption and homomorphic encryption.

To the best of our knowledge, [36] is the only work that addresses distances in a non-Euclidean space. However, the authors do not discuss how mapping a grid on a curved surfaces affects the accuracy of the protocol.

## 3. DISTANCE COMPUTATION BETWEEN TWO POINTS ON EARTH

Although Earth is usually considered a sphere, its shape is closer to an *oblate spheroid*, or *ellipsoid* [43]. The exact shape, called *geoid*, is defined as the global mean sea level. However, for simplicity and efficiency, geodetic computations are usually performed over an approximation of Earth's shape – a sphere or one of the various standard ellipsoids. In this paper, we consider the spherical approximation with radius 6,371 km (which is the radius for a sphere with the same surface area as the Earth's ellipsoid) and the World Geodetic System 84 (WGS84).

The WGS84 geoid is used in the Global Positioning System (GPS), and is thus a major source of coordinates available today. As this paper focuses mainly on location privacy, and not on geodesy, we refer the reader to [43] for proper treatment of this topic and introduce only the required notion below.

The position of any point on Earth can be described as *geodesic data* based on different reference system. A datum in WGS84 consists of two angles, expressed in degrees, denoted *latitude* and *longitude*. Latitude describes the north-south position of a point as the angle between the equatorial plane (i.e., the plane that intersects Earth through the equator) and the line between the point and Earth's center. The value of latitude is between $-90^\circ$ (South pole) and $90^\circ$ (North pole). Longitude describes the east-west position of a point as the angle between the plane containing the Prime Meridian and the line between the point and the Earth's center. The Prime

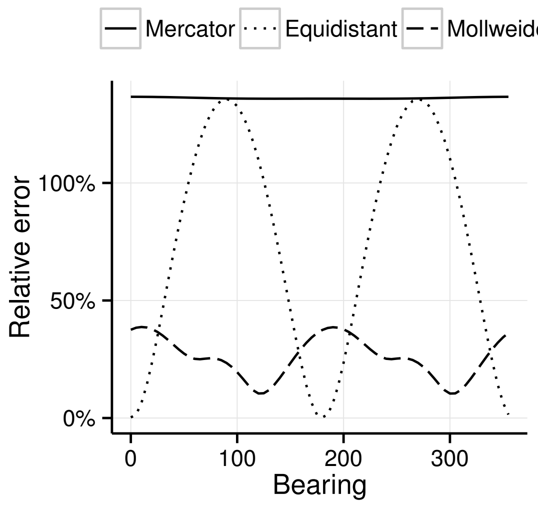


**Figure 1: Average distance measurement error over the Mercator, Equidistant Cylindrical and Mollweide projections. One party is located at a random location with latitude $65^\circ$, while the other is 10 km away, at different bearings.**

Meridian has longitude $0^\circ$, the values range from $-180^\circ$ to $180^\circ$ with negative values to the west of Prime Meridian.

The shortest path between two points on a sphere or an ellipsoid is along the *great circle*, i.e., the intersection between a sphere and a plane which passes through the center of the sphere. When viewed on a projection, the shortest distance may appear as a curve – this is why the airplanes trajectories usually appear as arcs on in-flight maps. Distance between two points on the Earth ellipsoid can be computed using Vincenty's formula [45], or Karney's algorithm [27]. The error introduced be these techniques is less than 1 mm [45] and 15 nm [27] respectively.

Faster – although less accurate – methods for distance computation exist. We describe three of these methods next.

### 3.1 Distances over UTM Projection

A straightforward approach to computing distances on Earth is to transform the problem into computing distances on a plane. This is done by selecting a projection (e.g., a map) of Earth's surface, or part of it. Two points are then mapped to the projection, where Euclidean distance can be computed. Although simple, this approach has at least two important drawbacks. First, two parties willing to measure their distance must agree on the same projection in order to properly map their locations. Second, and more importantly, it introduces errors.

Different projections (and projection sizes) have different properties, and lead to different distance measurement errors. Figure 1 illustrates measurement errors on the Mercator, Equidistant Cylindrical and Mollweide whole Earth projections, compared to the "exact" distance, computed over the WGS84 ellipsoid using Vincenty's formula. All three whole Earth projections introduce very high average error on the measured distance. To limit this problem, projections are usually computed from small slices of Earth, instead of the whole surface. Universal Transversal Mercator (UTM) [8] is one such projections.

UTM divides Earth's surface in 60 longitudinal areas, called *zones*. The width of each zone is 6°, with no zone covering the polar regions. Position of a point is given by the zone number and its (*northing*, *easting*) coordinate pair, expressed in meters. Northing of a point is the distance on the projection of the point from the equator, while easting is the distance of the point from the zone's central meridian. For navigational purposes, negative coordinates are avoided by adding large constants to northing and easting. These constants are called "false northing" (which corresponds to 10,000 km) and "false easting" (500 km). Negative values do not pose any problem for distance computation, so we ignore false northing and false easting in the rest of the paper.

Once two points $C_A = (x_A, y_A)$, $C_B = (x_B, y_B)$ have been mapped to the same UTM zone, their distance can be computed as:

$$\mathsf{D}_{\mathrm{UTM}}(C_A, C_B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

This approach can only be used if both points lay in the same zone. Therefore, as mentioned in Section 1, UTM distance should be considered as a baseline for accuracy and performance.

## 3.2 Distances over ECEF Coordinate System

The Earth-Centered Earth-Fixed coordinate system uses cartesian coordinates to describe the position of any point on Earth. Each location is described using a triplet $(x, y, z)$, where $(0, 0, 0)$ is the center of Earth's mass (hence, Earth-Centered). All coordinates are expressed in meters.

The $x, y$ axes lay on the plane going through the Equator, with the $x$ axis "pointing" towards the Prime Meridian and the $y$ axis is oriented to make the system right-handed. The $z$ axis is oriented towards North Pole. In order to prevent coordinates on Earth to change with time, the $x$ and $y$ axes (and thus the whole coordinate system) rotate together with Earth (hence, Earth-Fixed). Spherical $(lat, lon)$ coordinates from WGS84, or any other coordinate system, can be easily converted to ECEF coordinates.

Distance $\mathsf{D}_{\mathrm{ECEF}}(C_A, C_B)$ between two points $C_A = (x_A, y_A, z_A)$ and $C_B = (x_B, y_B, z_B)$ on the great circle can be computed as:

$$\begin{aligned} a =& \frac{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}{4R^2} \\ c =& 2\,\mathrm{atan}\left(\sqrt{a/(1-a)}\right) \\ d =& Rc \end{aligned}$$

where $\mathsf{D}_{\mathrm{ECEF}}(C_A, C_B) = d$.

Value $a$ is the squared Euclidean distance between points $C_A$ and $C_B$ (divided by a constant) in three-dimensional space, i.e., the distance between $C_A$ and $C_B$ over a straight line that cuts through Earth. Therefore, in order to compute the distance on the surface, we derive the central angle $c$ between $C_A$, Earth's center, and $C_B$. Finally, the length of the arc between the two points, represented as $d$, corresponds to the distance over the surface. For the angle and arc length computation, we use a spherical approximation of Earth with radius $R$.

## 3.3 Distances on a Sphere Using Haversine

According to [42], angular distance between two points (represented as (*latitude*, *longitude*) pairs) on a sphere has been traditionally expressed using the cosine formula. Let $C_A = (lat_A, lon_A)$ and $C_B = (lat_B, lon_B)$ denote the spherical coordinates of Alice and Bob, respectively. Distance between $C_A$ and $C_B$ is computed using the cosine formula as:

$$\begin{aligned} c =& \cos^{-1}(\sin(lat_A)\sin(lat_B) \\ &+ \cos(lat_A)\cos(lat_B)\cos(lon_A - lon_B)) \end{aligned}$$

where $c$ is central angle between $C_A$ and $C_B$. When used in conjunction with limited precision arithmetics (e.g., in privacy-preserving protocols, where precision affects performance), the cosine formula may introduce significant measurement error when two points are in close proximity. As an example, when Alice and Bob are one kilometer apart, both on the equator, $lon_A - lon_B \approx 0.083°$, and $\cos(lon_A - lon_B) \approx 0.9999999894$. Approximating $\cos(\alpha)$ to 8 significant digits, the resulting distance between Alice and Bob is zero.

The haversine formula [42] is often used as a replacement for the cosine formula in order to reduce measurement errors introduced by cosine of small angles. Haversine, i.e., *half the versed sine*, is defined as $\mathrm{hav}(\theta) = (1 - \cos(\theta))/2$.

Let $R$ be the radius of Earth. The haversine formula allows us to compute the distance $\mathsf{D}_{\mathrm{HS}}(C_A, C_B)$ as follows:

$$\begin{aligned} a =& \mathrm{hav}(lat_A - lat_B) \\ &+ \cos(lat_A)\cos(lat_B)\mathrm{hav}(lon_A - lon_B) \\ c =& 2\,\mathrm{atan}\left(\sqrt{a/(1-a)}\right) \\ d =& Rc \end{aligned}$$

where $\mathsf{D}_{\mathrm{HS}}(C_A, C_B) = d$. (The meaning of $a$, $c$ and $d$ is the same as that in the ECEF formula from Section 3.2.)

Given the trigonometric identity $\mathrm{hav}(\theta) = \sin^2(\theta/2)$, we can rewrite the haversine formula in such a way that $lat_A - lat_B$ and $lon_A - lon_B$ are not used as arguments for cosine:

$$\begin{aligned} a =& \sin^2((lat_A - lat_B)/2) \\ &+ \cos(lat_A)\cos(lat_B)\sin^2((lon_A - lon_B)/2) \\ c =& 2\,\mathrm{atan}\left(\sqrt{a/(1-a)}\right) \\ d =& Rc \end{aligned}$$

Therefore, although the cosine and haversine formulas are *mathematically* identical, the latter provides better accuracy with limited machine precision when Alice and Bob are separated by a small angle.

In the rest of the paper, we use WGS84 as a reference system for (*latitude*, *longitude*) coordinates.

## 3.4 Proximity Testing

Alice may be interested in determining only if Bob is in close proximity, i.e., within an arbitrary distance $\varepsilon$. Clearly, Alice could compute the distance between her and Bob along Earth's surface, and then check if it is greater than $\varepsilon$.

For ECEF and haversine distance, however, Alice can use a simple shortcut; she can convert $\varepsilon$ to $a_\varepsilon$ as follows:

$$a_\epsilon = \frac{(\tan\frac{\epsilon}{2R})^2}{1 + (\tan\frac{\epsilon}{2R})^2}$$

This way, $a_\varepsilon$ represents the threshold expressed as straight-through-Earth distance in the three-dimensional space. Since the distance between Alice and Bob is a monotonically increasing function of $a$, we have that $a < a_\epsilon \iff d < \varepsilon$.

## 4. CRYPTOGRAPHIC PRELIMINARIES

We use the term *adversary* to refer to insiders, i.e., Alice and Bob. This includes the case when one of the two parties is compromised. Outside adversaries, e.g., those who can eavesdrop on the communication channel, are not considered since their actions can be mitigated via standard network security techniques.

Our protocols are secure in the presence of semi-honest (also known as honest-but-curious or passive) participants. In this model, while participants follow prescribed protocol behavior, they might try to learn additional information beyond that obtained during normal protocol execution. Formally [13]:

DEFINITION 1. *Let $P_1$ and $P_2$ participate in protocol $\pi$ that computes function $f(\mathsf{in}_1, \mathsf{in}_2) = (\mathsf{out}_1, \mathsf{out}_2)$, where $\mathsf{in}_i$ and $\mathsf{out}_i$ denote $P_i$'s input and output, respectively. Let $\mathrm{VIEW}_\pi(P_i)$ denote the view of participant $P_i$ during the execution of protocol $\pi$. More precisely, $P_i$'s view is formed by its input, internal random coin tosses $r_i$, and messages $m_1, \ldots, m_t$ passed between the parties during protocol execution: $\mathrm{VIEW}_\pi(P_i) = (\mathsf{in}_i, r_i, m_1, \ldots, m_t)$.*

*We say that protocol $\pi$ is secure against semi-honest adversaries if for each party $P_i$ there exists a probabilistic polynomial time simulator $S_i$ such that:*

$$\{S_i(\mathsf{in}_i, f_i(\mathsf{in}_1, \mathsf{in}_2))\} \overset{c}{\equiv} \{\mathrm{VIEW}_\pi(P_i), \mathsf{out}_i\}$$

*where $\overset{c}{\equiv}$ denotes computationally indistinguishability.*

**Homomorphic Encryption.** Our constructions use a semantically secure (public key) additively homomorphic encryption scheme. Let $[\![m]\!]$ indicate the encryption of message $m$ using a homomorphic encryption scheme. (To keep notation simple, we omit specifying the public key used for encryption. All encryptions in our protocols are performed under Alice's public key.)

In an additively homomorphic encryption scheme, $[\![m_1]\!] \cdot [\![m_2]\!] = [\![m_1 + m_2]\!]$, which also implies that $[\![m]\!]^a = [\![m \cdot a]\!]$. While any encryption scheme with the above properties (such as the well known Paillier encryption scheme [37]) suffices for the purposes of this work, the construction due to Damgård et al. [7, 6] (DGK hereafter) is of particular interest here because it is fast and it produces small ciphertexts. In DGK a public key consists of (1) a (small, possibly prime) integer $u$ that defines the plaintext space; (2) $k$-bit RSA modulus $N = p \cdot q$ such that $p$ and $q$ are $k/2$-bit primes, $v_p$ and $v_q$ are $t$-bit primes, and $uv_p|(p-1)$ and $uv_q|(q-1)$; and (3) elements $g, h \in \mathbb{Z}_N^*$ such that $g$ has order $uv_pv_q$ and $h$ has order $v_pv_q$. Given a message $m \in \mathbb{Z}_u$, encryption is performed as $[\![m]\!] = g^m h^r \bmod N$, where $r \leftarrow \{0,1\}^{2.5t}$. We refer the reader to [7, 6] for any additional information.

**Garbled circuit evaluation.** Originated in Yao's work [48], garbled circuit evaluation allows two parties to securely evaluate any function represented as a boolean circuit. The basic idea is that, given a circuit composed of gates, Bob creates a garbled circuit by assigning to each wire two randomly chosen keys. Bob also encodes gate information in a way that given keys corresponding to the input wires (encoding specific inputs), the key corresponding to the output of the gate on those inputs can be recovered. Alice then evaluates the circuit using keys corresponding to inputs of both Alice and Bob (without learning anything in the process). At the end, the result of the computation can be recovered by linking the output keys to the bits which they encode.

Recent literature provides optimizations that reduce computation and communication overhead associated with circuit construction and evaluation. Kolesnikov and Schneider [29] describe an optimization that permits XOR gates to be evaluated *for free*, i.e., there is no communication overhead associated with such gates and their evaluation does no involve cryptographic functions. Pinkas et al. [38] additionally give a mechanism for reducing communication complexity of binary gates by 25%: now each gate can be specified by encoding only three outcomes of the gate instead of all four. Finally, Kolesnikov et al. [28] improve the complexity of certain commonly used operations such as addition, multiplication, comparison, etc. by reducing the number of non-XOR gates: adding two $n$-bit integers requires $5n$ gates, $n$ of which are non-XOR gates; comparing two $n$-bit integers requires $4n$ gates, $n$ of which are non-XOR gates; and computing the minimum of $t$ $n$-bit integers (without the location of the minimum value) requires $7n(t-1)$ gates, $2n(t-1)$ of which are non-XOR gates.

With the above techniques, evaluating a non-XOR gates involves one invocation of the hash function. During garbled circuit evaluation, Alice directly obtains keys corresponding to the Bob's inputs from the Bob, and engages in the oblivious transfer (OT) protocol to obtain keys corresponding to its own input.

**Oblivious Transfer.** In 1-out-of-2 Oblivious Transfer, $OT_1^2$, one party, the sender, has as its input two strings $m_0, m_1$ and another party, the receiver, has as its input a bit $b$. At the end of the protocol, the receiver learns $m_b$ and the sender learns nothing. Similarly, in 1-out-of-$N$ OT the receiver obtains one of the $N$ strings held by the sender. There is a rich body of research literature on OT, and in this chapter we use its efficient implementation from [35] as well as techniques from [22] that reduce a large number of OT protocol executions to $\kappa$ of them, where $\kappa$ is the security parameter.

**Discretization.** Because our privacy-preserving protocols are designed to work on integer values, we map each real-valued protocol input to integers according to the following formula: $\mathsf{discretize}_e(x) = \lfloor 10^e \cdot x + 0.5 \rfloor$. The $e$ parameter controls the shift of the decimal point (the choice of appropriate values for $e$ is discussed in Section 7). Therefore, the higher the $e$, the more digits after the decimal point are preserved.

## 5. OUR PROTOCOLS

In this section, we present our PP-UTM, PP-ECEF and PP-HS protocols. These protocols privately compute the distance between two points in the UTM, ECEF and WGS84 coordinate systems, respectively.

### 5.1 PP-UTM Protocol

Our PP-UTM protocol computes Euclidean distance between two points laying on the same UTM zone. (Due to this limitation, and as mentioned earlier, we consider this protocol

**Input:** Alice: her position, expressed as coordinates $C_A = (x_A, y_A)$ with respect to a specific UTM zone; her public/private keypair. Bob: his position, expressed as coordinates $C_B = (x_B, y_B)$ in the same zone as Alice; Alice's public key $pk_A$.
**Output:** Alice learns $\mathsf{D}_{\text{UTM}}(C_A, C_B)$, defined as the Euclidean distance between $C_A$ and $C_B$.
**Protocol steps**:

1. Alice computes $[\![x_A^2 + y_A^2]\!], [\![-2x_A]\!], [\![-2y_A]\!]$ and sends these ciphertexts to Bob.
2. Bob computes:
$$\begin{aligned}[\![a]\!] &= [\![x_A^2 + y_A^2]\!] \cdot [\![-2x_A]\!]^{x_B} \cdot [\![-2y_A]\!]^{y_B} \cdot [\![x_B^2 + y_B^2]\!] \\ &= [\![(x_A^2 + x_B^2 - 2x_Ax_B) + (y_A^2 + y_B^2 - 2y_Ay_B)]\!] \\ &= [\![\mathsf{D}_{\text{UTM}}(C_A, C_B)^2]\!]\end{aligned}$$
and sends $[\![a]\!]$ back to Alice.
3. Alice decrypts $[\![a]\!]$ and outputs $\mathsf{D}_{\text{UTM}}(C_A, C_B)$ as $\sqrt{a}$.

**Figure 2: PP-UTM Protocol**

**Input:** Alice: her position, expressed as coordinates $C_A = (x_A, y_A, z_A)$ in the ECEF coordinate system; her public/private keypair. Bob: his position, expressed as coordinates $C_B = (x_B, y_B, z_B)$ in the same coordinate system; Alice's public key $pk_A$.
**Output:** Alice learns $\mathsf{D}_{\text{ECEF}}(C_A, C_B)$, defined as the Euclidean distance between $C_A$ and $C_B$.
**Protocol steps**:

1. Alice computes $[\![x_A^2 + y_A^2 + z_A^2]\!], [\![-2x_A]\!], [\![-2y_A]\!], [\![-2z_A]\!]$ and sends these ciphertexts to Bob.
2. Bob computes:
$$\begin{aligned}[\![a']\!] &= [\![x_A^2 + y_A^2 + z_A^2]\!] \cdot [\![-2x_A]\!]^{x_B} \cdot [\![-2y_A]\!]^{y_B} \cdot [\![x_B^2 + y_B^2 + z_B^2]\!] \cdot [\![-2z_A]\!]^{z_B} \\ &= [\![(x_A^2 + x_B^2 - 2x_Ax_B) + (y_A^2 + y_B^2 - 2y_Ay_B) + (z_A^2 + z_B^2 - 2z_Az_B)]\!]\end{aligned}$$
and sends $[\![a']\!]$ back to Alice.
3. Alice decrypts $[\![a']\!]$, computes $a = a'/4R^2$ and outputs $\mathsf{D}_{\text{ECEF}}(C_A, C_B)$ as $2R\,\text{atan}\left(\sqrt{a/(1-a)}\right)$.

**Figure 3: PP-ECEF Protocol**

as a baseline for evaluating both accuracy and performance of our PP-ECEF and PP-HS protocols.)

Alice and Bob must determine whether their coordinates lay on the same zone, prior to running the PP-UTM protocol. To do so, they can assign a unique identifier to each zone, and then compare their respective identifiers using any protocol that implements private equality test. As an example, they can use the following simple algorithm:

> Let $z_A$ represent Alice's zone, and $z_B$ – Bob's zone. Alice sends $[\![z_A]\!]$ to Bob, who computes $[\![d]\!] = ([\![z_A]\!] \cdot [\![-z_B]\!])^r = [\![r \cdot (z_A - z_B)]\!]$, where $r$ is a random value uniformly chosen from the message space. $[\![d]\!]$ is sent back to Alice, who decrypts it. If Alice and Bob belong to the same zone (i.e., $z_A = z_B$), then $d = 0$. Otherwise, $d$ is uniformly distributed in the message space, and does not reveal any additional information about $z_B$.

Even if Alice and Bob are not in the same zone, Alice may still participate in the UTM protocol – although the result of the computation will be unrelated to their correct distance. This will prevent Bob from learning the output of the private equality test.

Alice and Bob discretize their coordinates, as discussed in Section 4, prior to using them in the protocol. The PP-UTM protocol is shown in Figure 2.

## 5.2 PP-ECEF Protocol

Our PP-ECEF protocol computes the distance between Alice and Bob in the ECEF coordinate system. Computation of distance is divided in two phases:

1. Alice and Bob interact to privately compute the straight-line (Euclidean) distance between their respective inputs; only Alice learns the results (i.e., variable $a$ in Section 3.2).
2. Then Alice converts this straight-line distance to the corresponding distance on Earth's surface (i.e., variable $d$ in Section 3.2).

Since Earth is represented as a sphere, step 2. is independent from the absolute locations of Alice and Bob, and therefore does not require any private input from the parties.

Similarly to the PP-UTM protocol, Alice and Bob's coordinates are discretized as discussed in Section 4. The PP-ECEF is detailed in Figure 3.

## 5.3 PP-HS Protocol

The PP-HS protocol involves private computation of haversine formula over Alice and Bob's private input. However, if the formula is used *as is* for the construction of a privacy-preserving protocol, it requires Alice and Bob to interactively compute $\sin^2((lat_A - lat_B)/2)$ and $\sin^2((lon_A - lon_B)/2)$. While both *can* be computed within a privacy-preserving framework (e.g., by implementing Taylor series expansion as shown in [31]), the cost of the resulting protocol would be prohibitive, especially on resource-constrained devices such as smartphones and tablets.

However, this can be avoided by simply using the following well-known trigonometric identity:

$$\sin(\varphi - \psi) = \sin(\varphi)\cos(\psi) - \cos(\varphi)\sin(\psi).$$

After we apply this identity to the haversine formula from Section 3.3, we can rewrite $a$ as:

$$\begin{aligned}a =& (\sin(lat_A/2)\cos(lat_B/2) - \cos(lat_A/2)\sin(lat_B/2))^2 \\ &+ \cos(lat_A)\cos(lat_B) \\ &\cdot (\sin(lon_A/2)\cos(lon_B/2) - \cos(lon_A/2)\sin(lon_B/2))^2\end{aligned}$$

**Input:** Alice: her position, expressed as coordinates $C_A = (lat_A, lon_A)$ as defined in Section 3 and her public/private keypair; Bob: his position, expressed as coordinates $C_B = (lat_B, lon_B)$ and Alice's public key.
**Output:** Alice learns $\mathsf{D}_{\rm HS}(C_A, C_B)$ as defined in Section 3.
**Protocol steps**:

1. Let $\alpha, \dots, \mu$ defined as in Section 3. Alice computes:
$$[\![\alpha^2]\!], [\![-2\alpha\gamma]\!], [\![\gamma^2]\!], [\![\zeta\eta\theta^2\lambda^2]\!], [\![-2\zeta\eta\theta\lambda]\!], [\![\zeta\eta]\!]$$
and sends the ciphertexts to Bob.
2. Bob computes:
$$\begin{aligned}[\![a]\!] &= [\![\alpha^2]\!]^{\beta^2} \cdot [\![-2\alpha\gamma]\!]^{\beta\delta} \cdot [\![\gamma^2]\!]^{\delta^2} \cdot [\![\zeta\eta\theta^2\lambda^2]\!] \cdot [\![-2\zeta\eta\theta\lambda]\!]^{\mu\nu} \cdot [\![\zeta\eta]\!]^{\mu^2\nu^2} \\ &= [\![\alpha^2\beta^2 - 2\alpha\beta\gamma\delta + \gamma^2\delta^2 + \zeta\eta\theta^2\lambda^2 - 2\zeta\eta\theta\lambda\mu\nu + \zeta\eta\mu^2\nu^2]\!]\end{aligned}$$
3. Bob sends $[\![a]\!]$ to Alice, which decrypts it and outputs $\mathsf{D}_{\rm HS}(C_A, C_B)$ as $2R\,\mathrm{atan}\left(\sqrt{a/(1-a)}\right)$.

**Figure 4: PP-HS Protocol**

Let us denote the cosine/sine values with greek letters, for more concise notation, as:

$$\begin{array}{ll} \alpha = \cos(lat_A/2) & \eta = \cos(lat_B) \\ \beta = \sin(lat_B/2) & \theta = \sin(lon_A/2) \\ \gamma = \sin(lat_A/2) & \lambda = \cos(lon_B/2) \\ \delta = \cos(lat_B/2) & \mu = \cos(lon_A/2) \\ \zeta = \cos(lat_A) & \nu = \sin(lon_B/2) \end{array}$$

We can then rewrite and expand $a$ as:

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

With this formulation, computing $a$ does not involve joint evaluation of any trigonometric function on private input. In fact, each party can separately compute their share of the formula, and combine their shares using only operations over encrypted data.

Similarly to the PP-ECEF protocol, computing the distance between Alice and Bob given $a$ does not require any private input from either party. (Moreover, $a$ can be easily reconstructed from the distance between the protocol participants.) For this reason, our PP-HS protocol allows Alice to learn $a$, and to subsequently convert it to the actual distance on Earth's surface – in the unencrypted domain.

Instead of discretizing their coordinates, Alice and Bob discretize values $\alpha, \dots, \nu$. The protocol is illustrated in Figure 4.

## 5.4 Privacy-Preserving Proximity Testing with Unconditional Response

We extend the protocols presented in sections 5.1-5.3 to perform privacy-preserving proximity testing (PPPT). PPPT reveals a single bit of information to Alice, which represents:

$$\mathrm{PPPT}_{\mathsf{D}(\cdot,\cdot)}(C_A, C_B, \varepsilon) \triangleq \left(\mathsf{D}(C_A, C_B) \overset{?}{<} \varepsilon\right)$$

where $\mathsf{D}(\cdot,\cdot)$ is a distance function (in our case, $\mathsf{D}$ is either $\mathsf{D}_{\rm UTM}(\cdot,\cdot)$, $\mathsf{D}_{\rm ECEF}(\cdot,\cdot)$, or $\mathsf{D}_{\rm HS}(\cdot,\cdot)$), $C_A$ and $C_B$ are Alice's and Bob's coordinates, and $\varepsilon$ is an arbitrary distance below which Alice and Bob are considered to be in close proximity.

Additionally, Bob can decide whether to faithfully participate in the PPPT protocol, allowing Alice to compute the correct results, or to defect, forcing Alice to compute a bit of his choice. This may be preferable to Bob than simply opting out of the protocol. In fact, by not interacting with Alice, Bob may reveal information about his location.

In order to provide different performance tradeoffs, we implemented PPPT using two different privacy-preserving comparison techniques: (1) the privacy-preserving homomorphic comparison protocol of Erkin et al. [10], and (2) the garbled circuits of Kolesnikov et al. [28].

**Homomorphic Comparison of [10].** This protocol is based on the observation that $a < a_\varepsilon$ is true iff the $l+1$-th bit of $w = 2^l + a - a_\varepsilon$, denoted $w_l$ from now on, is 0 (for $2^l > a - a_\varepsilon$). Given $[\![a]\!]$, $[\![a_\varepsilon]\!]$, encryption of $w$ is computed by Bob as $[\![w]\!] = [\![2^l]\!] \cdot [\![a]\!] \cdot [\![a_\varepsilon]\!]^{-1}$. Encryption of the $l$-th bit of $w$ is then computed as $[\![w_l]\!] = [\![2^{-l} \cdot (w - (w \bmod 2^l))]\!]$. Value $w$ is available to Bob only in encrypted form, and computing $w \bmod 2^l$ in the encrypted domain requires interaction between Alice and Bob: Bob "masks" $[\![w]\!]$ by selecting a random value $r$ and computing $[\![w']\!] = [\![w]\!] \cdot [\![r]\!]$. Then, Bob sends $[\![w']\!]$ to Alice, who decrypts it and returns the encryption of $c = w' \bmod 2^l$ to Alice. Next, Bob "unmasks" $[\![c]\!]$ by computing $[\![c]\!] \cdot [\![r]\!]^{-1} = [\![w \bmod 2^l]\!]$. We refer the reader to [10] for additional details.

After Bob computes $[\![a]\!]$ in our distance protocols ($[\![a']\!]$ for PP-ECEF), he does not return it to Alice. Instead, he computes – with Alice's help – value $[\![w - (w \bmod 2^l)]\!]$, which is 0 if and only if $\mathsf{D}(C_A, C_B) < \varepsilon$. This value is then sent to Alice, who decrypts it and outputs the corresponding plaintext. Since Bob operates on both $a$ and $a_\varepsilon$ in encrypted form, Alice can choose not to reveal the threshold to Bob during the protocol.

**Comparison Based on [28].** In order to reduce the cost of the integer comparison circuit, Kolesnikov et al. [28] minimize the use of non-XOR gates. The result is a circuit that uses $4n$ gates, out of which $n$ are non-XOR, to compare two unsigned $n$-bit integers: the comparison circuit contains $n$ 1-bit comparators with carry, which use three XOR gates and one AND gate (see figures 5 and 6 of [28]). The advantage of this technique is that a significant part of the computation can be performed offline by Alice and Bob, resulting in faster on-line computation.

After Bob computes $[\![a]\!]$ ($[\![a']\!]$ for PP-ECEF) using our distance protocols, he selects a random value $s \leftarrow \{0,1\}^{\kappa+l}$ where $\kappa$ is the security parameter and $l$ is the number of bits used to represent location information. Bob then *blinds* $a$ as $[\![\bar{a}]\!] = [\![a + s]\!]$ and returns $[\![\bar{a}]\!]$ to Alice. $s$ is uniformly selected from $\{0,1\}^{\kappa+l}$ instead of the message space of DGK for efficiency reasons. This allows us to achieve statistical hiding for $a$, instead of unconditional hiding, since the blinding value is $\kappa$ bits longer that the value it protects [3].

Alice and Bob then perform circuit evaluation, where Alice's input is $\overline{a}$ and Bob's input is $a_\varepsilon + s$. At the end of the circuit evaluation, Alice learns the correct result, since $a < a_\varepsilon \iff \overline{a} < a_\varepsilon + s$.

**Unconditional Response.** In order to return an *unconditional positive* to Alice, when using homomorphic comparison Bob discards $[\![a]\!]$ and $[\![a_\varepsilon]\!]$ and replaces them with $[\![\overline{a}]\!]$ and $[\![\overline{a_\varepsilon}]\!]$, for two arbitrary values $\overline{a}$ and $\overline{a_\varepsilon}$ such that $\overline{a} < \overline{a_\varepsilon}$. Analogously, an *unconditional negative* can be returned by selecting $\overline{a} > \overline{a_\varepsilon}$.

When using the garbled circuit for comparison, Bob's input to the circuit is 0 for unconditional negative ($0 < \overline{a}$ holds with overwhelming probability), and $s + a_{max} + 1$ for unconditional positive, where $a_{max}$ is the largest value that $a$ ($a'$ for PP-ECEF) can assume. ($s + a_{max} + 1 > \overline{a}$ always holds.)

The security of the comparison protocols guarantees that Alice cannot distinguish between a correct result and an unconditional response.

## 6. SECURITY ANALYSIS

Security of our protocols relies on the security of the underlying building blocks. In particular, we need to assume that the underlying homomorphic encryption scheme is semantically secure.

We instantiate $[\![\cdot]\!]$ using the DGK encryption scheme, which has been shown to be semantically secure under a hardness assumption that uses subgroups of an RSA modulus [7, 6]. The privacy-preserving comparison protocol of Erkin et al. was shown to be secure in [10], and therefore we do not include it in our analysis.

To show the security of the protocols, we informally sketch how to simulate the view of each party using its inputs and outputs alone. If such simulation is indistinguishable from the real execution of the protocol, for semi-honest parties this implies that the protocols do not reveal any unintended information to the participants (i.e., they learn only the output and what can be deduced from their respective inputs and outputs).

**Security of PP-UTM.** Alice's view of the protocols consists of the encryption and decryption keys for $[\![\cdot]\!]$, and ciphertext $[\![a]\!]$ from Bob. Alice's output is $\sqrt{a}$. Simulator $S_A$ provides Alice with decryption key for $[\![\cdot]\!]$ as input. It then encrypts the protocol output as $[\![(\sqrt{a})^2]\!] = [\![a]\!]$ and sends it to Alice. Since $[\![a]\!]$ is properly distributed, Alice cannot distinguish between the simulation and a real execution of the protocol. Therefore, the protocol is secure against a curious Alice.

Bob's view of the protocol consists in Alice's public key, and three ciphertexts from Alice, namely $[\![x_A^2 + y_A^2]\!]$, $[\![-2x_A]\!]$, $[\![-2y_A]\!]$. Bob has no output. Simulator $S_B$ selects a random pair $(\bar{x}_A, \bar{y}_A)$ and sends $[\![\bar{x}_A^2 + \bar{y}_A^2]\!]$, $[\![-2\bar{x}_A]\!]$ and $[\![-2\bar{y}_A]\!]$ to Bob. The semantic security of $[\![\cdot]\!]$ prevents Bob from determining that $\bar{x}_A^2 + \bar{y}_A^2$, $-2\bar{x}_A$ and $-2\bar{y}_A$ are not distributed properly. Therefore, no PPT algorithm can distinguish $[\![\bar{x}_A^2 + \bar{y}_A^2]\!]$, $[\![-2\bar{x}_A]\!]$ and $[\![-2\bar{y}_A]\!]$ from the encryption of properly distributed values. For this reason, Bob cannot distinguish between interaction with the $S_B$ and with a honest Alice. Hence the protocol is secure against a curious Bob.

**Security of PP-ECEF.** Alice's view of the protocols consists of the encryption and decryption keys for $[\![\cdot]\!]$, and ciphertext $[\![a]\!]$ from Bob. Alice's output is $2R\,\mathrm{atan}(\sqrt{a/(1-a)})$. Simulator $S_A$ provides Alice with decryption key for $[\![\cdot]\!]$ as input. It then computes $a$ from the protocol output and sends $[\![a]\!]$ to Alice. Since $[\![a]\!]$ is properly distributed, Alice cannot distinguish between the simulation and a real execution of the protocol. Therefore, the protocol is secure against a curious Alice.

Bob's view of the protocol consists in Alice's public key, and four ciphertexts from Alice, namely $[\![x_A^2 + y_A^2 + z_A^2]\!]$, $[\![-2x_A]\!]$, $[\![-2y_A]\!]$, and $[\![-2z_A]\!]$. Bob has no output. Simulator $S_B$ selects a random set of coordinates $(\bar{x}_A, \bar{y}_A, \bar{z}_A)$ and sends $[\![\bar{x}_A^2 + \bar{y}_A^2 + \bar{z}_A^2]\!], [\![-2\bar{x}_A]\!], [\![-2\bar{y}_A]\!], [\![-2\bar{z}_A]\!]$ to Bob. The semantic security of $[\![\cdot]\!]$ prevents Bob from determining that $\bar{x}_A^2 + \bar{y}_A^2 + \bar{z}_A^2$, $-2\bar{x}_A$, $-2\bar{y}_A$ and $-2\bar{z}_A$ are not distributed properly. Therefore, no PPT algorithm can distinguish $[\![\bar{x}_A^2 + \bar{y}_A^2 + \bar{z}_A^2]\!]$, $[\![-2\bar{x}_A]\!]$, $[\![-2\bar{y}_A]\!]$ and $[\![-2\bar{z}_A]\!]$ from the encryption of properly distributed values. For this reason, Bob cannot distinguish between interaction with the $S_B$ and with a honest Alice. Hence the protocol is secure against a curious Bob.

**Security of PP-HS.** Alice's view of the protocols consists of the encryption and decryption keys for $[\![\cdot]\!]$, and ciphertext $[\![a]\!]$ from Bob. Alice's output is $2R\,\mathrm{atan}(\sqrt{a/(1-a)})$. Simulator $S_A$ provides Alice with decryption key for $[\![\cdot]\!]$ as input. It then computes $a$ from the protocol output, encrypts it as $[\![a]\!]$, and sends it to Alice. Since $[\![a]\!]$ is properly distributed, Alice cannot distinguish between the simulation and a real execution of the protocol. Therefore, the protocol is secure against a curious Alice.

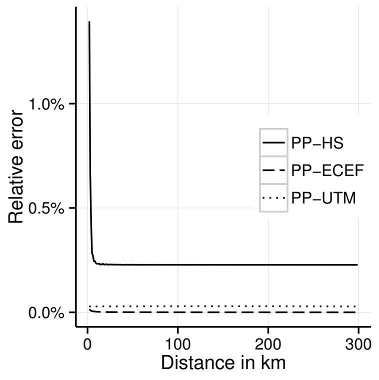
Bob's view of the protocol consists in Alice's public key, and six ciphertexts from Alice, namely $[\![\alpha^2]\!]$, $[\![-2\alpha\gamma]\!]$, $[\![\gamma^2]\!]$, $[\![\zeta\eta\theta^2\lambda^2]\!]$, $[\![-2\zeta\eta\theta\lambda]\!]$ and $[\![\zeta\eta]\!]$. Bob has no output. Simulator $S_B$ selects six random values $s_1, \ldots, s_6$ from the message space of $[\![\cdot]\!]$, encrypts them and sends $[\![s_1]\!], \ldots, [\![s_6]\!]$ to Bob. The semantic security of $[\![\cdot]\!]$ prevents Bob from distinguishing $[\![s_1]\!], \ldots, [\![s_6]\!]$ from the encryption of properly distributed values. For this reason, Bob cannot distinguish between interaction with the $S_B$ and with a honest Alice. Hence the protocol is secure against a curious Bob.

**Security of Comparison with Garbled Circuit of [28].** It is well known that Garbled Circuits can be used to implement secure two party computation in the honest but curious model [14, 28]. When using it as a building block, we have to show that combining our distance protocol with the comparison circuit of [28] does not affect security.

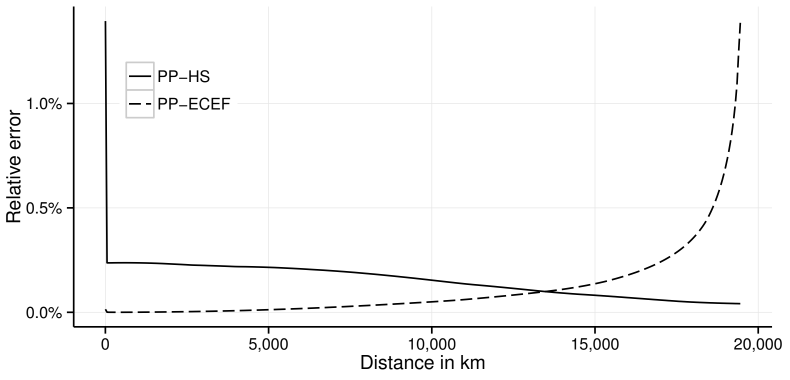
Before the circuit execution, Bob reveals $[\![\overline{a}]\!] = [\![a + s]\!]$ to Alice, where $a \in \{0,1\}^l$ is a representation of the distance between the two parties and $s \leftarrow \{0,1\}^{\kappa+l}$ is chosen uniformly at random. We argue that a computationally-bound adversary does not learn information about $a$ from $\overline{a}$, since $s$ provides statistical hiding. (Unconditional hiding can be obtained selecting $s$ from $s \leftarrow \{0,1\}^l$ and performing modulo subtraction within the circuit. However, this could negatively affect performance since it would add complexity to the comparison circuit.) More specifically, the success probability of the adversary decreases exponentially in $\kappa$.

Bob's view does not include any unencrypted values, so the security of the combined protocol simply relies on the security of its building blocks.

**Remark on Location Triangulation.** A potential security weakness inherent in distance computation is the ability of a set of three or more colluding parties, acting as Alice, to



(a)



(b)

**Figure 5: Comparison of PP-UTM, PP-ECEF and PP-HS accuracy for short (a) and long (b) distances. Discretization is performed with 1-meter accuracy for PP-UTM and PP-ECEF and roughly equivalent accuracy (9 digits) for PP-HS. Relative error is plotted from the 2 km distance.**

precisely determine the location of Bob through triangulation. This is not an issue specific to our protocol, but rather to the functionality that our protocol implements. Any protocol implementing the same functionality has the same issue. Similarly, proximity testing allows multiple colluding parties to determine Bob's exact location – although with significantly higher cost. We do not discuss this issue any further, since this is a limitation of the underlying functionality rather than of our protocols.

## 7. EVALUATION

We now present our protocol evaluation. We assess the proposed protocols in terms of accuracy and performance. For accuracy, we compare the exact distance with the output of the specific distance function (i.e., haversine formula, ECEF arc, Euclidean distance on UTM), after performing input discretization. To assess performance, we measure protocol execution time and bandwidth requirements. For garbled circuits, the time is divided into the *precomputation*, which can be done before knowing the input of either party, and *execution*. Both precomputation and execution require interaction between the protocol participants.

Our tests are performed on a prototype implementation, detailed below. For generality's sake, all tests were run on both a regular desktop computer and on a commodity Android device.

### 7.1 Accuracy

To evaluate the accuracy of our protocols, we use WGS84 as the reference shape of Earth. We select random pairs of coordinates and compute the reference distance on WGS84. This distance is then compared with the output of the protocols on the same coordinates. We report errors as a fraction of the WGS84 distance.

There are two main sources of imprecision in our protocols: (1) the distance computation methods we consider are inherently inaccurate, as they use a spherical approximation or a projection instead of the actual shape of Earth; (2) computation in the encrypted domain is performed on discretized values, i.e., over the integers; for efficiency reasons, the number of digits used for discretization is limited – this introduce approximation errors.

The effect of coordinate discretization on PP-UTM and PP-ECEF is that all points are mapped on a grid with fixed-size cells. Additions and subtractions of discretized values performed in the encrypted domain do not introduce further errors. Square root and trigonometric functions are computed in the unencrypted domain, and can therefore be performed with arbitrary precision. In our tests, we used 64-bit IEEE 754 floating point variables (i.e., `double` in Java). We evaluated cells of size 1 m and 100 m (i.e., $\mathsf{discretize}_e(\cdot)$ with $e = 0$ and $e = -2$ respectively). Let $u$ denote the cell size. Errors introduced in the encrypted domain are bounded by $\sqrt{2} \cdot u$ for PP-UTM and approximately $\sqrt{3} \cdot u$ for PP-ECEF.

This does not apply to PP-HS. In fact, the discretization is performed on the output of trigonometric functions computed over Alice and Bob's coordinates. As the number of digits increases, the error caused by discretization decreases exponentially. In all our experiments, we used $e = 9$, that is, 9 significant decimal digits. This value showed, in fact, a reasonable trade-off between cost and accuracy. Moreover, by further increasing $e$ the impact on accuracy was negligible.

The *relative* error of our protocols is shown in figures 5(a) and 5(b), for short and long distances respectively. The error on UTM projection is shown in Figure 5(a) only, since it is not meaningful to compare distances between coordinates laying on different zones (any two random points separated by over 300 km would likely fall in different zones).

PP-ECEF is better suited when the distance between the two parties is below 12,000 km. As a generic distance function (i.e., when there is no a-priory knowledge of the range of distances involved), both PP-HS and PP-ECEF provide accurate results. We compare the PP-ECEF protocol with existing protocols in Table 1.

Our prototype relies on the Java Geodesy Library [12] to compute distances on WGS84. The JScience library [25] is used to project WGS84 coordinates to UTM and ECEF, while the Java Map Projection Library [23] – to project spherical coordinates to the Mercator, Equidistant Cylindrical and Mollweide projections.

| | **PP-ECEF** | **Narayanan et. al.** [36] | **Louis** [50] | **Lester** [50] | **Pierre** [50] |
|---|---|---|---|---|---|
| Suitable for proximity testing | yes | yes | yes | yes | yes |
| Suitable for distance computation | yes | no | w/ small changes | w/ small changes | no |
| Location resolution independent of threshold | yes | no | yes | yes | no |
| Error characterization | linear with resolution | linear with threshold/resolution | linear with resolution | linear with resolution | linear with threshold/resolution |
| Max. error for threshold $\varepsilon < 100$ km | $\approx 3$ m (with 1 m resolution) | $> 135\%$ of $\varepsilon$ | $\approx 2$ m (with 1 m resolution) | $\approx 2$ m (with 1 m resolution) | $> 180\%$ of $\varepsilon$ |
| Requires third party | no | no | yes | yes | no |

**Table 1: Comparison of our PP-ECEF protocol to related work. For protocols that allow independent choice of resolution, 1m resolution was chosen for the 100m proximity test.**

## 7.2 Performance

Protocol performance was evaluated on a desktop computer (Intel Xeon E5420 2.5 GHz CPU, 16GB RAM) and on an Android smartphone (LG Nexus 4, Quad-core 1.5 GHz Krait CPU, 2 GB RAM). Both the distance computation protocol and the protocol for proximity testing were implemented in Java using the `BigInteger` API. For simplicity, the prototype implementation was single-threaded.

In this section we report time measurements only for computation; communication overhead is instead expressed in terms of amount of data exchanged between the parties during protocol execution. We ignore additional overhead necessary for establishing a secure channel between the Alice and Bob (e.g., via TLS).

In our experiments, we use 1024-bit modulus, 160-bit subgroup size for DGK cryptosystem. Plaintext space for the distance computation protocols and proximity testing based on homomorphic comparison was 48 bits for PP-UTM, 65 bits for PP-HS, PP-ECEF, and 165 bits for proximity testing with garbled circuits. The security parameter $\kappa$ for garbled circuits was set to 80 in all of our experiments. Average execution time and bandwidth usage for distance computation is reported in Table 2, for proximity testing in Table 3.

Our experiments show that all protocols are practical on both our desktop setup and on a regular smartphone. Distance computation, regardless of the protocol, requires up to 41 ms on the desktop computer, and 78 ms on the smartphone. Small amount of computation, in conjunction with negligible bandwidth usage (below 1 kB) make our protocols suitable for resource-constrained devices. For distance computation, discretization parameters have no impact on bandwidth usage and negligible effects on computation time. Therefore, we show results for 1-meter discretization for PP-UTM and PP-ECEF, and roughly equivalent accuracy for PP-HS.

While PP-UTM is the fastest of our protocols, both PP-HS and PP-ECEF offer better accuracy for longer and shorter distances, respectively. Moreover, the cost increase over PP-UTM is small.

Proximity testing is significantly slower (and requires higher bandwidth) than simple distance computation. We perform tests using different discretization parameters, which impact computation and communication cost for both the homomorphic comparison protocol and garbled circuits.

On the desktop, comparison with garbled circuits takes 0.5 to 0.6 seconds, compared to 0.7 to 1.7 seconds with Erkin's protocol. The time and bandwidth for precomputation and protocol execution are listed separately in Table 3. On the other hand, bandwidth requirements for garbled circuits is between 37.5 and 41.0 kB, more than twice as much as the 7 to 14 kB required for Erkin's protocol.

Unlike the Erkin's comparison protocol, garbled circuits allow part of the computation to be done without providing any user input. This precomputation can be done in advance (e.g., while charging the smartphone or during extended periods of inactivity), but still requires communication between protocol participants. Without precomputation, comparison using garbled circuits takes 2.4 to 2.8 s, while the Erkin's protocol is faster with 1.2 to 2.8 seconds, depending on discretization and distance function. With precomputation, however, protocol execution time and bandwidth usage is reduced to 1.2-1.6 s.

| | **Time** | **Communication** |
|---|---|---|
| *Desktop* | | |
| PP-HS | 41.3 ms | 896 B |
| PP-ECEF | 34.9 ms | 640 B |
| PP-UTM | 26.5 ms | 512 B |
| *Smartphone* | | |
| PP-HS | 78.0 ms | 896 B |
| PP-ECEF | 66.7 ms | 640 B |
| PP-UTM | 53.5 ms | 512 B |

**Table 2: Cost of our privacy-preserving distance measurement protocols. Discretization is performed with 1-meter accuracy for PP-UTM and PP-ECEF and roughly equivalent accuracy (9 digits) for PP-HS.**

## 8. CONCLUSION

In this paper we introduced PP-UTM, PP-ECEF and PP-HS, three privacy-preserving protocols for secure distance computation and secure proximity testing. In contrast with previous work, our PP-ECEF and PP-HS protocols compute distances over a spherical surface instead of a plane. This allows us to provide significantly more accurate results over long distances, while incurring in very small overhead.

We rely on different techniques for distance computation – namely, Euclidean distance on a plane (which we use as a baseline) and distance on the surface of a sphere – in order to offer different trade-offs between accuracy and cost.

To the best of our knowledge, this is the first work that provides a thorough characterization of the geometrical error and the approximation introduced by input discretization on distances computed using privacy-preserving protocols.

We evaluated the cost of our protocols on a commodity Android device and on a desktop computer via a prototype implementation. Our analysis shows that our protocols are

| | With Homomorphic Comparison | | With Garbled Circuit | | | |
|---|---|---|---|---|---|---|
| **Proximity testing** | **Time** | **Comm.** | **Precomp.** | **Comm.** | **Exec.** | **Comm.** |
| *Desktop* | | | | | | |
| PP-HS (9 decimal digits) | 1.7 s | 17 kB | 409 ms | 22.1 kB | 124 ms | 18.9 kB |
| PP-ECEF (1 m discretization) | 1.3 s | 14 kB | 404 ms | 22.1 kB | 109 ms | 17.7 kB |
| PP-ECEF (100 m discretization) | 1.0 s | 10 kB | 404 ms | 22.1 kB | 105 ms | 16.6 kB |
| PP-UTM (1 m discretization) | 1.1 s | 10 kB | 401 ms | 22.1 kB | 102 ms | 16.6 kB |
| PP-UTM (100 m discretization) | 0.7 s | 7 kB | 401 ms | 22.1 kB | 104 ms | 15.4 kB |
| *Smartphone* | | | | | | |
| PP-HS (9 decimal digits) | 2.8 s | 17 kB | 1.2 s | 18.1 kB | 1.6 s | 16.4 kB |
| PP-ECEF (1 m discretization) | 2.2 s | 14 kB | 1.2 s | 18.1 kB | 1.4 s | 15.2 kB |
| PP-ECEF (100 m discretization) | 1.6 s | 10 kB | 1.2 s | 18.1 kB | 1.3 s | 14.1 kB |
| PP-UTM (1 m discretization) | 1.8 s | 10 kB | 1.2 s | 18.1 kB | 1.3 s | 14.1 kB |
| PP-UTM (100 m discretization) | 1.2 s | 7 kB | 1.2 s | 18.1 kB | 1.2 s | 12.9 kB |

**Table 3: Cost of proximity testing protocols with Homomorphic Comparison and Garbled Circuits.**

practical and can be run efficiently on standard smartphones. Finally, we analyzed the security of our protocols under standard assumptions.

## 9. ACKNOWLEDGEMENTS

Jaroslav Šeděnka kindly acknowledges the support by the ESF Project CZ.1.07/2.3.00/20.0051 Algebraic Methods in Quantum Logic of the Masaryk University. Supported by

european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitiveness
INVESTMENTS IN EDUCATION DEVELOPMENT

## 10. REFERENCES

[1] Find my Friends. `http://www.apple.com/icloud/features/find-my-friends.html`.

[2] A. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *Pervasive*, 2(1), 2003.

[3] M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In *ESORICS*, 2011.

[4] Boost Mobile Loopt. `http://www.boostmobile.com/boostloopt/how.html`.

[5] Cab4me. `http://www.cab4me.com`.

[6] I. Damgård, M. Geisler, and M. Krøigård. A correction to efficient and secure comparison for on-line auctions. Cryptology ePrint Archive, Report 2008/321, 2008.

[7] I. Damgård, M. Geisler, and M. Krøigård. Homomorphic encryption and secure comparison. *IJACT*, 2008.

[8] Defense Mapping Agency, Hydrographic/Topographic Center. The universal grids: Universal transverse mercator (UTM) and universal polar stereographic (UPS). Technical Report TM8358.2, 1989.

[9] M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In *Pervasive*, 2005.

[10] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *PETS*, 2009.

[11] Flickr. `http://www.flickr.com`.

[12] Java geodesy library. `http://www.gavaghan.org/blog/free-source-code/geodesy-library-vincentys-formula-java/`.

[13] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[14] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[15] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive*, 2009.

[16] Google Latitude. `http://www.google.com/latitude`.

[17] Groupon Local. `http://www.groupon.com`.

[18] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, 2003.

[19] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. Bayen, R. Herring, J. Herrera, M. Gruteser, M. Annavaram, and J. Ban. Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines. *TMC*, 11(5), 2012.

[20] Find my iPhone - iCloud. `http://www.icloud.com`.

[21] Instagram. `http://instagram.com`.

[22] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious tranfers efficiently. In *Advances in Cryptology – CRYPTO*, 2003.

[23] Java map projection library. `http://www.jhlabs.com/java/maps/proj/`.

[24] T. Jiang, H. Wang, and Y.-C. Hu. Preserving location privacy in wireless lans. In *MobiSys*, 2007.

[25] Jscience library. `http://jscience.org/`.

[26] T. Jung and X.-Y. Li. Search me if you can: Privacy-preserving location query service. *CoRR*, abs/1208.0107, 2012.

[27] C. Karney. Algorithms for geodesics. *Journal of Geodesy*, 87(1), Jan. 2013.

[28] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security (CANS)*, 2009.

[29] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2008.

[30] J. Krumm. Inference attacks on location tracks. In *Pervasive*, 2007.

[31] P. Lory. Enhancing the efficiency in privacy preserving learning of decision trees in partitioned databases. In *PSD*, 2012.

[32] J. Manweiler, R. Scudellari, Z. Cancio, and L. Cox. We saw each other on the subway: secure, anonymous proximity-based missed connections. In *HotMobile*, 2009.

[33] S. Mascetti, C. Bettini, and D. Freni. Longitude: Centralized privacy-preserving computation of users' proximity. In *SDM*, 2009.

[34] MIT iFIND Project. `http://web.mit.edu/newsoffice/2006/ifind.html`.

[35] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *ACM-SIAM Symposium On Discrete Algorithms (SODA)*, 2001.

[36] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

[37] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

[38] B. Pinkas, T. Schneider, N. Smart, and S. Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT*, volume 5912 of *LNCS*, 2009.

[39] P. Shankar, V. Ganapathy, and L. Iftode. Privately querying location-based services with sybilquery. In *UbiComp*, 2009.

[40] R. Shokri, G. Theodorakopoulos, J. Boudec, and J. Hubaux. Quantifying location privacy. In *SP*, 2011.

[41] R. Shokri, G. Theodorakopoulos, G. Danezis, J.-P. Hubaux, and J.-Y. Boudec. Quantifying location privacy: The case of sporadic location exposure. In *PETS*, 2011.

[42] R. Sinnott. Virtues of the haversine. *Sky and Telescope*, 68(2), 1984.

[43] W. Torge. *Geodesy*. De Gruyter, 2001.

[44] TripAdvisor. `http://www.tripadvisor.com`.

[45] T. Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 22(176), 1975.

[46] Apple Pays Out $946 in Locationgate Settlement. `http://www.wired.com/gadgetlab/2011/07/apple-locationgate-settlement/`.

[47] Stalkers Exploit Cellphone GPS. `http://online.wsj.com/article/SB10001424052748703467304575383522318244234.html`.

[48] A. Yao. How to generate and exchange secrets. In *FOCS*, 1986.

[49] Yelp. `http://www.yelp.com`.

[50] G. Zhong, I. Goldberg, and U. Hengartner. Louis, Lester and Pierre: Three protocols for location privacy. In *PET*, 2007.