Scalable Privacy-preserving Geo-distance Evaluation for Precision Agriculture IoT Systems

QIBEN YAN and JIANZHI LOU, Michigan State University, United States
MEHMET C. VURAN, University of Nebraska-Lincoln, United States
SUAT IRMAK, Pennsylvania State University, United States

Precision agriculture has become a promising paradigm to transform modern agriculture. The recent revolution in big data and Internet-of-Things (IoT) provides unprecedented benefits including optimizing yield, minimizing environmental impact, and reducing cost. However, the mass collection of farm data in IoT applications raises serious concerns about potential privacy leakage that may harm the farmers' welfare. In this work, we propose a novel scalable and private geo-distance evaluation system, called SPRIDE, to allow application servers to provide geographic-based services by computing the distances among sensors and farms privately. The servers determine the distances without learning any additional information about their locations. The key idea of SPRIDE is to perform efficient distance measurement and distance comparison on encrypted locations over a sphere by leveraging a homomorphic cryptosystem. To serve a large user base, we further propose SPRIDE+ with novel and practical performance enhancements based on pre-computation of cryptographic elements. Through extensive experiments using real-world datasets, we show SPRIDE+ achieves private distance evaluation on a large network of farms, attaining 3+ times runtime performance improvement over existing techniques. We further show SPRIDE+ can run on resource-constrained mobile devices, which offers a practical solution for privacy-preserving precision agriculture IoT applications.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; • **Computer systems organization** → *Embedded and cyber-physical systems;*

Additional Key Words and Phrases: Privacy-preserving data analysis, distance evaluation, precision agriculture, IoT

ACM Reference format:
Qiben Yan, Jianzhi Lou, Mehmet C. Vuran, and Suat Irmak. 2021. Scalable Privacy-preserving Geo-distance Evaluation for Precision Agriculture IoT Systems. *ACM Trans. Sen. Netw.* 17, 4, Article 38 (July 2021), 30 pages.
https://doi.org/10.1145/3463575

This article is extended from a conference paper published at IEEE CNS 2017.
This research was partially supported by the National Science Foundation under Grant Numbers CNS-1619285, CNS-1731833, CNS-1816938, CNS1950171, CNS1949753. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government or any of its agencies.
Authors' addresses: Q. Yan and J. Lou, Michigan State University, 28 S. Shaw Lane, Room 3115, East Lansing, Michigan, 48824; emails: {qyan, loujianz}@msu.edu; M. C. Vuran, University of Nebraska-Lincoln, 214 Schorr Center, Lincoln, Nebraska, 68588; email: mcv@unl.edu; S. Irmak, The Pennsylvania State University, Department of Agricultural and Biological Engineering, 105 Agricultural Engineering Building, Shortlidge Road, University Park, Pennsylvania, 16802; email: sfi5068@psu.edu.

1550-4859/2021/07-ART38 $15.00
https://doi.org/10.1145/3463575

1 INTRODUCTION

Precision agriculture has been envisioned as a new paradigm to revolutionize modern agriculture by collecting and processing data from farming machines and **Internet-of-Things (IoT)** devices in real-time, with the objective of optimizing seeding, irrigation, fertilization, harvesting, and other farming practices [17]. Farm fields usually have spatial variations of soils types, moisture levels, and nutrient availability. Precision agriculture IoT applications utilize spatio-temporal information to determine field variability, ensure optimal use of inputs, and maximize the outputs and profits from a farm [14]. By leveraging *in situ* sensing, IoT infrastructure, **geographical information systems (GIS)**, and **global positioning systems (GPS)**, farmers can more precisely determine their resource allocations for best outcomes. Moreover, due to the high correlation in some of the farm-related data over large distances (e.g., precipitation, soil type), information from multiple farms can be fused to make more informed decisions [5, 54].

Despite its obvious advantages, data collection in precision agriculture has raised serious privacy concerns [15]. More specifically, one of the major concerns of geospatial data collection is *location privacy* [42]. The farming applications require the field data to contain detailed spatial information, which is shared with service providers or third-party data analysts to carry out geospatial data analytics. Unfortunately, we have seen a growing number of server data breach incidents in recent years [21]. The disclosure of location data will be harmful, since the physical traces of the farms' sensors/machines can not only expose their physical locations, but may also lead to other undesirable consequences, such as: exposure of proprietary data analysis algorithm, planting strategies, or crop yield information. This is especially harmful to the farmers, because this information is directly related to their livelihoods. Furthermore, state- and country-wide implications exist with respect to food security [17]. Yet, we are unaware of any work that addresses privacy issues in agriculture, which is one of the impediments for implementing precision agriculture technologies by many farmers in their production fields.

Recently, several location privacy protection mechanisms have been developed in other fields [28]. One popular type of protection mechanisms is based on location obfuscation techniques that transform the true locations into an area or a perturbed location [42]. However, most existing spatial transformation techniques are subject to advanced location inference attacks [55]. More importantly, the perturbation of true locations may affect the utility of geographic services for agriculture. With the continuous improvement of GPS technology, we envision that precise locations will become more in demand [33], which makes the obfuscation of location data an unattractive solution for privacy protection. Thus, *a privacy-preserving mechanism to protect location information without sacrificing the high-precision agricultural operation is in urgent need.*

Even though the location precision is important, we note that the actual location information may not be as important as the *difference* between these locations, i.e., the distance [43]. More specifically, farming apps can perform data analysis based on the geo-distances among farms and sensors. Several studies have used distance between the farms (fields) and counties and interpolation techniques to develop spatial maps for various agricultural and climate variables, including evapotranspiration and water use efficiency [40], grain yield and yield production functions [23], and to evaluate the impact of the distance on ET gauge performance in estimating reference evapotranspiration [22]. Yet, the accuracy and quality of these services depend on the information gathered from many farmers. Sharing information is a major obstacle for farmers until they are assured that their shared data do not leak any private information. In this article, rather than protecting the location traces through obfuscation techniques, we investigate the problem of **cloud-based privacy-preserving continual distance evaluation without requiring the farms to disclose any exact location data to the cloud server**.

Note that the distance computation can be conducted between stationary or mobile entities such as: farms, sensors, and/or machines on the farm. There are four main challenges in the design of a privacy-preserving continual distance evaluation mechanism for precision agriculture: ***C1*:** the distance evaluation process should not leak any additional information about any entities' location traces to servers or third-party analysts; ***C2*:** the adversaries eavesdropping communications between the entities and server should not obtain the location data of individual entities; ***C3*:** since the entities may move at a high speed (e.g., farm vehicles), or can be resource constrained but have a large quantity (e.g., sensors), the distance evaluation mechanism should not only be accurate but also efficient, such that the server is capable of tracking the distances among multiple entities; ***C4*:** the computation result (i.e., the distance or Boolean proximity result) could only be accessed by authorized parties. Recently, researchers have investigated privacy-preserving distance calculation [35, 50] and proximity protocols [20, 60], with the goal of addressing ***C1*** and ***C2*** for mobile applications. However, they all focus on privacy-preserving computation of distance between two users: e.g., Alice and Bob, while the distance evaluation among a number of entities in a continuous fashion (***C3***) remains an open problem. Hence, the scalability is still an ongoing issue in this realm. Furthermore, no existing studies are designed for the situation where more than two parties are involved. When a cloud server (who is not the private key owner but needs to access the result) participates in the privacy-persevering computation, extra protections should be provided to prevent the other parties from accessing the result (***C4***).

In this article, we present a new solution called *SPRIDE*, a cloud-enabled Scalable and PRIvate continual geo-Distance Evaluation for precision agriculture IoT applications. SPRIDE utilizes homomorphic cryptosystem (i.e., the Paillier cryptosystem) to protect the locations of individual farms, sensors or other IoT entities. Note that homomorphic cryptosystem has been used in privacy-preserving distance calculation [50, 60] for *a pair of users*. SPRIDE aims to provide distance evaluation for *multiple farms* at a *cloud server* in a *continuous and scalable* manner. SPRIDE allows mobile entities to interact with servers without disclosing their exact locations, while the server privately computes the distances between entities in an efficient manner. Then, the server can provide spatio-temporal services (e.g., irrigation recommendations) based on the calculated distances, or send the relevant distance information to the authorized farmers either for their own record or to facilitate their localized computations. The interactions between farms and servers are designed to be efficient with minimum overhead.

Besides agricultural IoT applications, SPRIDE has the potential to enable other privacy-preserving applications, such as navigation apps, fitness tracking apps, and so on. *Private smart navigation* is one emerging application, where a system can complement or serve as an alternative to Google Maps for traffic-aware smart navigation while protecting users' location privacy. SPRIDE can be used as a key component in private smart navigation, where the mobile users refrain from the current practice of uploading their GPS location data to the server. Instead, the server uses SPRIDE to privately and continuously calculate the distance between mobile users, e.g., to monitor traffic conditions on the road. Similarly, the location tracking functionalities in fitness tracking apps [10] can also be implemented using SPRIDE to avoid the location privacy leakage.

In summary, this article endeavors to address geospatial data privacy issues in precision agriculture, and it makes the following contributions:

- We design and implement a cloud-based privacy-preserving geo-distance evaluation system, SPRIDE, for practical privacy-preserving distance tracking in precision farming IoT applications. SPRIDE utilizes cloud-based distance measurement based on a homomorphic cryptosystem to support large-scale distance evaluations for multiple farms.

- We propose performance enhancements to improve the distance evaluation performance of SPRIDE system using precomputation of cryptographic elements and optimization of homomorphic operations of the Paillier cryptosystem. The enhanced SPRIDE+ attains superior distance tracking performance, which can be scaled to accommodate a large number of farms. *For the first time, we demonstrate that the Paillier cryptosystem can be optimized to support practical large-scale computation with reasonable performance.*
- We conduct comprehensive experiments to evaluate SPRIDE+'s distance computation performance using both synthetic and real-world datasets, and we show the real-world applicability of the SPRIDE system for distance tracking.

The rest of the article is organized as follows: Section 2 introduces our motivation, the system model, and relevant background information. Section 3 presents the SPRIDE system design. The system evaluation is demonstrated in Section 4. We provide discussions in Section 5. After reviewing related work in Section 6, the article concludes with Section 7.

2 MOTIVATION, BACKGROUND, AND SYSTEM MODEL

In this section, we illustrate our motivation, system model for privacy-preserving distance evaluation mechanisms, and describe the background knowledge including Homomorphic Encryption and UTM projection for the representation of geo-locations.

2.1 Motivation

Even though agriculture, farming practices, and yield productivity are extremely private enterprise, recent advances in satellite, remote sensing (including use of drones), IoT systems, and GIS technologies are enabling the understanding, qualification, and evaluation of various farming practices as well as associated practices (irrigation water applications, crop type, nitrogen applications, crop water use (evapotranspiration), pesticides use, etc.). These new technologies may lead to the leakage of individual farm's productivity and income without the knowledge of farmer himself/herself. This has been a serious concern for farming communities. Farmers have numerous critical reasons as to why they do not wish to share these practices openly in public platforms. Some of these concerns revolve around privacy concerns of their individual and private income as a result of farming. For example, in a region or area where moratoriums for water use are in place by state government for regulation, farmers cannot pump more than a certain amount of water allocated for their operations. Farmers would like to keep this information away from public domain, especially in cases where regulations are involved. However, using precision agriculture technologies, it is possible to estimate this private data/information. Farmers also do not wish to share their grain yield at the harvest, because knowing how much grain yield a given farm obtains at the harvest, his/her gross income can be easily calculated.

These privacy concerns have actually increased in recent years with the advent of precision geolocation technologies when the precise location of the farmer or the farm is known. Knowing the exact location of a farm, others can estimate these private data and information with a good degree of accuracy that farmers do not wish to share. Besides the location of the farm itself, the exact locations of agricultural IoT devices can also disclose information. For example, by monitoring the location of a tractor or harvester, others could estimate the farmer's regular activities, and further infer safety- or financial-related private information. Thus, controllable data sharing that protects the privacy of the farmer (name, address, phone number, etc.) as well as his/her location are needed. Many farmers would be open to share data with researchers in platforms that do not reveal their identity and other private data and information [22].

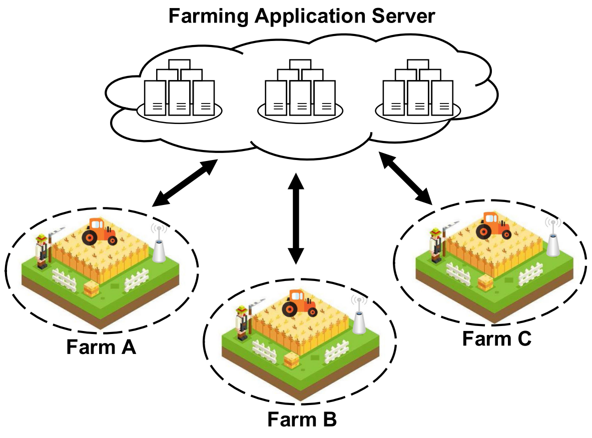


Fig. 1. Precision agriculture IoT system model.

In agricultural fields, there are numerous instrumentation (soil moisture, precipitation, air temperature, relative humidity, solar radiation, soil temperature, etc.) and heterogeneous IoT systems including large machinery (planter, harvester, cultivator, sprayer, etc.) and small mobile devices (phones, drones, etc.) that communicate with the base station (farmer) when in operation. It is very important that these instruments and machinery have effective and real-time communication for various decision-making throughout the growing season. For example, when irrigating, center pivot irrigation system, soil moisture sensors, and weather station need to have real-time communication to determine the proper irrigation timing and amount. When cultivating soil, cultivators need to communicate with tractors' control system to follow a certain path in the field using GIS for proper soil tillage practice. During harvesting, it is critical that the grain wagon and the combine (harvester) have real-time and uninterrupted communication for complete harvest and for creating continuous yield map of the grain yield. **This motivates us to design a highly performant privacy-preserving distance evaluation system for agricultural applications.**

2.2 System and Attack Model

We consider a generic precision agricultural system consisting of app cloud server and *mobile/stationary users (including farms, farmers, sensors, and machines),* as shown in Figure 1. Each user generates the geographical agricultural data to be shared with the app cloud. App cloud server performs agricultural data collection and analysis and provides geographic services to the farmers, while the farmers receive services without disclosing their true locations (i.e., the location coordinates) to the app cloud server. We consider the cloud server as *semi-honest (honest but curious),* a common model adopted by others [11, 56, 57], i.e., the cloud server is curious about *individual users*' whereabouts but follows the protocols honestly. The farmers are also considered as semi-honest, who follow the protocol but may attempt to infer other farms/devices' locations when they are not nearby. Also, there is no need to protect users' location privacy against nearby users (i.e., when the distance value is small). We assume the origin of the users' geospatial data is successfully hidden by anonymized protocols such as Tor networks [45] so IP geolocation mechanism could not determine the geolocation of hosts based on IP addresses of the packets.

The adversaries can eavesdrop on the communication between the farms and servers and try to obtain location information of users. Note that we do not consider malicious users faking their locations, which can be mitigated by tamper-resistant devices or unforgeable location tags [35].

Moreover, we do not consider colluding with users who share information with each other, or the denial of service attack [37], the defense of which is orthogonal to this work.

2.3 Homomorphic Encryption

SPRIDE utilizes **Homomorphic Encryption (HE)** to protect the location data. HE allows arbitrary computations (e.g., additions, multiplications, quadratic functions) on ciphertexts while preserving decryptability. The most powerful HE, namely, **Fully Homomorphic Encryption (FHE)**, supports an unlimited number of additions and multiplications, which has been utilized to support privacy-preserving data analytics [18]. However, FHE system is too computationally costly to be used in real-world applications [34]. **Partially Homomorphic Encryption (PHE)** supports limited operations on ciphertexts, which is more efficient and also more applicable to our scenario. One implemention of PHE, namely, the Paillier's system [36], is a simpler and widely used PHE that we will use in our mechanisms. Paillier system only involves one multiplication for each homomorphic addition and one exponentiation for each homomorphic multiplication. In the system, a user can encrypt the plaintext $m \in \mathcal{Z}_n$ with a public key $pk = (g, n)$ as:

$$c = E_{pk}(m) = g^m r^n \ mod \ n^2, \tag{1}$$

where n is the product of two large primes p and q, and $r \in \mathcal{Z}_n^*$ is selected randomly and privately by the user, and $\mathcal{Z}_n^*$ denotes the multiplicative group of invertible elements in $\mathcal{Z}_n$. The homomorphic property of Paillier system can be described as follows [36]:

$$\begin{aligned} D_{pk}(E_{pk}(m_1) \cdot E_{pk}(m_2)) &= m_1 + m_2, \\ D_{pk}\left(E_{pk}(m_1)^{m_2}\right) &= m_1 \cdot m_2. \end{aligned} \tag{2}$$

The random number r provides additional security without affecting the decryption result. Because of the randomness of r in every encryption, $E_{pk}(m_1) \cdot E_{pk}(m_2)$ and $E_{pk}(m_1 + m_2)$ are not necessarily numerically equal, but their decryption results are identical. Generally, the following equation can be used to describe the homomorphic property:

$$\begin{aligned} E_{pk}(m_1) \cdot E_{pk}(m_2) &= E_{pk}(m_1 + m_2), \\ E_{pk}(m_1)^{m_2} &= E_{pk}(m_1 \cdot m_2). \end{aligned} \tag{3}$$

For simplicity, we will use Equation (3) instead of Equation (2), when we introduce the algorithms of SPRIDE.

2.4 UTM Projection

UTM (Universal Transverse Mercator) is a projected coordinate system, which is a type of plane rectangular coordinate system. UTM serves as an alternative coordinate system to the popular geographic coordinate (i.e., Latitude and Longitude) system as used by GPS navigation systems. The GPS coordinate system uses curved grids to accommodate the curved surface of the earth. The geographic latitude, longitude coordinates are measured in degrees, minutes, and seconds of arc. These geographic coordinates can be converted into plane coordinates by means of map projections, which essentially transforms the earth's curved surface into a flat two-dimensional surface, creating a projected UTM coordinate system. UTM coordinate system provides a referencing frame to define the positions of objects. Since the UTM system greatly simplifies the distance calculation between two objects on earth, SPRIDE is specially designed to work with UTM-formatted data.

With UTM, the Earth is divided into 60 UTM zones, each being a six-degree band of longitude. To minimize the scale distortion within each UTM zone, each of the 60 UTM zones gets projected onto a plane separately. The meridian at the center of each UTM zone is called the ***central meridian (CM).*** Each zone is divided into horizontal grids eight degrees of latitude wide, which are labeled with grid letters ranging from C to X from south to north. The position of a point in the rectangular coordinate system is defined by the distance from x and y axis, which uses a measurement unit such as meters.

A point can be represented as (z, x, y), where z specifies the zone including the zone number and grid letter, x denotes the easting coordinate, and y denotes the northing coordinate. The value of x falls in the range of $[166,000, 834,000]$, and the value of y falls into $[0, 9,999,999]$ [19]. For instance, the USA contiguous states span across 10 UTM zones, and y ranges in $[2,700,000, 5,500,000]$. To avoid negative values, on the east-west position, the CM is assigned a value of $500,000$ meters, and points lying to the east of CM have $x > 500,000$, while points lying to the west of CM have $x < 500,000$. In the northern hemisphere, y specifies the distance of a point to the equator, while in the southern hemisphere, y is equal to $10,000,000m$ minus its distance from the equator. One can then use the *Pythagorean Theorem* to calculate the distance between any two points in the UTM form [19].

3 SPRIDE DESIGN

This section illustrates the design of SPRIDE, which is a continual distance evaluation system for precision agriculture. We first design a privacy-preserving distance measurement scheme for SPRIDE, which allows the cloud to compute the distance between any farms or sensors accurately. Then, we enhance the performance of SPRIDE to scale to a large user base. SPRIDE also offers a privacy-preserving distance comparison scheme to compare the distance to a threshold to achieve better privacy. In a nutshell, SPRIDE is supported by a cloud infrastructure with a cloud server serving a large network of farms, which allows cross-farm data analytics.

3.1 Location Data Preprocessing

SPRIDE operates over UTM format location data. Therefore, after receiving the (latitude, longitude) location data from GPS module, the local data server will first convert the location data into UTM format using the formulas of Karney [25]. Every user converts the location data as an offline computation before interacting with the app cloud server. The UTM data has the format of (z, x, y). In the case that two locations (e.g., $L_1 : (z, x_1, y_1)$ and $L_2 : (z, x_2, y_2)$) reside in the same zone,[1] the distance can be computed easily (i.e., $d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$). In the following sections, we also deal with the case when two objects reside in different zones.

3.2 Cloud-based Privacy-preserving Distance Measurement

SPRIDE allows the server to compute the distance between any pair of users requesting the service, while the users only submit encrypted locations to the cloud. In case there are a large number of farms, pairwise distance computations will incur considerable computational costs. Since distance evaluation for users that are too far away is generally not helpful, the server divides a large service area into several *service sections*. The users will disclose the service section information to the server, and pairwise distance evaluation will be conducted inside each section. Note that the section size should be large enough to constrain privacy leakage and also small enough to limit the computational costs. The service sections for different applications can overlap, as one user may request multiple services. For example, a farm can participate in both the local information sharing network and statewide information sharing network. The service area segmentation for farming applications is orthogonal to this work. In our evaluation, we configure a statewide service section for ease of illustration.

[1]For ease of presentation, zone is used to denote UTM zone hereafter.

ALGORITHM 1: High-level Structure of Distance Measurement

```
Setup: Each mobile device obtains their locations in UTM format $(z_i, x_i, y_i)$, $i \in [1, N]$;
The app cloud server receives $z_i = \{zone\_number_i, is\_north_i\}$, $i \in [1, N]$, and the distance between
  user $U_i : (z_i, x_i, y_i)$ and $U_j : (z_j, x_j, y_j)$ will be computed;
if $z_i = z_j$ then
    Apply Algorithm 2 to compute the distance;
else
    if $zone\_number_i = zone\_number_j$, $is\_north_i \neq is\_north_j$ then
        Apply Algorithm 2 to compute the distance;
    else
        if $|zone\_number_i - zone\_number_j| = 1$ then
            Apply Algorithm 3 to project the location;
            Apply Algorithm 2 to compute $x_i \cdot x_j + y_i \cdot y_j$ (or $x_i \cdot x_j - y_i \cdot y_j$);
            Compute the distance based on Equation (4) or (5);
        else
            return;
        end
    end
end
```

To start the process, each user will send the service section information to the cloud, based on which the cloud identifies each user's zone, including the zone number and grid letter. SPRIDE uses zone information to identify three different types of ***positional relationships (PRs)*** when we use different metrics to calculate the geo-distance between users: (1) ***Type I PR:*** two locations are within the same hemisphere in the same zone; (2) ***Type II PR:*** two locations are within different hemispheres in the same zone; (3) ***Type III PR:*** two locations are in the neighboring zones.

Note that when the distance between two objects enlarges, error in distance calculation also accumulates [19], as confirmed in Section 4.2. However, farming applications mostly consider to associate data with farms that are not too distant from each other (but also not nearby). Thus, SPRIDE practically avoids distant users, thereby ensuring the accuracy of distance measurement, as shown in Section 4.2. The complete high level algorithm is shown in **Algorithm 1**, where $zone\_number_i$ denotes the zone number of user U_i, and is_north_i is an indicator denoting whether U_i is in the north hemisphere. **Algorithm 1** covers three cases corresponding to three different types of PRs. The distance between two nodes with either Type I PR (see lines 4–5) or Type II PR (see lines 7–8) can be computed using **Algorithm 2**, while Type III PR nodes can be measured using a mapping algorithm (i.e., **Algorithm 3**) before applying **Algorithm 2** (see lines 10–12).

Case I: Same Zone Same Hemisphere: Now, we present the algorithm for Type I PR, corresponding to the most common case when two locations reside in the same hemisphere in the same zone. We use Paillier cryptosystem to achieve the privacy-preserving distance measurement between any two users. One major concern with an asymmetric cryptosystem, such as Paillier, is its expensive computational cost. As our distance measurement mechanism may need to serve a large number of farms, we strive to improve the mechanism's computational efficiency.

Specifically, to compute the distance of U_i and U_j with Type I PR, or d_{ij}, the cloud computes the following squared distance value using Pythagorean Theorem:

$$d_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 = x_i^2 + x_j^2 + y_i^2 + y_j^2 - 2x_ix_j - 2y_iy_j. \tag{4}$$

Algorithm 2 implements a secure distance measurement mechanism. In the algorithm, U_i computes three encryptions and one decryption, the cloud computes one encryption, and U_j computes one encryption, one decryption, and two exponentiations. Specifically, in the *Preparation* phase, U_i encrypts every location component. In the *First Round*, the cloud generates a random number δ (used to mask the distance) and interacts with each user U_j to get $\mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$ leveraging the homomorphic property. In the *Second Round*, the cloud interacts with the other user U_i to decrypt the masked distance, remove the mask, and derive the distance. In this algorithm, all location components exchanged between users and the cloud are encrypted, and the cloud server cannot decrypt any of these location components. In the end, the cloud server learns the distance without learning any location components of each user.

Note that the outer layer encryption in line 4 of Algorithm 2, which is used to prevent U_i's eavesdropping, can be replaced by symmetric crypto systems. However, considering the simplicity of key management, we still use the Paillier encryption. Moreover, the length of the ciphertext $\mathsf{Enc}_{pk_i}(\delta)$ may exceed n, so the server only encrypts the lowest 128 bits using pk_j. Specifically, the server splits $\mathsf{Enc}_{pk_i}(\delta)$ into two components: c_{low} and c_{high}, where c_{low} contains the lowest 128 bits and c_{high} contains the remaining bits. The server delivers c_{high} and $\mathsf{Enc}_{pk_j}(c_{low})$ to U_j, and then U_j decrypts $\mathsf{Enc}_{pk_j}(c_{low})$ and combines these two components.

ALGORITHM 2: Privacy-preserving Distance Measurement

```
1 Setup: $N$ users have their own locations in UTM format $(z_i, x_i, y_i)$, $i \in [1, N]$. If the user is in southern
    hemisphere, update $y_i = 10,000,000 - y_i$;
2 Each user $U_i$ is assigned a pair of private key and public key of Paillier's cryptosystem $(sk_i, pk_i)$. $\mathsf{Enc}_{pk}$
    denotes the Paillier encryption;
3 Preparation: $U_i$ encrypts $x_i^2 + y_i^2$, $x_i$, and $y_i$ using $pk_i$. Then $U_i$ uploads $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2)$, $\mathsf{Enc}_{pk_i}(x_i)$,
    and $\mathsf{Enc}_{pk_i}(y_i)$ to cloud server;
4 First Round: If cloud server initiates the process of computing the distance of user $U_i$ and $U_j$, it
    generates a large random integer $\delta$ and encrypts it using both $pk_i$ and $pk_j$, i.e., $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$.
    Cloud server sends $pk_i$, $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2)$, $\mathsf{Enc}_{pk_i}(x_i)$, $\mathsf{Enc}_{pk_i}(y_i)$, and $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ to $U_j$;
5 $U_j$ decrypts $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ to get $\mathsf{Enc}_{pk_i}(\delta)$,and encrypts $x_j^2 + y_j^2$ using $pk_i$, and then computes
    $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2) \cdot \mathsf{Enc}_{pk_i}(\delta) \cdot \mathsf{Enc}_{pk_i}(x_j^2 + y_j^2) \cdot \mathsf{Enc}_{pk_i}(x_i)^{-2x_j} \cdot \mathsf{Enc}_{pk_i}(y_i)^{\pm 2y_j} =$
    $\mathsf{Enc}_{pk_i}(\delta + (x_i - x_j)^2 + (y_i \pm y_j)^2) = \mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$. $U_j$ uploads $\mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$ to cloud server;
6 Second Round: cloud server sends $\mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$ to $U_i$ for decryption; $U_i$ decrypts $\delta + dist_{ij}^2$, and
    sends it back to cloud server;
7 The cloud computes $dist_{ij}$.
```

Case II: Same Zone Different Hemispheres: In case two mobile devices are located in the same zone, but in different hemispheres, i.e., one location resides in southern hemisphere and the other in northern hemisphere, the squared distance value is very similar to that of the previous case, except that the vertical distance now becomes $(y_i + y_j)$ (the northing coordinate y_i of user U_i in the south has already been converted into $10,000,000 - y_i$), therefore, the squared distance can be depicted as follows:

$$d_{ij}^2 = (x_i - x_j)^2 + (y_i + y_j)^2 = x_i^2 + x_j^2 + y_i^2 + y_j^2 - 2x_i x_j + 2y_i y_i. \tag{5}$$

Similarly, as the cloud obtains $(x_i - x_j)^2 + (y_i \pm y_j)^2$ from the *Second Round*, the distance d_{ij} can be computed accordingly.

Case III: Neighboring Zones: There is one rare case when two farms are located in the neighboring zones. We need to conduct distance measurement across zones, which becomes more challenging. To the best of our knowledge, *we are the first to consider privacy-preserving distance measurement for nodes across two UTM zones.*

Our basic idea is to project one location in one zone into the other zone before computing the distance. When the server discovers that two users requesting the service are in the neighboring zones, the server will notify one user to perform a zone projection. The neighboring zone projection algorithm is described in **Algorithm 3**. In line 3, we use *haversine formula* [53] to calculate the spherical distance between the zone meridian and the node. The haversine formula is written as:

$$d = 2 \cdot R \cdot arcsin\left(\sqrt{sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + cos(\phi_1)cos(\phi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right), \quad (6)$$

where (λ_1, ϕ_1) is the longitude and latitude of node 1, (λ_2, ϕ_2) is the longitude and latitude of node 2, and R is the earth radius ($R = 6,371$ km).

After the zone projection, two users are located inside the same zone. Essentially, we reduce the Case III problem into Case I and Case II problems, which can be solved accordingly. It is possible that U_s in zone S is involved in the distance calculation with another node located in zone $S - 1$, S, or $S + 1$. Thus, U_s can prepare and upload three versions of its locations, including the original one and two projected ones, onto the server. When these two nodes are from different UTM zones, Algorithm 3 projects node U_j into U_i's zone.

ALGORITHM 3: Neighboring Zone Projection

```
1 User $U_s$ in location $(z_s, x_s, y_s)$ in zone $S$ projects into zone $M$, where $S$ and $M$ are the zone numbers,
    and $|S - M| = 1$. The longitude and latitude of $U_s$ are $(\lambda_s, \phi_s)$, where $\lambda_s$, $\phi_s$ are in radians;
2 Compute the longitude of the central meridian of zone $M$ as: $\lambda_c = (-183 + M * 6)$ degrees;
3 The spherical distance $d$ from $U_s$ to the central meridian of zone $M$ with coordinates $(\lambda_c, \phi_s)$ can be
    computed using haversine formula in Equation (6);
4 if $S > M$ then
5     The projected location of $U_s$ is: $(z_m, 500,000 + d, y_s)$;
6 else
7     The projected location of $U_s$ is: $(z_m, 500,000 - d, y_s)$;
8 end
```

The cloud-based distance measurement can effectively compute the distance between any pair of users on a map. As a result, the cloud is capable of computing multiple distance measurements from one end (e.g., U_i) to multiple users. In this case, the cloud will send the *encrypted location components* (i.e., encrypted $x_i^2 + y_i^2$, x_i, and y_i) from U_i to all the other users. All the other users receiving multiple components can perform the homomorphic operations using their own locations. Note that the cloud does not have the private keys of the ciphertexts, thus is unable to decrypt the *encrypted location components*. We provide a detailed security analysis in the following section.

3.3 Security Analysis of Privacy-preserving Distance Measurement

In this section, we perform security analysis of the aforementioned privacy-preserving distance measurement scheme. In **Algorithm 2**, only the cloud server learns the distance, while other parties would not. The only ***unencrypted shared information (USI)*** is the section information, which is used to identify PR and user pair (U_i, U_j) for distance measurement. The disclosed section information is a tradeoff for computational efficiency and privacy. Intuitively, if the section size is large, then the exact location of a user remains well protected. As discussed previously, the section size is an important factor influencing the privacy of SPRIDE. The investigation of a proper section size to balance efficiency and privacy is on our agenda.

With every location component encrypted, Algorithm 2 does not disclose any location information to anyone (including the eavesdroppers and any participating parties), except that only the cloud server learns the distance. It is worth noting that U_i learns $\delta + dist_{ij}^2$. Yet, δ, as a large random number, can completely mask the distance value. Also, the value of δ is encrypted by the public keys of both U_i and U_j. The outer layer encryption (Enc_{pk_j}) prevents U_i from eavesdropping, and the inner layer encryption (Enc_{pk_i}) enables the homomorphic operations without leaking δ to U_j. As such, neither U_i nor U_j could obtain the plaintext δ. Moreover, in line 5 of **Algorithm 2**, U_j receives the encrypted location of U_i, which cannot be decrypted without U_i's private key. However, although U_i owns his/her own private key, he/she only receives the $\delta + dist_{ij}^2$. Therefore, SPRIDE protects any user location from other users and external adversaries, even when the adversaries have abundant background knowledge.

In summary, we demonstrate that the cloud server and external adversaries are unable to extract sensitive location data. However, due to the nature of SPRIDE system, the distance information is presented to the cloud server. Now, we evaluate the privacy concerns brought by the distance disclosure. By knowing the distance between any pair of users, it will be difficult to guess the exact location of individual users. For example, if U_i and U_j are separated by a distance of γ, without knowing the location of U_i or U_j, then it is almost impossible to guess the exact location of the other counterpart.

Location Triangulation: One potential security weakness in distance measurement is that multiple colluding parties can do location triangulation to pinpoint a specific user's location. However, in SPRIDE system, only the cloud server knows the distance between any two users. Without the knowledge of distances, no user or attackers would be able to launch location triangulation to pinpoint a specific user.

In case the location of one end of distance measurement is known, the cloud might be able to pinpoint the location of the other end. For example, if one end of distance measurement is a riverbank, then by learning the distance between a user and the riverbank, a determined cloud can pinpoint the location of this user by examining the circular area with the specific distance to the riverbank on a map. Even if the locations of both ends are unknown, the cloud may be able to match the measured distances to a real map to identify users' locations. As a result, we propose a more secure *distance comparison* mechanism (suitable for certain applications such as precipitation prediction as described below) to further strengthen location privacy especially when the distance measurement is performed against a **point of interest (PoI)** with a guessable location.

3.4 Cloud-based Privacy-preserving Distance Comparison

To alleviate the aforementioned potential security issue, we propose a new cloud-based privacy-preserving distance comparison mechanism as an integral part of SPRIDE. The privacy-preserving distance comparison compares the distance between a pair of users to a *distance comparison threshold.* The algorithm returns whether the distance is less than, equal to, or greater than a threshold value τ, which is determined by the app server. To avoid significant privacy leakage, τ should not be set as a small value. The distance comparison mechanism also builds upon Paillier homomorphic encryption, and we incorporate a random number δ to introduce randomness for protecting the distance value against both the cloud and the users. While the privacy-preserving distance comparison is not suitable for services that require exact distance values, it can support numerous farming applications, such as building a privacy-preserving model for predicting the precipitation within a circle centered around a farm.

The complete algorithm is shown in **Algorithm 4**, including *Preparation* step, two rounds of communications between cloud and users. The *Preparation* step and *First Round* are the same as that of **Algorithm 2**. In the *Second Round*, the cloud transmits an encrypted randomized distance and threshold to U_i. After that, U_i computes the *randomized distance value*: $dist_{ij}^2 + \delta$ and performs the comparison. In this algorithm, U_i is oblivious to the distance value, as the computed distance is randomized by cloud server with δ. The cloud server also gains no knowledge of the distance, since the result the server receives is only a binary output. Compared to the distance measurement algorithm, this algorithm only allows the cloud to learn the binary comparison results, which effectively counteracts the location triangulation attack. However, "malicious" cloud server may conduct multiple distance comparisons to confine the target user into a small range of possible locations. To counteract such threats, the user can set up a policy to only allow server to perform distance comparison once per one particular location and reject extra requests. We skip the detailed security analysis for this algorithm, which is similar to the security analysis in Section 3.3.

To facilitate different types of location-based services, the cloud server can select the distance comparison threshold strategically to suit special needs of services. For distance comparison of a user and a fixed PoI, the threshold determines the range of the user w.r.t. the fixed PoI. In this case, the threshold should be large enough to limit the privacy leakage from the comparison results. As a result, the system sets a lower bound for the threshold, e.g., as 10 miles, which means a user can be identified as residing within the 10-mile area without leaking his/her exact location.

It is worth noting that precise distance measurement and distance comparison support different types of precision agriculture applications. Therefore, SPRIDE implements both systems for different purposes.

ALGORITHM 4: Privacy-preserving Distance comparison

```
1 Setup: $N$ users have their own locations in UTM format $(z_i, x_i, y_i)$, $i \in [1, N]$. If the user is in southern
    hemisphere, update $y_i = 10{,}000{,}000 - y_i$;
2 Each user $U_i$ is assigned a pair of private key and public key of Paillier's cryptosystem $(sk_i, pk_i)$. $\mathsf{Enc}_{pk}$
    denotes the Paillier encryption. The distance comparison threshold value is $\tau$;
3 Preparation: $U_i$ encrypts $x_i^2 + y_i^2$, $x_i$, and $y_i$ using $pk_i$, and uploads $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2)$, $\mathsf{Enc}_{pk_i}(x_i)$ and
    $\mathsf{Enc}_{pk_i}(y_i)$ to cloud server;
4 First Round: If cloud server initiates the process of comparing the distance of $U_i$ and $U_j$ with the
    threshold value, cloud server generates a large random integer $\delta$ and encrypts it using both $pk_i$ and
    $pk_j$, i.e., $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ . Cloud server sends $pk_i$, $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2)$, $\mathsf{Enc}_{pk_i}(x_i)$, $\mathsf{Enc}_{pk_i}(y_i)$, and
    $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ to $U_j$;
5 $U_j$ decrypts $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ to get $\mathsf{Enc}_{pk_i}(\delta)$, and encrypts $x_j^2 + y_j^2$ using $pk_i$, and then computes
    $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2) \cdot \mathsf{Enc}_{pk_i}(\delta) \cdot \mathsf{Enc}_{pk_i}(x_j^2 + y_j^2) \cdot \mathsf{Enc}_{pk_i}(x_i)^{-2x_j} \cdot \mathsf{Enc}_{pk_i}(y_i)^{-2y_j} =$
    $\mathsf{Enc}_{pk_i}((x_i - x_j)^2 + (y_i - y_j)^2 + \delta) = \mathsf{Enc}_{pk_i}(dist_{ij}^2 + \delta)$. $U_j$ uploads $\mathsf{Enc}_{pk_i}(dist_{ij}^2 + \delta)$ to cloud server;
6 Second Round: Cloud server sends $\mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$ and randomized threshold value $(\tau^2 + \delta)$ to $U_i$;
7 $U_i$ then decrypts $dist_{ij}^2 + \delta$, compares it with $\tau^2 + \delta$. Then, we have:
    $(dist_{ij})^2 + \delta < (or\ \geq) \tau^2 + \delta \Leftrightarrow dist_{ij} < (or\ \geq) \tau$, and $U_i$ submits the binary result (0 denotes $< \tau$, and
    1 denotes $\geq \tau$) to the cloud.
```

Table 1. The Total Runtime Performance of **Algorithm 2** W.R.T. Different Key Sizes (1,024 and 2,048 Key Sizes Are Our Main Focus)

Key size (bits)	Runtime (ms)
256	1.561
512	7.547
1,024	**48.613**
2,048	**346.079**

Table 2. The Decomposition of Operation Time (ms) for SPRIDE (**Algorithm 2**) with Different Key Sizes

Key Size (bits)	256	512	**1,024**	**2,048**
Encryption	0.903	4.498	**29.169**	**208.148**
Decryption	0.523	2.738	**18.520**	**134.934**
Homomorphic exponentiation	0.130	0.304	**0.904**	**2.878**
Homomorphic multiplication	0.001	0.004	**0.014**	**0.111**
Other computations	0.003	0.003	**0.006**	**0.008**

3.5 Runtime Performance of Privacy-preserving Distance Evaluation

We implement all the algorithms in Java and run experiments to evaluate the runtime performance of SPRIDE. All experiments are performed on a desktop computer with AMD R7-2700 processor and 32 GB DDR4 RAM. We run each measurement 1,000 times, and count the average of total runtime.

At present, 2,048-bit modulus is considered secure for a Paillier cryptosystem, while 1,024-bit Paillier cryptosystem can be used to protect short-lived messages, i.e., the messages that expire within several days or weeks [4]. Table 1 shows the runtime performance of both distance measurement/comparison algorithms. We can see that when the key size increases to 2,048 bits, the performance degrades dramatically. To put it into context, the computation time for computing the distance of 1,000 pairs of users reaches 346 seconds or 6 minutes with 2,048-bit key size. We further decompose **Algorithm 2** to measure the runtime performance of each operation with different key sizes, the result of which is shown in Table 2. It shows that encryption and decryption operations take the majority of time. Although most of the cryptographic operations are performed locally, the cloud server still has one encryption (Step 4 in **Algorithm 2**) for each request, which costs a considerable processing delay when dealing with concurrent, large-quantity requests. To provide a more scalable solution, we strive to optimize the runtime performance via precomputation.

3.6 SPRIDE+: Performance Enhancement of SPRIDE based on Precomputation of Paillier Components

In this section, we propose a new system, SPRIDE+, to significantly improve the runtime performance. The basic idea is to precompute components in Paillier cryptosystem to save computational time [36]. Let us first scrutinize the decryption process of Paillier cryptosystem:

$$m = L(c^{\lambda} \mod n^2) \cdot \mu \mod n,$$

Table 3. The Decomposition of Operation Time (ms) for SPRIDE+ with Different Key Sizes

Key Size (bits)	256	512	**1,024**	**2,048**
Encryption	0.189	0.547	**1.819**	**6.729**
Decryption	0.245	1.377	**9.041**	**66.728**
Homomorphic exponentiation	0.118	0.310	**0.869**	**2.834**
Homomorphic multiplication	0.001	0.004	**0.013**	**0.103**
Other computations	0.002	0.003	**0.005**	**0.007**

where $\mu = L(g^{\lambda} \mod n^2)^{-1} \mod n$, $\lambda = LCM(p-1, q-1)$, and L is an operation defined by $L(x) = \frac{x-1}{n}$, LCM is least common multiple. As component μ is irrelevant to the ciphertext, we can precompute μ during the key generation. After precomputing μ, the decryption time with 1,024-bit key in **Algorithm 2** decreases from 18.52 ms (in Table 2) to 9.04 ms, as shown in Table 3.

Similarly, in Paillier's encryption, $c = g^m \cdot r^n \mod n^2$, r^n can also be precomputed. Before every encryption, the users can prepare a random number r in advance and compute r^n. When our input m is a seven-digit number, g^m requires much less computation than r^n, given that n has 1,024 or 2,048 bits. Thus, the precomputation of r^n significantly reduces the encryption time. Table 3 shows that the encryption process now takes 1.82 ms instead of 29.17 ms for 1,024-bit key size.

However, precomputing r^n has a major caveat for end-users. Recall in **Algorithm 2**, U_i performs three encryptions (i.e., $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2)$, $\mathsf{Enc}_{pk_i}(x_i)$, $\mathsf{Enc}_{pk_i}(y_i)$), the cloud server performs one encryption (i.e., $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$), and U_j performs one encryption $\mathsf{Enc}_{pk_i}(x_j^2 + y_j^2)$ using $pk_i = (g_i, n_i)$. While U_i owns the key-pair (sk_i, pk_i), and the cloud gets pk_i in preparation step, precomputing r^{n_i} is not a problem for them; however, to allow U_j to precompute r^{n_i}, every U_j has to know every pk_i in the system in advance, i.e., each end-user has to store and update all the public keys from other users. This requires a complicated (public) key management system to be in place and takes extensive computing and storage resources from end-users.

To bypass this hurdle, we propose to let U_j remove r^n entirely instead of precomputing them. In Paillier cryptosystem, the following homomorphic property holds [36]: $\mathsf{Dec}(\mathsf{Enc}(m_1) \cdot g^{m_2}) = m_1 + m_2$. Therefore, to compute $m_1 + m_2$, we can multiply g^{m_2} without fully encrypting m_2, thereby eliminating the need of r^n. As a result, during the encryptions at U_j, we can remove r^n completely to save computational time without affecting the correctness of distance computation. Therefore, in SPRIDE+, the Step 5 of **Algorithm 2** can be rewritten as follows (other steps remain):

```
5: $U_j$ decrypts $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ to get $\mathsf{Enc}_{pk_i}(\delta)$, and computes $g^{x_j^2+y_j^2} \mod n^2$,
   and then compute $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2) \cdot \mathsf{Enc}_{pk_i}(\delta) \cdot (g^{x_j^2+y_j^2} \mod n^2) \cdot \mathsf{Enc}_{pk_i}(x_i)^{-2x_j} \cdot \mathsf{Enc}_{pk_i}(y_i)^{-2y_j} = \mathsf{Enc}_{pk_i}(\delta + (x_i - x_j)^2 + (y_i - y_j)^2) = \mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$. $U_j$ sends the
   result the cloud server.
```

Security Analysis of SPRIDE+: Here, we prove SPRIDE+ is secure by showing that SPRIDE+ and SPRIDE are equally secure.

THEOREM 1. *SPRIDE+ is as secure as SPRIDE and/or the Paillier cryptosystem.*

Table 4. The Total Runtime Performance of SPRIDE+

Key Size (bits)	Runtime (ms)
256	0.555
512	2.243
1,024	**11.746**
2,048	**76.401**

Proof. SPRIDE is secured by Paillier cryptosystem, the security of which is provided by the hardness of factorizing a large integer n^2. To prove SPRIDE+ is as secure as SPRIDE, we show that attacking SPRIDE+ is as difficult as attacking Pailliar cryptosystem.

In Paillier cryptosystem, the homomorphic multiplication result of two encrypted messages is $\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) = (g^{m_1} \cdot r_1^n \mod n^2) \cdot (g^{m_2} \cdot r_2^n \mod n^2) = g^{m_1+m_2} \cdot (r_1 r_2)^n \mod n^2$, where r_1, r_2 are randomly chosen to encrypt m_1, m_2, respectively. The homomorphic multiplication result is equivalent to the result when encrypting $m_1 + m_2$ directly, using random number $r = r_1 r_2$, which we call *ciphertext I*. Meanwhile, if $(g^{m_2} \mod n^2)$ is multiplied with $\mathsf{Enc}(m_1)$, the result is $(g^{m_1} \cdot r_1^n \mod n^2) \cdot (g^{m_2} \mod n^2) = g^{m_1+m_2} \cdot r_1^n \mod n^2$, which is equivalent to the case that encrypts $m_1 + m_2$ directly using r_1, and we call the result as *ciphertext II*. The *ciphertext I* and *ciphertext II* are equally secure, since the security for both of them can be mapped to the security of Paillier encryption. Therefore, **we prove that multiplying $g^{m_2} \mod n^2$ with an existing encrypted component does not degrade the security of the homomorphic encryption.**

Assume an attacker eavesdrops the ciphertext, $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2) \cdot \mathsf{Enc}_{pk_i}(\delta) \cdot (g^{x_j^2+y_j^2} \mod n^2) \cdot \mathsf{Enc}_{pk_i}(x_i)^{-2x_j} \cdot \mathsf{Enc}_{pk_i}(y_i)^{-2y_j}$, which is sent by U_j in Step 5. Note that there are encrypted components (e.g., $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2)$) in the multiplication. As a result, attacking the entire ciphertext directly is as hard as attacking a Paillier encryption.

To conclude, the attacker cannot decrypt the ciphertext without breaking the Pailliar encryption. Therefore, SPRIDE+ is secure. □

As a result, SPRIDE+ with precomputation is both correct and secure, and we can use it to significantly speed up the runtime of distance computation. From Table 4, we can see that the runtime performance improves significantly from 346 ms to 76 ms (i.e., 4.5× improvement) per distance measurement for 2,048-bit key size. With such a performance improvement, *SPRIDE+* is able to perform distance computation with 2,048-bit key with a reasonable speed.

3.7 Continual Distance Tracking

When the users are moving, distance measurement among users needs to be carried out continuously to allow the cloud server to keep track of users' moving trajectory. In cases when the users are moving at a steady speed relative to each other, we have the opportunity to further reduce the distance computation costs by leveraging distance prediction. In this section, we propose a distance prediction algorithm that aims at reducing the distance measurement frequencies by capturing moving statistics. In a nutshell, SPRIDE enters the *prediction mode* based on historic distance measurements, during which a series of distance measurements are predicted without any computations/communications, thereby conserving computational resources and communication overhead. The complete distance prediction algorithm is presented in **Algorithm 5**.

The number of distance measurement samples to determine whether to enter prediction mode is denoted as *prediction range* r. For instance, r continual samples are taken with a sampling rate of 1 *sample per* 5 *seconds*, i.e., $d_1, d_2, \ldots, d_r$. Note that, since the location is inaccessible to the

ALGORITHM 5: Distance Prediction Algorithm

```
1  Set *prediction range r*, *prediction STD threshold λ*, and *error bound err*;
2  repeat
3      Take a continuous set of distance measurements with cardinality equal to the prediction range $r$,
         including $d_1, d_2, \ldots, d_r$;
4      Compute moving speed $speed_i = |d_{i+1} - d_i|, i \in [1, r-1]$;
5      if $std(speed_i) < \lambda$ then
6      repeat
7          Set $r$ to $2r$ except the initial repeat round;
8          Enter prediction mode;
9          Predict a set of distance measurements with cardinality equal to the prediction range $r$,
             including $d'_{r+1}, d'_{r+2}, \ldots, d'_{2r}$, where $d'_j = d'_{j-1} + mean(\{speed_i\}), j \in [r+1, 2r], i \in [1, r-1]$,
             $d'_r = d_r$;
10         Perform a distance measurement $d_{2r}$ for distance verification;
11     until $|d_{2r} - d'_{2r}| > err$;
12     Exit prediction mode;
13 until;
```

cloud server, the moving direction/duration of the user cannot be inferred. However, the moving speed can still be estimated using the difference of two measurements. As a result, the entry to the prediction mode can be determined by the standard deviation of moving speed values measured inside prediction range. Specifically, when evaluating the distance between users (or the distance between a user and a fixed PoI), the moving (relative) speed can be calculated using the difference between consecutive distance measurements, i.e., $speed_i = |d_{i+1} - d_i|, i \in [1, r-1]$. If the standard deviation $std(\{speed_i\})$ is less than a threshold, called *prediction STD threshold* λ, then SPRIDE enters the prediction mode and predicts a prediction range of distance measurements using an estimated speed of $mean(\{speed_i\})$, generating $d'_{r+1}, d'_{r+2}, \ldots, d'_{2r}$. Immediately after the distance prediction, we start a distance verification by comparing the final predicted distance d'_{2r} to the real distance measurement d_{2r}. If the error is less than an *error bound err*, then we continue the distance prediction; otherwise, we exit the prediction and restart the process.

Using the distance prediction algorithm, not only does one save computation/communication time spent on distance evaluation, but one can also allow the farming apps to sample locations less frequently on resource-constrained sensors while preserving the app functionality. Note that three parameters are involved in the prediction including: *prediction range r*, *prediction STD threshold* λ, and *error bound err*, and we evaluate the performance improvement w.r.t. these parameters in Section 4.6.

To further improve the distance prediction performance, we incorporate a new parameter, namely, *prediction level*, which determines the cardinality of the prediction set, i.e., with a higher level, more distance predictions will be generated. The new algorithm is termed as *level-deepening distance prediction algorithm*, as shown in **Algorithm 6**. For a user (or pair), if we continuously enter the prediction mode, meaning the moving speed and moving direction are stable, then the prediction level will be extended. The *prediction level increase* parameter is denoted as Δl, which denotes the number of new predictions in case the prediction level deepens. Therefore, if two nodes' positions are relatively stable to each other, the prediction level will keep increasing, producing more distance predictions. By setting $\Delta l = 0$, the algorithm returns to the original distance prediction algorithm. The level-deepening algorithm further increases the number of distance predictions via analyzing the moving statistics, thereby reducing the computation costs even further.

ALGORITHM 6: Level-deepening Distance Prediction Algorithm

```
1  Set prediction range $r$, prediction STD threshold $\lambda$, error bound $err$, and prediction level increase $\Delta l$;
2  repeat
3      Take a continuous set of distance measurements with cardinality equal to the prediction range $r$,
         including $d_1, d_2, \ldots, d_r$;
4      Compute moving speed $speed_i = |d_{i+1} - d_i|, i \in [1, r-1]$;
5      Reset $l$ to 0;
6      if $std(speed_i) < \lambda$ then
7      repeat
8          Set $r$ to $2r + l$ except the initial repeat round;
9          Enter prediction mode;
10         $l = l + \Delta l$ except the initial repeat round;
11         Predict a set of distance measurements with cardinality equal to the prediction range $r + l$,
             including $d'_{r+1}, d'_{r+2}, \ldots, d'_{2r+l}$;
12         Perform a distance measurement $d_{2r+l}$ for distance verification;
13     until $|d_{2r+l} - d'_{2r+l}| > err$;
14     Exit prediction mode;
15     Reset $l$ to 0;
16 until;
```

4 EVALUATION

In this section, we evaluate the distance evaluation performance of the SPRIDE+ system. Specifically, we first focus on the accuracy of distance measurement based on UTM location coordinates, and then we evaluate the scalability of SPRIDE+ system. Note that the users denote farms, farmers, sensors, machines, or other IoT entities, depending on specific IoT applications. Then, we evaluate the application generality of the SPRIDE+ system over different regions by comparing the accuracy over different latitude bands. Moreover, we evaluate the performance improvement brought by the distance prediction algorithm. Finally, overhead evaluation is presented to show the applicability of SPRIDE+ system in real-world applications. We expect the SPRIDE+ system to run on agricultural IoT devices, which has similar computation power as commodity PC and mobile phones. Therefore, our experiments are performed on regular PCs or mobile devices. Specifically, all the algorithms are implemented in Java and run with **Java Virtual Machine (JVM)** on a desktop computer with AMD R7 and 32 GB memory, or Android Nexus 5 phones (only for experiments in Section 4.7).

4.1 Datasets

Due to the lack of real-world datasets of farming applications, we produce synthetic location data at the state of Nebraska, which is a typical agricultural producing state [46] located in the Great Plains. As shown in Figure 2, Nebraska is a state that covers three UTM Zones, and it allows us to synthesize data with both Type I and Type III PRs within the state. The major area of the state is located in the UTM Zone 14, while there is also a rectangular area in Zone 13 and a tiny area in Zone 15. In our experiments, we mainly focus on Zone 13 and 14. We randomly select locations with latitude and longitude inside the state, based on which we created three lists with each list containing 500 locations: $List_1$ contains 500 locations in Zone 14; $List_2$ contains another 500 locations in the same area; $List_3$ contains locations in Zone 13. Figure 2 shows all the locations on the map.

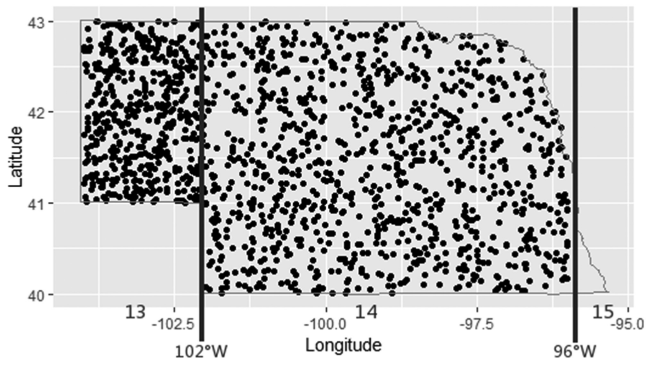
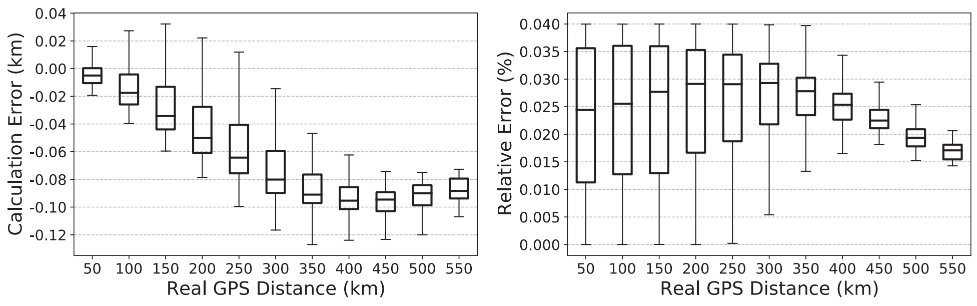


Fig. 2. Synthetic locations inside the state of Nebraska.



(a) Distance calculation error for Type I and Type II PR (b) Relative error for Type I and Type II PR

Fig. 3. Distance measurement errors for Type I and Type II PR.

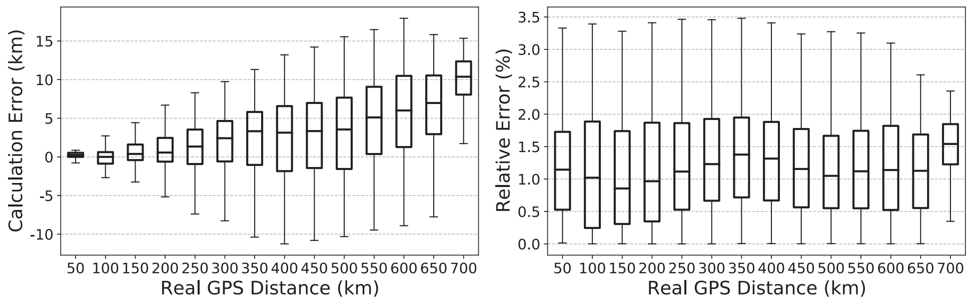
We also use a real-world dataset, called Geolife dataset [59], from mobile applications to evaluate SPRIDE+, and we believe the adopted geolocation dataset should be representative for farming apps as well. Geolife data was collected from 182 users over a period of three years, which is sufficient for the evaluation of our continual distance tracking algorithm. It recorded a wide range of users' outdoor movements, represented by a series of tuples containing latitude, longitude, and timestamp. The trajectories were updated every $1-5$ seconds.

4.2 Evaluation of Distance Measurement Accuracy

In this section, we evaluate the distance measurement accuracy of SPRIDE+. The ground-truth distance is computed using the Vicinity's formulae [49]. We define the *distance calculation error* as the difference between the measured distance and ground-truth distance, and *relative error* as ratio of the difference and the ground-truth distance. We compute and represent the distance calculation error and relative error as box plots, and we keep the sign of the distance calculation error value to show its variance.

When computing the distance for user pairs in the same UTM zone (Type I or Type II PR), we randomly pick one location from $List_1$ and the other location from $List_2$. For computing the distance in neighboring zones (Type III PR), we pick one from $List_1$ and the other from $List_3$. We pick 10,000 pairs for Type I, II, and Type III, respectively, and run the distance measurement to record the distance errors of SPRIDE+.

For the locations in the same UTM zone (Type I, II PR), as Figure 3(a) shows, the distance calculation error increases with a larger distance. The relative error, however, is more stable. For the



(a) Distance calculation error for Type III PR (b) Relative error for Type III PR

Fig. 4. Distance measurement errors for Type III PR.

Table 5. Confusion Matrices for Distance Comparison of Type I, II PR

τ=10 km		SPRIDE Result	
		$\leq \tau$	$> \tau$
Real Distance	$\leq \tau$	3	0
	$> \tau$	0	1,997

τ=100 km		SPRIDE Result	
		$\leq \tau$	$> \tau$
Real Distance	$\leq \tau$	318	0
	$> \tau$	1	1,681

Table 6. Confusion Matrices for Distance Comparison of Type III PR

τ=10 km		SPRIDE Result	
		$\leq \tau$	$> \tau$
Real Distance	$\leq \tau$	0	0
	$> \tau$	0	2,000

τ=100 km		SPRIDE Result	
		$\leq \tau$	$> \tau$
Real Distance	$\leq \tau$	40	0
	$> \tau$	0	1,960

location pairs in the same zone, Figure 3(b) shows that the relative error is between 0 and 0.40%. To put it into context, for two points separated by 100 km, the average distance calculation error is only around 20 meters, demonstrating the accuracy of SPRIDE+.

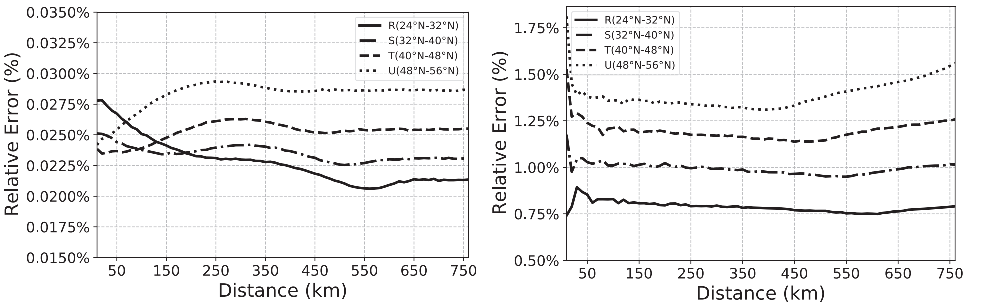
As for locations in neighboring UTM zones (Type III PR), we also notice distance error increases with a larger distance. But we observe a larger variation in the distance error compared with Type I, II PR, as shown in Figure 4(a). Figure 4(b) shows that the relative error is also considerably larger than that of Type I, II PR. This is brought by the inaccuracy of neighboring zone projection. For example, for two locations with 100 km distance, the maximum distance error is <3 km.

4.3 Evaluation of Distance Comparison Accuracy

In this section, we evaluate the distance comparison accuracy of SPRIDE+. The real-world agricultural applications require different encompassing areas of farms. We use different distance comparison threshold τ to evaluate how SPRIDE+ classifies the locations. We set threshold $\tau = 10$ and 100 km, respectively. For each threshold, we randomly select 2,000 pairs of locations and use **Algorithm 4** to determine whether $dist_{ij} \leq \tau$ or not. Meanwhile, we compute the ground-truth distance using the GPS data to validate the actual classification, the results of which are displayed in Table 5 and 6. The results indicate very rare false classifications for all Type I, II, and III PRs, and demonstrate the accuracy of SPRIDE+'s distance comparison.

4.4 Application Generality for Different Regions

The UTM projection maps points of the earth surface onto a 2-dimensional plane, which inevitably introduces errors. Here, we run experiments to show that SPRIDE+ works on different latitudes,



(a) Relative error over different bands for Type I, II PR (b) Relative error over different bands for Type III PR

Fig. 5. Relative error over different bands.

especially the moderate-latitude, food-supplying regions, which are particularly important for agriculture, such as the Great Plains in the United States.

As demonstrated earlier, the UTM projection segments each zone into 8-degree latitude bands, while each band is represented by an alphabetical letter. Particularly, Bands R(24°N-32°N), S(32°N-40°N), T(40°N-48°N), and U(48°N-56°N) cover most of the Great Plains regions [52]. As a result, we evaluate the accuracy of SPRIDE+ system over these bands. For each band, we repeatedly select 10,000 pairs of random points from the same UTM zone (Type I or Type II PR) and from neighboring UTM zones (Type III PR), respectively. Figure 5 compares the average relative errors over different bands. The results show that the locations in higher latitudes present higher errors after the UTM projection. However, the relative errors are still acceptable for agricultural applications (i.e., less than 0.03% for Type I and II PR, and less than 1.8% for Type III PR) even on high-latitude bands. For two farms separated by 50 km distance, the upper bound of projection error will be approximately 15 meters (Type I, II) and 900 meters (Type III), respectively. The result illustrates the generality of SPRIDE+ system across different regions.

4.5 Scalability Performance with Multiple Users

Distance Measurement: We implement the SPRIDE+ system with precomputation and evaluate the performance of distance evaluation with multiple users using the Geolife dataset. Recall that the user who initiates the measurement (U_i) is responsible for three encryptions and one decryption, while the app cloud will perform one encryption ($\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$), and the other user ($U_j$) will perform a decryption, a partial encryption ($g^{x_j^2+y_j^2}$) and homomorphic operations. In our experiment, we assume the user and app cloud have the same hardware configurations (i.e., a desktop computer with AMD R7 and 32 GB memory).

We randomly select k pairs of users to compute the total time consumption for different k values, the results of which are displayed in Figure 6. With 2,048-bit key size, the distance measurement with $1,000$ pairs of users takes less than 80 seconds. For farms with fixed locations, SPRIDE+ is efficient enough to support the distance computation or comparison services for a large number of farms. For instance, SPRIDE+ can compute the distance of 1,000,000 pairs of users within one single day. For each pair of users such as (moving) farming machines, SPRIDE+ takes 80 ms to compute their distance. This indicates that SPRIDE+ can support real-time distance computation. As for the network communication, both **Algorithms 2 and 4** require two round-trips between app cloud and users, the overhead of which is displayed in Section 4.7.

Distance Comparison: We then implement and evaluate distance comparison of SPRIDE+ in **Algorithm 4**. We set the distance comparison threshold τ as 10 km. The total computation time is

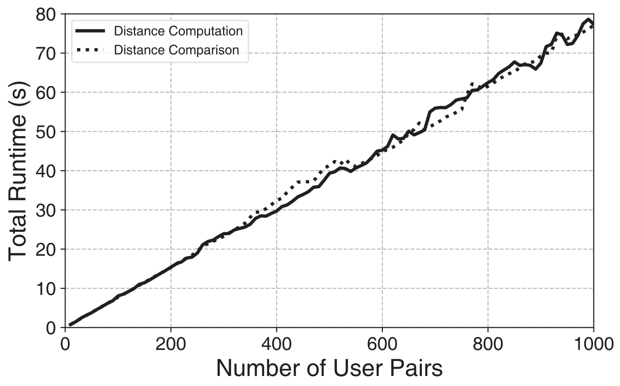


Fig. 6. Total runtime for up to 1,000 pairs of farms.

presented in Figure 6, which resembles the result of distance measurement. Therefore, we conclude that SPRIDE+ is a scalable distance evaluation system that can accommodate a large number of farms.

Parallel Queuing SPRIDE+: So far, we run SPRIDE+'s distance computation sequentially, i.e., when a node (the cloud server or a user) is executing a task (e.g., encryption or homomorphic operation), it occupies the computational resource exclusively. Apparently, the result serves as a runtime upper bound, while in fact all the nodes can run certain tasks in parallel to get a more accurate estimation of SPRIDE+'s runtime performance. For example, $\mathsf{Enc}(x_i^2 + y_i^2)$ and $\mathsf{Enc}(\delta)$ can be processed parallelly, while $\mathsf{Enc}(x_i^2 + y_i^2) \cdot \mathsf{Enc}(\delta)$ should wait until the prerequisites complete.

We implement a parallel queuing simulation to simulate the parallel version of SPRIDE+ distance computation. Here, every node has its FIFO queue to store the tasks to process, with which the tasks are scheduled by *"first come, first serve"* without concurrency. All prerequisite tasks are executed in parallel, which are supplied to queue once completed. We do not consider communication delays in this experiment.

We use the queuing system to simulate two SPRIDE+ applications: *App A*, in which a user requests distances between itself and all the other nodes; *App B*, in which multiple users request distance computations, and each of them wants to calculate the distance with another node. For *App A*, SPRIDE+ can initiate distance computations: either (1) using the key from the user who requests the service, or (2) using the key from the other user. In case (1), the node that requests the service will play the role of U_i in every computation, and its queue will be piled with all the final decryption tasks for computing $dist^2 + \delta$, making itself a bottleneck. In case (2), the node will play the role of U_j, except one specific decryption (to get $\mathsf{Enc}_{pk_i}(\delta)$) in each computation, all the remaining decryptions can be processed in parallel, reducing the total runtime dramatically. For *App B*, the task allocation is more evenly distributed. On average, each node only serves as U_i and U_j once, respectively.

We use the queuing system to simulate two cases of *App A* and *App B*, the results of which are displayed in Figure 7. For *App A*, both cases (1) and (2) take about 38 seconds to compute 1,000 user pairs, which demonstrates a dramatic runtime improvement compared with the upper-bound. For *App B*, Figure 7 illustrates a promising result, in which 1,000 distance computation can be completed in about 4 seconds. The simulation results indicate that *App A* is able to serve more users compared with the original SPRIDE+ system, while *App B* can achieve a more efficient real-time computation (i.e., 4 ms for each distance computation) for multiple moving users.

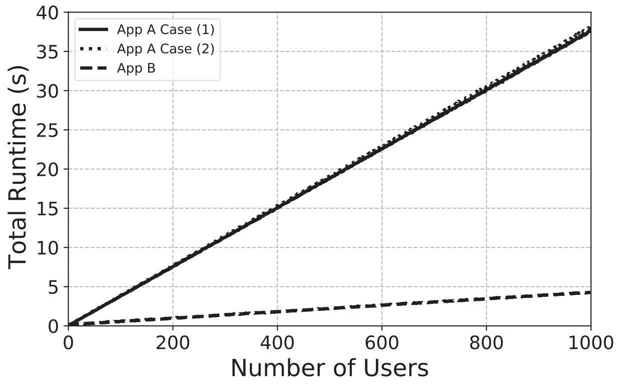


Fig. 7. Runtime of different Apps with parallel queuing SPRIDE+.

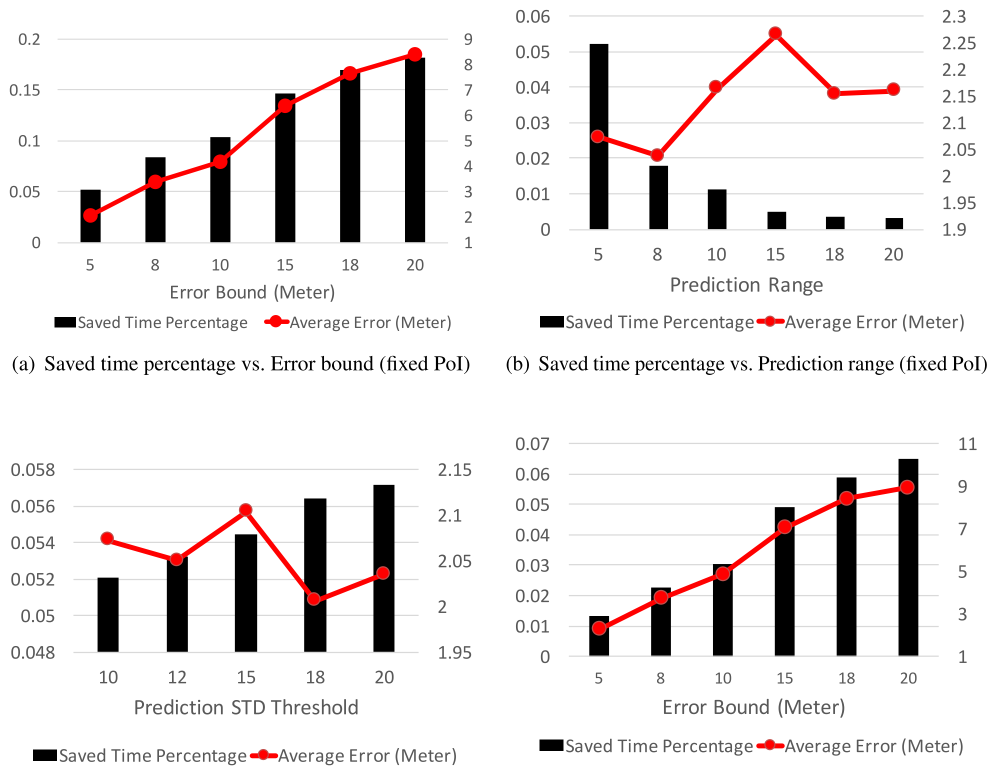
Table 7. Time Cost Comparison for 1,000 User Pairs

Methods	Parties	Result Type	Receiver	Time (s)
PLQP (1,024 bit) [29]	2	Real/Boolean	Querier	741
PP-UTM (1,024 bit) [50]	2	Real	Querier	33
PP-UTM (2,048 bit) [50]	2	Real	Querier	231
SPRIDE+ (1,024 bit)	3	Real/Boolean	Server	**12**
SPRIDE+ (2,048 bit)	3	Real/Boolean	Server	**76**
SPRIDE+ (2,048 bit, *App A* Simulation)	3	Real/Boolean	Server	**38**
SPRIDE+ (2,048 bit, *App B* Simulation)	3	Real/Boolean	Server	**4**

Comparing with Other Methods: For comparison, we implement two existing privacy-preserving distance measurement methods: PLQP [29] and PP-UTM [50], both of which measure distance between a single pair of users using homomorphic encryption, and PP-UTM further computes distance over UTM projection. As shown in Table 7, PLQP has different query levels and returns real (distance) or Boolean (less than a threshold or not) values, while PP-UTM can only return the distances. Furthermore, PLQP and PP-UTM only supports two parties in their protocols, and the querier (private key owner) can access the result. However, SPRIDE+ allows the server to take control of the computation, and it can prevent the privacy key owner (U_i), as well as other parties (e.g., U_j and adversaries), from accessing the distance results. Table 7 also shows that SPRIDE+ achieves 62 times improvement over PLQP, and 3 times improvement over PP-UTM in time consumption with the same modulus size, which highlights the scalability of SPRIDE+ system, making it *particularly suitable for agricultural applications processing geolocation data continuously from a large number of farms.*

4.6 Performance Improvement of Distance Prediction Algorithm

Next, we evaluate the performance improvement by distance prediction algorithm in terms of *saved time percentage*, which is defined as the ratio of saved computation time to total computation time. Since the performance may vary for different users, we chose 200 trajectories from 200 users of GeoLife dataset, each of which has around 1,000–3,000 timestamps, to evaluate the average performance improvement of distance prediction algorithm. The default settings of the parameters, including *Error Bound, Prediction Range, Prediction STD Threshold, Prediction Level Increase,* are shown in Table 8. For the following experiments, we vary the value of one parameter



(a) Saved time percentage vs. Error bound (fixed PoI) (b) Saved time percentage vs. Prediction range (fixed PoI)

(c) Saved time percentage vs. Prediction STD threshold (fixed PoI) (d) Saved time percentage vs. Error bound (users)

Fig. 8. Impact of parameters on performance improvement using distance prediction: (a)(b)(c) Impact of parameters on computation time savings for fixed PoI distance evaluation; (d) Impact of error bound on computation time savings for multi-user distance evaluation.

Table 8. Default Parameter Settings

Error Bound	5
Prediction Range	5
Prediction STD Threshold	10
Prediction Level Increase	0

while retaining default values for other parameters if not mentioned. The average performance is reported in Figure 8.

Impacts of Error Bound: In Figure 8(a), the saved time percentage increases with larger error bound for distance evaluation between a user and a fixed PoI. The result shows that if we allow a large distance prediction error, then we will spend less time on distance computation. For instance, when the error bound is set as 10 meters, 10% of total computation time could be saved. Meanwhile, the average prediction error rises with increasing error bound. As a result, the app server can pick an error bound to strike the balance between the computation time savings and prediction errors. For distance evaluation between users, the performance improvement trend is similar as shown in Figure 8(d), but the saved time percentage is significantly less than that of the fixed PoI case. The reduced time savings are caused by the elevated difficulty in predicting distance between two

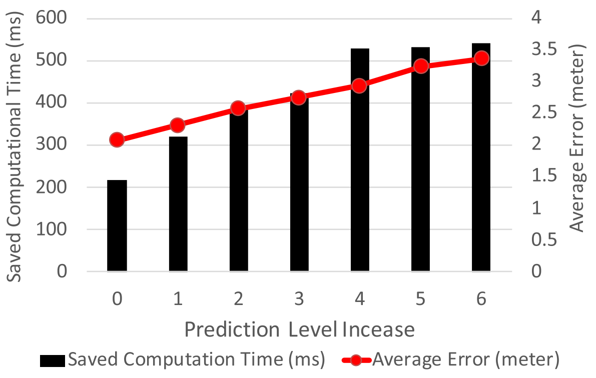


Fig. 9. Performance improvement w.r.t. prediction level increase.

users. In fact, only when two users are moving with a steady speed on the same direction can we enter prediction mode to predict their future distance, which accounts for less time savings. Nevertheless, the time saving percentage can still reach beyond 5% of total computation time when the error bound is set to 15 meters.

Impacts of Prediction Range: The relationship between saved time percentage and prediction range is shown in Figure 8(b), which shows that time savings decline with an increasing prediction range. This is due to the increasing difficulty in entering prediction mode. If we take more points for prediction mode evaluation, then we will have a higher chance to encounter the moving statistic change, which translates into a lower chance of entering prediction mode or reduced time savings. In addition, the average prediction error, which fluctuates slightly around 2.1 meters, is unaffected by different prediction ranges. Here, we only show the result of fixed PoI case, as the result of multiple user case is similar.

Impacts of Prediction STD Threshold: Higher prediction STD threshold brings more time savings, as shown in Figure 8(c). The reason is obvious, as it becomes easier to enter the prediction mode with a higher prediction STD threshold. Similar to prediction range, the prediction STD threshold does not affect the average prediction error.

Impacts of Prediction Level Increase: Figure 9 shows that saved computational time increases with a larger value of *prediction level increase*, because we make more predictions and thus conduct less computations using level-deepening distance prediction algorithm. However, there is also a slight increase in average prediction errors using the level-deepening algorithm. Overall, the level-deepening distance prediction further reduces the computation costs of SPRIDE+ system.

4.7 Overhead Evaluation on Mobile Device

Since mobile devices have limited computational power, we run the SPRIDE+ system on mobile devices to evaluate the computation costs in real devices (Android Nexus 5). Each mobile device participating in the distance evaluation will perform Paillier encryption, decryption, and homomorphic operation. With SPRIDE+ and 1,024-bit key size, each distance evaluation takes U_i around 10 ms for Paillier encryption and decryption, while it takes U_j around 3 ms for homomorphic operation, which is acceptable. The communication overhead is listed in Table 9, the total of which is less than 15 KB. Therefore, the SPRIDE+ system introduces low overhead and can be applied in real-world privacy-preserving agricultural applications on mobile and IoT devices.

Table 9. Communication Overhead of SPRIDE+ (in Bytes)

$U_i \rightarrow$ Cloud	Cloud $\rightarrow U_i$	$U_j \rightarrow$ Cloud	Cloud $\rightarrow U_j$
1,792	5,376	5,376	1,792

5 DISCUSSION

In this section, we discuss the factors that impact the accuracy of SPRIDE system by going over all the processing steps of SPRIDE. We also illustrate the feasibility of transferring SPRIDE system from UTM projection to other systems.

5.1 The Impact of Location Discretization

The preparation step requires each user to convert their locations into the UTM format, which contains the easting and northing coordinates. To satisfy the requirement of Homomorphic Encryption, the value of easting and northing should be discretized into integers. This discretization step will cause minor errors. Since each point will be approximated into the closest integer lattice on the grid, the maximum drift for a single node is approximately $\sqrt{0.5^2 + 0.5^2} \approx 0.71$ meter. In the worst case (when two points are drifted along the line that connects them but in opposite directions), the maximum error is expected to be $2 \times 0.71 = 1.42$ meters. When the distance between two locations is a few kilometers, this minor approximation loss is negligible.

5.2 The Impact of Homomorphic Encryption

As mentioned in Section 2, we utilize the Paillier system to implement the Homomorphic Encryption, which works on multiplicative groups $\mathcal{Z}_n$ whose elements are all integers. While the intermediate results are encrypted, the addition and multiplication operations are preserved by the homomorphic property. As a result, after completing all the operations, the decrypted result should be exactly the same as the one after directly applying the *Pythagorean Theorem* to the plaintext locations.

However, two exceptions exist: (1) the input is invalid, and (2) the result $(\delta + dist_{ij}^2)$ falls outside the range of the multiplicative group $\mathcal{Z}_n$. The discretization step prevents exception (1) from happening, while exception (2) can also be avoided when the key size is large enough. When using keys in the length of 1,024 bits or above, the range of $\mathcal{Z}_n$ (which will be approximately $2^{1,024}$ or higher) can certainly cover all the legitimate results. To conclude, with a proper design, Homomorphic Encryption will not affect the accuracy of the distance evaluation.

5.3 The Impact of Coordinate Systems

Measuring the distance under a certain coordinate system inevitably introduces errors, since all of them, such as UTM projection, **Earth-Centered Earth-Fixed (ECEF)** coordinates (which maps the earth surface to three-dimensional Euclidean space) or Latitude-Longitude system, are abstracted from the real world. For example, UTM flattens a piece of earth surface into a two-dimensional plane during the projection, which introduces some errors. The error is more significant at a higher latitude or when two points are in different UTM zones, as illustrated in Section 4.

When using Latitude-Longitude system (haversine formula) or ECEF system to compute the Great Circle Distance, computation errors also exist due to the simplification of the earth shape from the actual one (oblate spheroid [51]) to the idealized one (spheroid). Furthermore, the evaluation from Reference [50] shows high errors of haversine formula when the two locations are close to each other.

ALGORITHM 7: ECEF-based SPRIDE

```
1 Setup: $N$ users have their own locations in ECEF format $(x_i, y_i, z_i)$, $i \in [1, N]$, where $(x_i, y_i, z_i)$
    represents the coordinate of user $U_i$ in 3-dimensional Euclidean space; Each user $U_i$ is assigned a pair
    of private key and public key of Paillier's cryptosystem $(sk_i, pk_i)$. $\mathsf{Enc}_{pk}$ denotes the Paillier
    encryption;
2 Preparation: $U_i$ encrypts $x_i^2 + y_i^2 + z_i^2$, $x_i$, $y_i$, and $z_i$ using $pk_i$. Then $U_i$ uploads $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2 + z_i^2)$,
    $\mathsf{Enc}_{pk_i}(x_i)$, $\mathsf{Enc}_{pk_i}(y_i)$, and $\mathsf{Enc}_{pk_i}(z_i)$ to cloud server;
3 First Round: If cloud server initiates the process of computing the distance of user $U_i$ and $U_j$, it
    generates a large random integer $\delta$ and encrypts it using both $pk_i$ and $pk_j$, i.e., $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$.
    Cloud server sends $pk_i$, $\mathsf{Enc}_{pk_i}(x_i^2 + y_i^2 + z_i^2 + \delta)$, $\mathsf{Enc}_{pk_i}(x_i)$, $\mathsf{Enc}_{pk_i}(y_i)$, $\mathsf{Enc}_{pk_i}(z_i)$ and
    $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ to $U_j$;
4 $U_j$ decrypts $\mathsf{Enc}_{pk_j}(\mathsf{Enc}_{pk_i}(\delta))$ to get $\mathsf{Enc}_{pk_i}(\delta)$, and encrypts $x_j^2 + y_j^2 + z_j^2$ using $pk_i$, and then computes
    $\mathsf{Enc}_{pk_i}(x_i^2+y_i^2+z_i^2)\cdot\mathsf{Enc}_{pk_i}(\delta)\cdot\mathsf{Enc}_{pk_i}(x_j^2+y_j^2+z_j^2)\cdot\mathsf{Enc}_{pk_i}(x_i)^{-2x_j}\cdot\mathsf{Enc}_{pk_i}(y_i)^{-2y_j}\cdot\mathsf{Enc}_{pk_i}(z_i)^{-2z_j} =$
    $\mathsf{Enc}_{pk_i}(\delta + (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2) = \mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$. $U_j$ uploads $\mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$ to
    cloud server;
5 Second Round: cloud server sends $\mathsf{Enc}_{pk_i}(\delta + dist_{ij}^2)$ to $U_i$ for decryption; $U_i$ decrypts $\delta + dist_{ij}^2$, and
    sends it back to cloud server;
6 The cloud computes $dist_{ij}$.
```

5.4 Transferring SPRIDE to Other Coordinate Systems

Transferring SPRIDE into other coordinate systems will be beneficial if the location data from the agricultural systems cannot be easily converted into the UTM format. Here, we present **Algorithm** 7, which implements SPRIDE on ECEF coordinate system, to transfer SPRIDE to support other coordinate systems. The major difference is the three-dimensional representation of a location and the removal of UTM zone and hemisphere information. However, the involvement of the third dimension requires more encryption operations, which may affect the runtime performance and scalability of the distance evaluation algorithms. This idea can be applied to transfer SPRIDE into Latitude-Longitude system, such as GPS systems, and then use haversine formula to compute the distance. The performance enhancement and comparison of different SPRIDE implementations will be our future work.

6 RELATED WORK

In this section, we discuss prior research efforts in location privacy and privacy-preserving data analysis in light of this research.

6.1 Location Privacy

A rich set of existing work has been developed to address the problem of location privacy in location-based services. In this section, we discuss additional relevant work that has not been covered. Location obfuscation is a prevalent non-cryptographic technique to protect location privacy. It can be done entirely on the user's side by perturbing the location coordinates [2, 42]. Several location obfuscation techniques add noise to the users' location coordinates [2], hide the real location among a set of dummy locations [26], or use cloaking algorithm to conceal real location [9]. Recently, privacy-preserving proximity test has been studied. InnerCircle [20] is a proximity protocol based on homomorphic cryptosystem. Freni et al. [16] propose to provide a coarse granularity of location data to protect location privacy, while Mascetti et al. [32] extend this work to a centralized scenario. Different from the previous work mostly focusing on proximity testing, SPRIDE is a practical system that tracks geo-distances continuously on earth for precision agriculture.

Other privacy-preserving approaches generate fake locations to hide users' true locations by constructing fake trips with more probable paths traveled by drivers [27]. Chen et al. [8] presented a privacy-preserving map generation using crowd-sourced location data, which lets users upload unorganized sparse location points to avoid privacy leakage. Recently, Sedenka et al. [50] homomorphically computed the distance using UTM projection, **ECEF (Earth-Centered Earth-Fixed)** coordinates, and haversine formula, which is the most relevant work to ours. However, they only consider a one-time distance computation for two users, while we focus on cloud-based, continuous distance tracking for multiple users in a scalable manner. With performance enhancement, our system consumes less resources. Differential privacy has also been used to protect location release from inference attack. Using Markov model, Xiao et al. [55] proposed δ-location set–based differential privacy to account for temporal correlations to hide true locations among a set of indistinguishable locations. However, the perturbed location leads to a degraded utility of location-based service. Recently, Li et al. [29] proposed privacy-preserving location query services using homomorphic encryption and CP-ABE to provide fine-grained access control. They mainly focused on access control for location query service, whereas we strive to develop scalable distance evaluation for location-based services in precision agriculture. Finally, we contrast our work with anonymization. Anonymization removes identifiers of individuals in the data and publishes only the resulting sanitized dataset. While anonymization preserves utility, it fails to provide adequate privacy protection. Researchers have shown that anonymized traces can be easily de-anonymized [31].

In summary, the existing work cannot provide a scalable privacy-preserving distance evaluation when the app server and a large number of participants are involved. In particular, the distance evaluation based on perturbed data will not be accurate enough for some applications demanding precise locations.

6.2 Privacy-preserving Data Analytics

The big data analytics requires massive data collection that presents various privacy concerns. The data owners, such as farmers, are reluctant to share their data due to the privacy and confidentiality concerns. As a result, privacy-preserving data analysis has become a popular research area. **Secure multi-party computation (SMC)** has been used to protect the intermediate steps of the computation when multiple parties perform collaborative data analysis on their proprietary inputs. For example, SMC has been used for privately developing machine learning models based on decision trees [30], Naive Bayes classifiers [48], linear regression functions [12], k-means clustering [24], and association rules [47]. SMC-based techniques generally impose substantial performance overheads, which are hard to deploy in a large-scale network.

Differential privacy [13] is a popular approach for privacy-preserving data analysis. It has been used to provide privacy-preserving linear and logistic regression [6, 58], principle component analysis [7], support vector machines [38], and continuous data processing [39]. Abadi et al. [1] further demonstrate a differentially privacy **stochastic gradient descent (SGD)** algorithm to train the deep neural network. Differential privacy has been used in real Apple devices, since iOS 10 [3], however, its implementation detail is obscure and may still lead to potential privacy leakage [44]. Moreover, developing models with differential privacy guarantee is difficult, because the sensitivity of models that determine the data perturbation is unknown for most data analysis and machine learning approaches. Recently, Shokri et al. [41] propose a privacy-preserving deep learning model that shares model parameters between a local device and a parameter server without sharing users' sensitive input data. However, the share of model parameter may still lead to potential information leakage.

Unlike the previous approaches, SPRIDE focuses on providing a continual and scalable distance evaluation system, which protects privacy of geographical data in agricultural fields. We rely on an efficient homomorphic encryption scheme and enhance its performance to make it suitable for real deployment in a large-scale network.

7 CONCLUSION

In this article, we designed a scalable and private continual geo-distance evaluation system, SPRIDE, to tackle the location privacy issue for the first time in precision agriculture IoT applications. SPRIDE leverages homomorphic cryptosystem to perform distance evaluation on user-encrypted location data in UTM format. During the distance evaluation, the geolocations are protected against other farms, IoT app cloud, and external adversaries. We further enhanced the performance of the distance evaluation using the precomputation technique, and proposed SPRIDE+, which achieves at least 3× runtime performance improvement over existing techniques. We showed through experiments with both synthetic and real-world datasets that SPRIDE+ can provide accurate distance measurements and can process a large number of farms' encrypted locations to offer geographic computations based on distance evaluations. SPRIDE serves as the first step in addressing the farmers' growing concerns in voluntarily or involuntarily contributing their farming data to agricultural IoT applications, which will finally help popularize the information sharing-based precision agriculture. We believe in the power of big data analytics and the development of IoT systems in revolutionizing the agricultural productions to serve the welfare of farmers and general public. In the future, we will continue to implement and enhance SPRIDE on other coordinate systems and also integrate SPRIDE into real-world agricultural IoT applications such as irrigation, planting planning, and yield management.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential Privacy for Location-based Systems. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'13)*. 901–914.

[3] Apple. 2017. Accessed at October 9, 2019. Differential Privacy. Retrieved from https://images.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.

[4] BlueKrypt. 2020. Accessed at Oct. 17, 2020. Cryptographic key length recommendation. Retrieved from https://www.keylength.com/en/8/.

[5] Aida Boghossian, Scott Linsky, Alicia Brown, Peter Mutschler, Brian Ulicny, Larry Barrett, Glenn Bethel, Michael Matson, Thomas Strang, Kellyn Ramsdell, and Susan Koehler. 2018. Threats to precision agriculture (2018 public private analytic exchange program report). Department of Homeland Security. DOI: https://doi.org/10.13140/RG.2.2.20693.37600

[6] Kamalika Chaudhuri and Claire Monteleoni. 2009. Privacy-preserving logistic regression. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 289–296.

[7] Kamalika Chaudhuri, Anand D. Sarwate, and Kaushik Sinha. 2013. A near-optimal algorithm for differentially private principal components. *J. Mach. Learn.ing Res.* 14, 1 (2013), 2905–2943.

[8] X. Chen, X. Wu, X. Y. Li, Y. He, and Y. Liu. 2014. Privacy-preserving high-quality map generation with participatory sensing. In *Proceedings of the IEEE International Conference on Computer Communications*. 2310–2318.

[9] Chi-Yin Chow, Mohamed F. Mokbel, and Xuan Liu. 2006. A Peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proceedings of the 14th ACM International Symposium on Advances in Geographic Information Systems (GIS'06)*. 171–178.

[10] Graham Cluley. 2018. Accessed at Jan 24, 2021. Privacy of fitness tracking apps in the spotlight after soldiers' exercise routes shared online. Retrieved from https://www.welivesecurity.com/2018/01/30/privacy-fitness-tracking-apps-spotlight-soldiers-exercise-routes-shared-online/.

[11] Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. 2007. Over-encryption: Management of access control evolution on outsourced data. In *Proceedings of the 33rd International Conference on very Large Data Bases.* 123–134.

[12] Wenliang Du, Yunghsiang S. Han, and Shigang Chen. 2004. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the SIAM International Conference on Data Mining.* 222–233.

[13] Cynthia Dwork. 2011. Differential privacy. In *Encyclopedia of Cryptography and Security.* Springer, Boston, MA. 338–340.

[14] Esri. 2008. GIS for sustainable agriculture. *GIS Best Pract.* (Sep. 2008) ESRI Publications, New York.

[15] Jody L. Ferris. 2017. Data privacy and protection in the agriculture industry: Is federal regulation necessary? *Minnesota J. Law, Sci. Technol.* 18, 1 (2017).

[16] Dario Freni, Carmen Ruiz Vicente, Sergio Mascetti, Claudio Bettini, and Christian S. Jensen. 2010. Preserving location and absence privacy in geo-social networks. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10).* 309–318.

[17] Robin Gebbers and Viacheslav I. Adamchuk. 2010. Precision agriculture and food security. *Science* 327, 5967 (Feb. 2010), 828–831.

[18] Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme.* Ph.D. Dissertation. Stanford, CA.

[19] Geokov. 2014. Accessed at October 9, 2019. UTM Projection. Retrieved from http://geokov.com/education/utm.aspx.

[20] P. Hallgren, M. Ochoa, and A. Sabelfeld. 2015. InnerCircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *Proceedings of the 13th Annual Conference on Privacy, Security and Trust (PST).* 1–6.

[21] Informationisbeautiful. 2021. Accessed at October 9, 2019. World's Biggest Data Breaches. Retrieved from http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/.

[22] Suat Irmak, Jennifer M. Rees, Gary L. Zoubek, Brandy S. van DeWalle, William R. Rathje, Rodney DeBuhr, Dan Leininger, Darrel D. Siekman, James W. Schneider, and Andrew P. Christiansen. 2010. Nebraska agricultural water management demonstration network (NAWMDN): Integrating research and extension/outreach. *Appl. Eng. Agric.* 26, 4 (2010), 599–613.

[23] Suat Irmak and Vivek Sharma. 2015. Large-scale and long-term trends and magnitudes in irrigated and rainfed maize and soybean water productivity: Grain yield and evapotranspiration frequency, crop water use efficiency, and production functions. *Trans. ASABE* 58, 1 (2015), 103–120.

[24] Geetha Jagannathan and Rebecca N. Wright. 2005. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the ACM SIGKDD Conference.* 593–599.

[25] Charles F. F. Karney. 2011. Transverse Mercator with an accuracy of a few nanometers. *J. Geodesy* 85, 8 (2011), 475–485.

[26] H. Kido, Y. Yanagisawa, and T. Satoh. 2005. Protection of location privacy using dummies for location-based services. In *Proceedings of the 21st International Conference on Data Engineering Workshops.* 1248–1248.

[27] John Krumm. 2009. *Realistic Driving Trips for Location Privacy.* Springer Berlin, 25–41.

[28] John Krumm. 2009. A survey of computational location privacy. *Pers. Ubiq. Comput.* 13, 6 (Aug. 2009), 391–399.

[29] X. Y. Li and T. Jung. 2013. Search me if you can: Privacy-preserving location query service. In *Proceedings of the IEEE International Conference on Computer Communications. 2013.* 2760–2768.

[30] Yehuda Lindell and Benny Pinkas. 2000. Privacy preserving data mining. In *Proceedings of the Advances in Cryptology Conference.* 36–54.

[31] Chris Y. T. Ma, David K. Y. Yau, Nung Kwan Yip, and Nageswara S. V. Rao. 2010. Privacy vulnerability of published anonymous mobility traces. In *Proceedings of the 16th International Conference on Mobile Computing and Networking (MobiCom'10).* 185–196.

[32] Sergio Mascetti, Dario Freni, Claudio Bettini, X. Sean Wang, and Sushil Jajodia. 2011. Privacy in geo-social networks: Proximity notification with untrusted service providers and curious buddies. *VLDB J.* 20, 4 (Aug. 2011), 541–566.

[33] Samuel K. Moore. 2017. Accessed at October 9, 2019. Superaccurate GPS Chips Coming to Smartphones in 2018. Retrieved from https://spectrum.ieee.org/tech-talk/semiconductors/design/superaccurate-gps-chips-coming-to-smartphones-in-2018.

[34] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption Be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop (CCSW'11).* 113–124.

[35] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and Dan Boneh. 2011. Location privacy via private proximity testing. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'11).*

[36] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'99).* 223–238.

[37] Vern Paxson. 2001. An analysis of using reflectors for distributed denial-of-service attacks. *ACM SIGCOMM Comput. Commun. Rev.* 31, 3 (2001), 38–47.

[38] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. 2009. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *arXiv preprint arXiv:0911.5708* (2009).

[39] Anand D. Sarwate and Kamalika Chaudhuri. 2013. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE Sig. Proc. Mag.* 30, 5 (Aug. 2013), 86–94.

[40] Vivek Sharma, Suat Irmak, Koffi Djaman, and Vasudha Sharma. 2015. Large-scale spatial and temporal variability in evapotranspiration, crop water-use efficiency, and evapotranspiration water-use efficiency of irrigated and rainfed maize and soybean. *J. Irrig. Drain. Eng.* 142, 3 (2015), 04015063.

[41] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* 1310–1321.

[42] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying location privacy. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'11).* 247–262.

[43] John V. Stafford. 2000. Implementing precision agriculture in the 21st century. *J. Agric. Eng. Res.* 76, 3 (Jul. 2000), 267–275.

[44] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. 2017. Privacy loss in Apple's implementation of differential privacy on MacOS 10.12. *arXiv preprint arXiv:1709.02753* (2017).

[45] TOR. 2021. Accessed at October 9, 2019. TOR Project. Retrieved from https://www.torproject.org/.

[46] USDA. 2019. Accessed at Jan. 30, 2021. Which are the top 10 agricultural producing states? Retrieved from https://www.ers.usda.gov/faqs/#Q1.

[47] Jaideep Vaidya and Chris Clifton. 2002. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the ACM SIGKDD Conference.* 639–644.

[48] Jaideep Vaidya, Murat Kantarcıoğlu, and Chris Clifton. 2008. Privacy-preserving naive Bayes classification. *VLDB J.* 17, 4 (2008), 879–898.

[49] Thaddeus Vincenty. 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Surv. Rev.* 23, 176 (1975), 88–93.

[50] Jaroslav Šeděnka and Paolo Gasti. 2014. Privacy-preserving distance computation and proximity testing on earth, done right. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIACCS'14).* 99–110.

[51] Wikipedia. 2021. Accessed at June 29, 2020. Figure of the Earth. Retrieved from https://en.wikipedia.org/wiki/Figure_of_the_Earth#Ellipsoid_of_revolution.

[52] Wikipedia. 2021. Accessed at June 29, 2020. Universal Transverse Mercator coordinate system. Retrieved from https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system.

[53] Wikipedia. 2021. Accessed at October 9, 2019. Haversine formula. Retrieved from https://en.wikipedia.org/wiki/Haversine_formula.

[54] Sjaak Wolfert, Lan Ge, Cor Verdouw, and Marc-Jeroen Bogaardt. 2017. Big data in smart farming–a review. *Agric. Syst.* 153 (2017), 69–80.

[55] Yonghui Xiao and Li Xiong. 2015. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* 1298–1309.

[56] Jia Xu, Ee-Chien Chang, and Jianying Zhou. 2013. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security.* 195–206.

[57] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. 2010. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the IEEE International Conference on Computer Communications.* 1–9.

[58] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional mechanism: Regression analysis under differential privacy. *Proc. VLDB Endow.* 5, 11 (2012), 1364–1375.

[59] Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data(base) Eng. Bull.* (June 2010).

[60] Ge Zhong, Ian Goldberg, and Urs Hengartner. 2007. Louis, Lester and Pierre: Three protocols for location privacy. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies (PET'07).* 62–76.

Received October 2020; revised February 2021; accepted April 2021