SPRIDE: Scalable and Private Continual Geo-Distance Evaluation for Precision Agriculture

Qiben Yan* Hao Yang* Mehmet C. Vuran* Suat Irmak†
*Computer Science and Engineering, University of Nebraska-Lincoln, USA Email: {qyan,hyang,mcvuran}@cse.unl.edu
†Biological Systems Engineering, University of Nebraska-Lincoln, USA Email: suat.irmak@unl.edu

***Abstract*—Precision agriculture relies on real-time data gathering and analysis to maximize yield, minimize environmental impact and reduce cost, which has been envisioned as a new paradigm to revolutionize modern agriculture. However, the collection of farming data, especially geospatial data, raises concerns about potential privacy leakage. In this paper, we propose a novel *scalable and private continual geo-distance evaluation* system, called *SPRIDE*, to allow application servers to provide geographic based services by computing the distances among sensors and farms privately and continuously. The servers determine the distances without learning any additional information about their locations. The key idea of *SPRIDE* is to perform efficient distance evaluations on encrypted locations over a sphere by leveraging a homomorphic cryptosystem. To scale for a large user base, we propose novel and practical performance enhancements based on data segmentation and distance prediction techniques for reducing computation/communication costs. Through extensive experiments on a real world mobile trace dataset, we show *SPRIDE* achieves real-time private distance evaluation on a large network of farms, attaining at least 17 times runtime performance improvement over existing methods. We further show *SPRIDE* can run on resource-constrained mobile devices with low overhead.**

I. Introduction

Precision agriculture has been envisioned as a new paradigm to revolutionize modern agriculture by collecting and processing data in real-time, with the objective of optimizing seeding, irrigation, fertilization, harvesting, and other farming practices [1]. Farm fields usually have spatial variations of soils types, moisture levels, and nutrient availability. Precision agriculture utilizes spatio-temporal information to determine field variability, ensure optimal use of inputs, and maximize the outputs and profits from a farm [2]. By leveraging in-situ sensing, 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 farmers can be fused to make informed decisions. Consequently, precision farming applications rely on the data across multiple farms with detailed spatial information.

Despite its potential advantages, data collection in precision agriculture has raised several concerns about data privacy [3]. More specifically, one of the major concerns of geospatial data collection is *location privacy* [4]. 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 analysis. Unfortunately, we have seen a growing number of server data breach incidents in recent years [5]. The disclosure of location data will be disastrous, since the physical traces of the farms' sensors/machines can not only expose their physical locations, but may also lead to other unintended consequences, such as: exposure of proprietary data analysis algorithm, planting strategies, or crop yield information. This is significantly 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 [1]. Yet, we are not aware of any work that addresses privacy issues in agriculture.

Recently, several location privacy protection mechanisms have been developed in other fields [6]. One popular type of protection mechanisms is based on location obfuscation techniques that transform the true locations into an area or a perturbed location [4]. However, most existing spatial transformation techniques are subject to advanced location inference attacks [7]. More importantly, the perturbation of true locations will affect the utility of geographic services for agriculture. Thus, *a privacy-preserving mechanism to protect location information without sacrificing the application utilities is needed.*

We observe that many farming applications may not need the actual locations to provide geographic services [8]. More specifically, farming apps can perform data analysis based on the geo-distances among farms and sensors. For instance, relationships between the distance of farms and the difference in the soil moisture or temperature levels can be used to guide a better irrigation procedure. Yet, the accuracy and quality of these services depend on gathering information from many farmers. Sharing information is a major issue for farmers until they are ensured that their shared data does not leak any private information. In this paper, rather than protecting the location traces through obfuscation techniques, we investigate the problem of **privacy-preserving continual distance evaluation initiated by a server on a large network of farms, without requiring the farms to disclose any exact location data to the server**.

Note that the distance computation can be conducted between entities such as: farms, sensors, and/or machines on

This work was supported in part by the US National Science Foundation under grants CNS-1566388 and CNS-1619285.

the farm, which can be stationary or mobile. There are three main challenges in designing a privacy-preserving continual distance evaluation mechanism for precision agriculture: *C1*: distance evaluation process should not leak any additional information about any entities' location traces to servers or third-party analysts; *C2*: adversaries eavesdropping communications between entities and the server should not have any chance in compromising the location privacy of individual entities; *C3*: since the entities are sometimes moving 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, so that the server is capable of tracking the distances among multiple entities in real-time. Recently, researchers have investigated privacy-preserving distance calculation [9], [10] and proximity protocols [11], [12], with the goal of addressing C1 and C2 for mobile applications. However, they all focus on privacy-preserving computation of distance between two users: Alice and Bob, while the distance evaluation among a number of entities in a continuous manner (C3) remains untouched. Hence, scalability is still an open issue in this realm.

In this paper, we present *SPRIDE*, a cloud-enabled Scalable and PRIvate continual geo-Distance Evaluation for precision agriculture. SPRIDE utilizes homomorphic cryptosystem to protect the locations of individual farms or sensors. Note that homomorphic cryptosystem has been used in privacy-preserving distance calculation [10], [11] 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 regimen) based on the calculated distances, or send the distance information to the 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 applications, there are other potential mobile applications that can build upon SPRIDE. *Private smart navigation* is one attractive example, 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 uploading their GPS location data to the server. The server uses SPRIDE to privately and continuously calculate the distance between mobile users, which can facilitate the *identification of traffic conditions* on the road.

To the best of our knowledge, this is the first work that investigates geospatial data privacy issues in precision agriculture. This paper makes the following contributions:

- We design and implement a cloud-based private continual geo-distance evaluation system, SPRIDE, for practical privacy-preserving distance tracking in precision farming applications. SPRIDE utilizes cloud-based distance measurement based on a homomorphic cryptosystem to support large-scale distance evaluations for multiple farms.
- We propose practical performance enhancements to improve the real-time distance evaluation performance of SPRIDE system, using data segmentation and distance prediction techniques. The enhanced SPRIDE attains superior real-time distance tracking performance, which can be scaled to accommodate a large number of farms.
- We conduct comprehensive experiments to evaluate SPRIDE's distance computation performance using a real world dataset, and show the real-world applicability of the SPRIDE system for distance tracking.

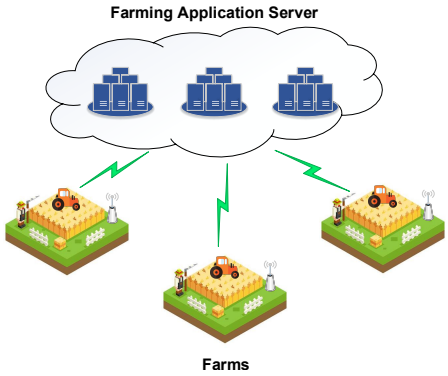


Figure 1: Precision agriculture system model

The rest of the paper is organized as follows. Section II introduces the system model and relevant background information. Then, Section III presents the SPRIDE system design. The system evaluation is demonstrated in Section IV. After reviewing related work in Section V, the paper concludes with Section VI.

II. System Model and Background

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

A. 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 Fig. 1. App cloud server performs data collection and analysis, and provides geographic services to the farmers, while the farmers receive services without disclosing their true locations to the app cloud server. The cloud server is considered as semi-honest (honest but curious), who is curious about *individual users*' whereabouts but follows the protocols honestly. We assume the origin of the users' geospatial data is successfully hidden by using anonymized protocols such as Tor networks [13], so that IP geolocation mechanism will not be able to determine the geolocation of hosts based on IP addresses of

the packets. The farmers are also considered as semi-honest, who may attempt to infer other farms/devices' locations when they are not nearby. On the other hand, there is no need to protect users' location privacy against nearby users (i.e., the distance value is small). Note that we do not consider malicious users faking their locations, who can be mitigated by tamper-resistant devices or unforgeable location tags [9]. We do not consider colluding users sharing information with each other, and also the denial of service attack is out of scope.

B. Homomorphic Encryption

SPRIDE utilizes Homomorphic Encryption (HE) to protect the location data. HE allows arbitrary computations (e.g., additions, multiplications, quadratic functions, *etc.*) on ciphertexts while preserving decryptability. The most powerful HE, namely fully Homomorphic Encryption (FHE), supports unlimited number of additions and multiplications, which has been utilized to support privacy-preserving data analytics [14]. However, FHE system is computational costly to be used in real-time applications [15]. Partially Homomorphic Encryption (PHE) supports limited operations on ciphertexts, which is more efficient and also applicable to our scenario. One implemention of PHE, namely the Paillier's system [16], is a simpler and more efficient PHE which will be used 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 $r \in \mathcal{Z}_n^*$ 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: $E_{pk}(m_1) \cdot E_{pk}(m_2) = E_{pk}(m_1 + m_2),\ E_{pk}(m_1)^{m_2} = E_{pk}(m_1 \cdot m_2)$.

C. 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 geographic coordinate (i.e. Latitude and Longitude) system as used by GPS navigation systems. UTM coordinate system provides a referencing frame to define the positions of objects. SPRIDE is designed to work on UTM format data, as the UTM system greatly simplifies the distance calculation between two objects on earth.

With UTM, the Earth is divided into 60 zones, each being a six-degree band of longitude. To minimize the scale distortion within each zone, each of the 60 zones gets projected onto a plane separately. The meridian at the center of each zone is called the *central meridian* (CM). Each zone is divided into horizontal grids with 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. The 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 $[0, 9,999,999]$ [17]. For instance, the USA Contiguous States are residing in 10 UTM zones, and y ranges in $[2,700,000, 5,500,000]$. One can use the simple *Pythagorean Theorem* to calculate the distance between two coordinates in UTM form [17].

III. SPRIDE Design

In a nutshell, SPRIDE offers a continual distance evaluation system for precision farming. 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 in real time. Then, we enhance the performance of SPRIDE to scale to a large user base. SPRIDE is supported by a cloud infrastructure with a cloud server serving a large network of farms to enable cross-farm data analytics.

A. 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 [18]. 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., $d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$).

B. Cloud-Based Privacy-Preserving Distance Measurement

The server will compute the distance between any pair of users requesting the service, while the users are required to 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 useful, the server divides a large service area into several sections. The users will disclose the *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 area segmentation for farming applications is orthogonal to this work, and we plan to investigate it in future.

To start the process, each user will send the section information to the cloud, based on which the cloud identifies each user's zone. SPRIDE uses zone information to identify three different types of *positional relationships* (PRs), when we need to 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

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

locations are within different hemispheres in the same zone; (3) Type III PR: two locations are in neighboring zones.

Note that when the distance between two objects enlarges, errors in distance calculation increases [17]. However, farming applications mostly consider to associate data from farms that are not too far away from each other (but also not nearby). Thus, SPRIDE practically avoids distant users, thereby ensuring the accuracy of distance measurement. Here, we present the algorithm for Type I PR, corresponding to the most common case when two locations are residing 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 asymmetric cryptosystem 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. As a result, we minimize the usage of Paillier encryption in SPRIDE. In fact, we only apply Paillier encryption to compute the *scalar product*, which greatly improves the computational efficiency. The complete privacy preserving distance measurement algorithm is shown in Algorithm 1.

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

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

For the distance measurement between user U_i and user U_j, U_i needs to compute two encryptions and one decryption, while U_j only needs to perform two exponentiations. There are two rounds of communications between cloud server and users: In the *Preparation*, the user' location components, including $x_i^2 + y_i^2$ and *encrypted location*, are uploaded periodically, allowing cloud server to track the real-time distance measurement. In the *First Round*, the cloud then interacts with each user (in this case, U_j) to get the *encrypted product*. In the *Second Round*, the cloud interacts with the other user U_i to get the decrypted *scalar product*. In the end, the cloud server gets the distance measurement, without knowing the locations of any users. As the cloud receives $x_i^2 + y_i^2$, $x_j^2 + y_j^2$, and obtains $x_i x_j + y_i y_j$ during the *Second Round*, the distance d_{ij} can be computed according to (2). Meanwhile, each user keeps its location private from other users. This algorithm can be easily extended to compute the distance between a user and a fixed Point of Interest (PoI), in which case, the cloud will execute the homomorphic operations with the known PoI location.

The cloud-based distance measurement can effectively compute the distance between any pair of users or the distance from a user to a fixed PoI on a map. The cloud is capable of computing multiple distance measurements from one end (e.g. U_i) to multiple users. In such case, the cloud will send the *encrypted location* from U_i to all the other users. All the other users receiving multiple *encrypted locations* 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 product*.

Algorithm 1: Cloud-based Privacy Preserving Distance Measurement

1. **Setup**: N users have their own locations in UTM format (z_i, x_i, y_i), $i \in [1, N]$;
2. Each user U_i is assigned a pair of private key and public key of Paillier's cryptosystem (sk_i, pk_i). $\texttt{Enc}_{pk}$ denotes the Paillier encryption;
3. **Preparation**: Each user U_i uploads $x_i^2 + y_i^2$ to cloud server, and generates the ciphertexts of *encrypted location* $\texttt{Enc}_{pk_i}(x_i)$, $\texttt{Enc}_{pk_i}(y_i)$ and sends them to cloud server;
4. **First Round**: If cloud server initiates the process of computing the distance of user U_i and user U_j , the cloud server sends the *encrypted location* from U_i to U_j;
5. After receiving the ciphertexts, U_j executes the homomorphic operation and sends the *encrypted product* $\texttt{Enc}_{pk}(x_i)^{x_j} \cdot \texttt{Enc}_{pk}(y_i)^{y_j}$ to the cloud;
6. **Second Round**: the cloud sends the *encrypted product* back to U_i for decryption;
7. U_i then decrypts the *scalar product* $x_i x_j + y_i y_j$, and sends it back to the cloud;
8. The cloud computes the distance between U_i and U_j using Eq. (2).

C. Security Analysis of Cloud-Based Privacy Preserving Distance Measurement

In Algorithm 1, it can be observed that users share some unencrypted information with the cloud server. In this section, we analyze the potential security risks of sharing these information. The first piece of 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, 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.

The second piece of USI is $x_i^2 + y_i^2$ (note that $x_i \geq 166,000$, $y_i \geq 0$), which is the squared sum of northing and easting coordinates. As $|500,000 - x_i|$ is the horizontal distance between CM and the user, and y_i is the vertical distance between CM and the user, the privacy leakage of $x_i^2 + y_i^2$ depends on the values of x_i, y_i. To quantify the privacy leakage, we hypothetically regard x_i as the horizontal distance. Then, the value of $\sqrt{x_i^2 + y_i^2}$ stands for the radius of a circular area, and the user can be located on any point of the circle. Note that x_i is always a six-digit number as shown in Section II-C, as a result, $\sqrt{x_i^2 + y_i^2}$ is at least a six-digit radius with circumference of at least $600km$. Thus, even the lower-bound probability of learning the exact location of a user on the circle is tiny.

The third piece of USI is the *scalar product*: $x_i x_j + y_i y_j$, which represents a mixture of two location data. The disclosure of *scalar product* also does not compromise the individual user's location privacy, since the two locations under consideration are intertwined with each other in *scalar product*. As for the encrypted information from users, the cloud server does not possess the decryption capability without the private keys, which are held only by the respective users.

The attackers eavesdropping the communications between the users and the cloud server can only receive the same three types of unencrypted information. They basically can only gain the same knowledge as the cloud server. Also, without knowing the private keys, sensitive location data enclosed in the encrypted message will not be leaked to the attackers. In step 5, user U_j receives the encrypted location of U_i, which cannot be decrypted without the private key. On the other hand, although U_i owns the private key, he/she only receives the encrypted product that contains no sensitive information. Therefore, SPRIDE protects any user location from other users.

In summary, we demonstrate that the cloud server and outside attackers are not able to extract sensitive location data. However, on the other hand, due to the nature of SPRIDE system, the distance information is presented to both cloud server and the attackers. Now, we evaluate the privacy concerns brought by the distance disclosure. For the server, 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 distance γ, without knowing the location of U_i or U_j, it is almost impossible to guess the exact location of the other counterpart.

Location Triangulation: One potential security weakness in SPRIDE is that distance measurement allows multiple colluding parties to do location triangulation to pinpoint a specific user's location. Note that any distance measurement protocol will be subject to colluding attacks [10]. In this paper, we do not consider colluding attacks from malicious users, which is an interesting research topic in its own right.

In case one end of distance measurement is known, the adversaries might be able to pinpoint the location of the other end. For example, if one end of distance measurement is a river, by knowing the distance between a user and the river, a determined adversary can pinpoint the location of this user by examining the circular area with the specific distance to the river on a map. Thus, instead of measuring distance, we can design a distance comparison mechanism to further strengthen location privacy by providing a coarse-grained binary comparison output. The design of privacy-preserving distance comparison mechanism will be our future work.

D. Enhanced SPRIDE: Performance Enhancement of SPRIDE

Efficient Privacy-Preserving Distance Evaluation Using Data Segmentation: The UTM location data contains easting and northing coordinates, which are usually 6-7 digits. The major computations of SPRIDE include Paillier encryption, Paillier decryption and the homomorphic operations. According to our experiments, the most time consuming computations of Algorithm 1 are Step 5 (homomorphic operations) and Step 7 (Paillier decryption). Here, we use the key size of 1024 bits, which corresponds to the ciphertexts of 2048 bits. The 2048-bit ciphertext will be raised to the power of a 6 or 7 digits exponent, which consumes a considerable amount of computational power. Moreover, the decryption of this huge ciphertext is very computationally expensive. In our experiment, we generate different lengths of messages to evaluate the time costs of the encryption, decryption, and homomorphic operations. For each time cost evaluation, we average the time consumption performance over 10 runs with random messages. We show the average time consumption breakdown in Table I, where a significant portion of time is spent on homomorphic operations and Paillier decryption. The experimental setting is discussed in Section IV.

Operations	3-digits	4-digits	5-digits	6-digits	7-digits
Original Encryption	0.0329	0.0358	0.0367	0.0376	0.0387
Original Homomorphic Operations	0.043	0.525	3.46	80.9	207.4
Original Decryption	0.045	0.49	1.36	39.5	118.9
Original SPRIDE	-	-	-	-	533.6
Enhanced SPRIDE	-	-	-	-	1.40

Table I: Time consumption breakdown for processing a single message with different number of digits using original/enhanced SPRIDE system (in ms)

To improve the computational efficiency, we propose an enhancement of SPRIDE. The basic idea is to segment the UTM location data into single-digit data, and to perform homomorphic operations over the single-digit exponent, the results of which can be aggregated to form the expected SPRIDE output.

Generally, x_i is a six-digit number, and y_i is a seven-digit number. We first convert the easting coordinate into a seven-digit number by adding a "0" in front of the original coordinate. As an example, seven-digit UTM locations $x_i = A_1A_2A_3A_4A_5A_6A_7$, $y_i = A'_1A'_2A'_3A'_4A'_5A'_6A'_7$ from U_i and $x_j = B_1B_2B_3B_4B_5B_6B_7$, $y_j = B'_1B'_2B'_3B'_4B'_5B'_6B'_7$ from U_j are segmented into seven single-digit data A_m, A'_m, and B_m, B'_m, $m \in [1,7]$, where A_1 and B_1 are "0". Then, U_i performs Paillier encryption over these single-digit data, and U_j computes $\texttt{Enc}_{pk}(A_m)^{B_n}$, $\texttt{Enc}_{pk}(A'_m)^{B'_n}$, where $m \in [1,7]$, $n \in [1,7]$. There will be $7^2 = 49$ values of $\texttt{Enc}_{pk}(A_m)^{B_n}$ and $\texttt{Enc}_{pk}(A'_m)^{B'_n}$, while $x_i x_j$ can be written as:

$$\begin{aligned} x_i x_j = &(A_1B_1 10^{12} + A_1B_2 10^{11} + \ldots + A_1B_7 10^6) \\ &+ (A_2B_1 10^{11} + A_2B_2 10^{10} + \ldots + A_2B_7 10^5) \\ &+ \ldots \\ &+ (A_7B_1 10^6 + A_7B_2 10^5 + \ldots + A_7B_7). \end{aligned} \tag{3}$$

Moreover, $y_i y_j$ can be presented in a similar form. Homomorphic operation can be performed over these single-digit

data, for example, $\texttt{Enc}_{pk}(A_m)^{B_n} * \texttt{Enc}_{pk}(A'_m)^{B'_n}$ can be decrypted into $A_m B_n + A'_m B'_n$, which can then be aggregated and powered (with appropriate powers ranging from 10^0 to 10^{12}) to compute $x_i x_j + y_i y_j$.

One caveat of this approach is that: as A_1 and B_1 are both 0, U_j's location privacy can be easily compromised. For example, U_i will receive $\texttt{Enc}_{pk}(A_1)^{B_2} * \texttt{Enc}_{pk}(A'_1)^{B'_2}$ in *Second Round*, which is decrypted into $A_1 B_2 + A'_1 B'_2$. In case that $A_1 = 0$, user U_i can recover B'_2. Similarly, user U_i is able to retrieve $B'_1, B'_2, \ldots, B'_7$ for y_j, as well as $B_1, B_2, \ldots, B_7$ for x_j. To address this issue, we first lay out the components containing A_1 or B_1 in the final result of $x_i x_j + y_i y_j$:

$$A_1 B_1 10^{12} + A_1 B_2 10^{11} + \ldots + A_1 B_7 10^6$$

$$+ A_2 B_1 10^{11} + A_3 B_1 10^{10} + \ldots + A_7 B_1 10^6,$$

and the corresponding encrypted products are: $\texttt{Enc}_{pk}(A_1)^{B_1} * \texttt{Enc}_{pk}(A'_1)^{B'_1}$, $\ldots$, $\texttt{Enc}_{pk}(A_7)^{B_1} * \texttt{Enc}_{pk}(A'_7)^{B'_1}$, which will lead to the leakage of U_j's location. Rather than computing the above encrypted products, U_j will compute additional products to conceal U_j' locations, including: for instance,

$$\texttt{Enc}_{pk}(A_1)^{B_2} * \texttt{Enc}_{pk}(A'_1)^{B'_2} * \texttt{Enc}_{pk}(A_2)^{B_1} * \texttt{Enc}_{pk}(A'_2)^{B'_1},$$

$$\texttt{Enc}_{pk}(A_1)^{B_7} * \texttt{Enc}_{pk}(A'_1)^{B'_7} * \texttt{Enc}_{pk}(A_7)^{B_1} * \texttt{Enc}_{pk}(A'_7)^{B'_1},$$

which are decrypted into: $A'_1 B'_2 + A'_2 B'_1$, $A'_1 B'_3 + A'_3 B'_1$, $\ldots$, $A'_1 B'_7 + A'_7 B'_1$ (since A_1, $B_1 = 0$). In this case, with two unknowns in each component (for instance, B'_1 and B'_2), user U_i will not be able to retrieve U_j's location data from the decrypted products. Now, the only missing component of $x_i x_j + y_i y_j$ is: $\texttt{Enc}_{pk}(A_1)^{B_1} * \texttt{Enc}_{pk}(A'_1)^{B'_1}$, which is used to compute $A_1 B_1 10^{12} + A'_1 B'_1 10^{12}$ (with $A_1 B_1 10^{12} = 0$). Recall that we have $(A'_1 B'_2 + A'_2 B'_1) * 10^{11}$ inside $x_i x_j + y_i y_j$. To include the missing component is straightforward, U_j simply computes: $\texttt{Enc}_{pk}(A_1)^{B_1 * 10} * \texttt{Enc}_{pk}(A'_1)^{B'_1 * 10}$, which decrypts into: $A_1 B_1 * 10 + A'_1 B'_1 * 10$. Then, U_j can add them into $(A'_1 B'_2 + A'_2 B'_1) * 10^{11}$, resulting in the computation of:

$$\texttt{Enc}_{pk}(A_1)^{B_2} * \texttt{Enc}_{pk}(A'_2)^{B'_1} * \texttt{Enc}_{pk}(A_2)^{B_1} *$$
$$\texttt{Enc}_{pk}(A'_2)^{B'_1} * \texttt{Enc}_{pk}(A_1)^{B_1 * 10} * \texttt{Enc}_{pk}(A'_1)^{B'_1 * 10},$$

which decrypts into $A_1 B'_2 + A_2 B'_1 + A'_1 * B'_1 * 10$. After that, the cloud can raise it into the power of 10^{11} for computing $x_i x_j + y_i y_j$.

To this end, the homomorphic operation now turns into 49 pairs of simple operations $\texttt{Enc}_{pk}(A_m)^{B_n} \cdot \texttt{Enc}_{pk}(A'_m)^{B'_n}$, where $m \in [1, 7]$, $n \in [1, 7]$. Correspondingly, the Paillier decryption results in $A_m B_n + A'_m B'_n$. The cloud can use Eq. (3) to recover $x_i x_j + y_i y_j$. Essentially, this improved algorithm converts a computation of a long-digit number into multiple computations of a short-digit number. In this case, we performed 49 pairs of simple homomorphic operations instead of one pair of super-complex homomorphic operation. The performance improvement is significant.

The computational time with/without the enhancement is shown in Table I, where we can see a noteworthy performance improvement brought by the enhancement scheme (improved from $534ms$ to $1.40ms$ for processing a seven-digits message).

Algorithm 2: Distance Prediction Algorithm

```
Set prediction range r, prediction STD threshold λ, error bound err;
repeat
    Take a continuous set of distance measurements with cardinality equal to the prediction range r, including d_1, d_2, ..., d_r;
    Compute moving speed speed_i = |d_{i+1} − d_i|, i ∈ [1, r − 1];
    if std(speed_i) < λ then
    repeat
        Set r to 2r except the initial repeat round;
        Enter prediction mode;
        Predict a set of distance measurements with cardinality equal to the prediction range r, including d'_{r+1}, d'_{r+2}, ..., d'_{2r}, where d'_j = d'_{j−1} + mean({speed_i}), j ∈ [r + 1, 2r], i ∈ [1, r − 1], d'_r = d_r;
        Perform a distance measurement d_{2r} for distance verification;
    until |d_{2r} − d'_{2r}| > err;
    Exit prediction mode ;
until;
```

Reducing Computation/Communication Time using Distance Prediction: When the users are moving, distance measurement among users needs to be carried out periodically to allow app cloud server to keep track of users' real-time movement. 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 using distance prediction algorithm. The distance prediction algorithm aims to reduce the distance measurement frequencies by capturing moving statistics. SPRIDE enters the *prediction mode* based on historic distance measurements, when a series of distance measurements are predicted without any computations/communications, thereby conserving computational resources and communication overhead. The complete distance prediction algorithm is shown in Algorithm 2.

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. The entry to the prediction mode is determined by the standard deviation of moving speed values measured inside prediction range. 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, 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}, ..., 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, we continue the distance prediction; otherwise, we exit the distance prediction and restart the whole process.

Using distance prediction algorithm, not only can we save computation/communication time spent on distance evaluation, but we 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 IV-D.

IV. Evaluation

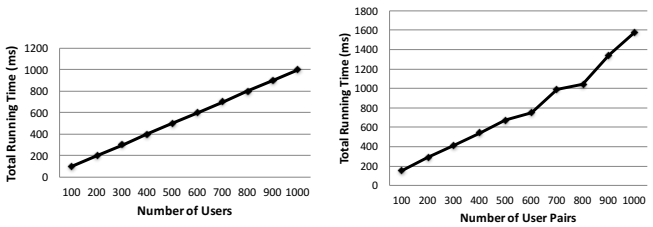
In this section, we evaluate the distance evaluation performance of the SPRIDE system. Specifically, we first focus on the runtime performance of SPRIDE system for distance measurement between users and a fixed PoI, or among multiple users, respectively. Note that the users denote farms, farmers, sensors, or machines, depending on specific applications. Then, 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. All the algorithms are implemented in Java and run on a MacBook with 2.2GHz Intel Core i5 and 8GB memory, or Android Nexus 5 phones (only for experiments in Section IV-E).

A. Datasets

As we cannot find real-world datasets of farming applications, we use a dataset, called Geolife dataset [19], 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. 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. We randomly selected trajectories from users inside the dataset to perform real-time distance evaluation.

B. Runtime Performance for Distance Evaluation with a Fixed Location

We implement the enhanced SPRIDE system using data segmentation, and evaluate the performance of distance evaluation between users and a fixed PoI using Algorithm 1. Recall that the user is responsible for Paillier encryption and decryption, while the app cloud will perform homomorphic operation. In our experiment, we assume the user and app cloud have the same hardware configurations (i.e. MacBook). We evaluate the total computation time of the users w.r.t. the number of users involved in distance evaluation. We run



(a) (b)

Figure 2: (a). Total computation time for computing distance between different number of users and a fixed PoI; (b). Total computation time for computing distance between different user pairs

Method	PLQP [20]	PP-UTM [10]	SPRIDE
Time (s)	741	27	1.6

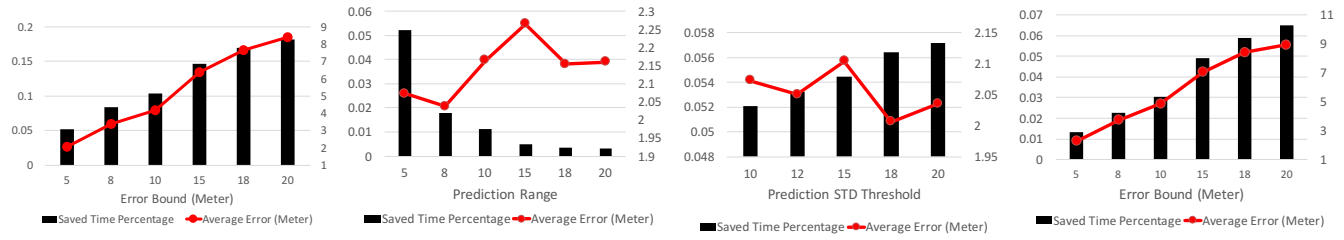
Table II: Time cost for 1000 user pairs

experiments on a randomly selected set of users 10 times, and compute the average total time consumption. The result is shown in Fig. 2(a), which demonstrates a linear increase in computation time with the increasing number of users. Using the MacBook, the distance measurement with $1,000$ users takes only about 1 second. Therefore, SPRIDE can support the distance evaluation of a large number of users. As for the network communication, both Algorithm 1 require two round-trips between app cloud and users, the overhead of which is shown in Section IV-E.

C. Runtime Performance for Distance Evaluation with Multiple Farms

In real-world applications, the users on a farm update their locations periodically, for instance, every five seconds. With superior runtime performance, SPRIDE system can track and update distance evaluation for a large number of users in real-time, without compromising users' location privacy. Next, we evaluate the distance measurement for multiple users, where U_i is responsible for encryption and decryption, while U_j performs homomorphic operations. The total computation time is evaluated w.r.t. the pairs of users. We randomly select pairs of users, repeat the experiments for 10 times, and report average time costs. The results are shown in Fig. 2(b). Similar to the previous result, the total computation for 700 user pairs takes around 1 second. Note that in this case, the cloud is merely a message relay, who delegates most of the computations to each individual user. Apparently, the app cloud is capable of supporting a large user base.

Comparing with Other Methods: For comparison, we implement two existing privacy-preserving distance measurement methods: PLQP [20] and PP-UTM [10], both of which measure distance between a single pair of users using homomorphic encryption, and PP-UTM further computes distance over UTM projection. Table II shows that SPRIDE achieves 463 times improvement over PLQP, and 17 times improvement over PP-UTM in time consumption, which highlights the scalability of SPRIDE system, making it particularly suitable



(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)

Figure 3: 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 distance evaluation.

Error Bound	5
Prediction Range	5
Prediction STD Threshold	10

Table III: Default parameter settings

$U_i \rightarrow$ Cloud	Cloud $\rightarrow U_i$	$U_j \rightarrow$ Cloud	Cloud $\rightarrow U_j$
1792	5376	5376	1792

Table IV: Communication overhead of enhanced SPRIDE (in Bytes)

for farming applications processing geolocation data from a large set of farms.

D. 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, 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 are shown in Table III. For the following experiments, we vary the value of one parameter while retaining default values for other parameters if not mentioned. The average performance is reported in Fig. 3.

The Impact of Error Bound: In Fig. 3(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, we will spend less time on distance computation. For instance, when error bound is set as 10 meters, 10% of total computation time can be saved. Meanwhile, the average prediction error rises with increasing error bound. 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, performance improvement trend is similar as shown in Fig. 3(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 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 error bound is set to 15 meters.

The Impact of Prediction Range: The relationship between saved time percentage and prediction range is shown in Fig. 3(b), which shows that time savings decline with increasing prediction range. This is due to the increasing difficulty in entering prediction mode. If we take more points for prediction mode evaluation, 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 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.

The Impact of Prediction STD Threshold: Higher prediction STD threshold brings more time savings as shown in Fig. 3(c). The reason is obvious, as it becomes easier to enter the prediction mode with a higher prediction STD threshold. Average prediction error does not seem to have any relationship with prediction STD threshold.

E. Overhead Evaluation on Mobile Device

Since mobile device has constrained computation 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 enhanced SPRIDE, each distance evaluation takes U_i around 0.5ms for Paillier encryption and decryption, while it takes U_j around 3ms for homomorphic operation, which is acceptable. The communication overhead is listed in Table IV, the total of which is less than 15 Kilobytes. In addition, using the distance prediction algorithm, we significantly reduce the distance evaluation frequency and computation/communication costs. Therefore, SPRIDE system introduces low overhead, and can be applied in real-world privacy-preserving farming applications on mobile devices.

V. Related Work

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 [4]. Several location obfuscation techniques add noise to the users' location coordinates [21], hide the real location among a set of dummy locations [22], or use cloaking algorithm to conceal real location [23]. Privacy-preserving proximity test has been widely studied. InnerCircle [12] is a proximity protocol based on homomorphic cryptosystem. Freni *et al.* [24] propose to provide a course granularity of location data to protect location privacy, while Mascetti *et al.* [25] extends this work to a centralized scenario. Different from the previous work, SPRIDE is a practical system that tracks geo-distances continuously on earth for precision farming, rather than proximity test.

Other approaches generate fake locations to hide users' true locations, by constructing fake trips with more probable paths traveled by drivers [26]. Chen *et al.* [27] 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.* [10] homomorphically compute 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 continuous distance tracking for multiple users in a scalable manner. With performance enhancement, our system consumes less resources.

VI. Conclusion

In this paper, 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. SPRIDE leverages homomorphic cryptosystem to perform distance evaluation on user-encrypted location data. During the distance evaluations, the geolocations are protected against other farms, app cloud and external adversaries. We further enhanced the performance of the real-time distance evaluation using data segmentation and distance prediction techniques. We showed through experiments with a real-world dataset that SPRIDE can process a large number of farms' encrypted locations to provide geographic applications based on distance evaluations in real-time.

References

[1] R. Gebbers and V. I. Adamchuk, "Precision agriculture and food security," *Science*, vol. 327, no. 5967, pp. 828–831, 2010.

[2] Esri, "Gis for sustainable agriculture," *GIS Best Practices. New York: ESRI Publications*, 2008.

[3] J. L. Ferris, "Data privacy and protection in the agriculture industry: Is federal regulation necessary?" *Minnesota Journal of Law, Science & Technology*, vol. 18, no. 1, 2017.

[4] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, ser. SP '11, 2011, pp. 247–262.

[5] informationisbeautiful, "World's biggest data breaches," http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/, Accessed at July 9, 2016.

[6] J. Krumm, "A survey of computational location privacy," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, 2009.

[7] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, 2015, pp. 1298–1309.

[8] J. V. Stafford, "Implementing precision agriculture in the 21st century," *Journal of Agricultural Engineering Research*, vol. 76, no. 3, pp. 267 – 275, 2000.

[9] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location privacy via private proximity testing," 2011.

[10] J. Šeděnka and P. Gasti, "Privacy-preserving distance computation and proximity testing on earth, done right," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '14, 2014, pp. 99–110.

[11] G. Zhong, I. Goldberg, and U. Hengartner, "Louis, lester and pierre: Three protocols for location privacy," in *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, ser. PET'07, 2007, pp. 62–76.

[12] P. Hallgren, M. Ochoa, and A. Sabelfeld, "Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol," in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, July 2015, pp. 1–6.

[13] TOR, "Tor project," https://www.torproject.org/, Accessed at July 9, 2016.

[14] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford, CA, USA, 2009.

[15] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, ser. CCSW '11, 2011, pp. 113–124.

[16] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, ser. EUROCRYPT'99, 1999, pp. 223–238.

[17] Geokov, "UTM projection," http://geokov.com/education/utm.aspx, Accessed at July 9, 2016.

[18] C. F. F. Karney, "Transverse mercator with an accuracy of a few nanometers," *Journal of Geodesy*, vol. 85, no. 8, pp. 475–485, 2011.

[19] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *IEEE Data(base) Engineering Bulletin*, June 2010.

[20] X. Y. Li and T. Jung, "Search me if you can: Privacy-preserving location query service," in *Prof. of IEEE INFOCOM 2013*, April 2013, pp. 2760–2768.

[21] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '13, 2013, pp. 901–914.

[22] H. Kido, Y. Yanagisawa, and T. Satoh, "Protection of location privacy using dummies for location-based services," in *21st International Conference on Data Engineering Workshops*, April 2005, pp. 1248–1248.

[23] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, ser. GIS '06, 2006, pp. 171–178.

[24] D. Freni, C. Ruiz Vicente, S. Mascetti, C. Bettini, and C. S. Jensen, "Preserving location and absence privacy in geo-social networks," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ser. CIKM '10, 2010, pp. 309–318.

[25] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia, "Privacy in geo-social networks: Proximity notification with untrusted service providers and curious buddies," *The VLDB Journal*, vol. 20, no. 4, pp. 541–566, Aug 2011.

[26] J. Krumm, *Realistic Driving Trips For Location Privacy*. Springer Berlin Heidelberg, 2009, pp. 25–41.

[27] X. Chen, X. Wu, X. Y. Li, Y. He, and Y. Liu, "Privacy-preserving high-quality map generation with participatory sensing," in *Proc. of IEEE INFOCOM 2014*, April 2014, pp. 2310–2318.