Homomorphic Proximity Computation in Geosocial Networks

Peizhao Hu, Tamalika Mukherjee, Alagu Valliappan and Stanisław Radziszowski
Department of Computer Science
Rochester Institute of Technology, USA
Email: ph@cs.rit.edu

***Abstract*—With the growing popularity of mobile devices that have sophisticated localization capability, it becomes more convenient and tempting to give away location data in exchange for recognition and status in the social networks. Geosocial networks, as an example, offer the ability to notify a user or trigger a service when a friend is within geographical proximity. In this paper, we present two methods to support secure distance computation on encrypted location data; that is, computing distance functions without knowing the actual coordinates of users. The underlying security is ensured by the homomorphic encryption scheme which supports computation on encrypted data. We demonstrate feasibility of the proposed approaches by conducting various performance evaluations on platforms with different specifications. We argue that the novelty of this work enables a new breed of pervasive and mobile computing concepts, which was previously not possible due to the lack of feasible mechanisms that support computation on encrypted location data.**

I. Introduction

Emerging geosocial networks have made it easier to stay connected with friends, especially those who are within close geographical proximity [1], [2]. These networks are motivated by the rapid adoption of mobile devices [3], and the sophistication of localization techniques. For example, social mobile applications (such as Facebook, Tinder) recommend like-minded individuals to users based on geographic proximity, or notify users if a friend is within close vicinity. The growth in this practice raises concerns about user location privacy [4], [5]. Similar concerns have been raised in other pervasive systems, such as participatory sensing and peer-to-peer social gaming in which location data of the participants should be protected [2], [6].

Although many service providers support privacy preserving mechanisms (anonymity, obfuscation, classical cryptography) when they collect users' location data, there are limitations to the existing approaches [2], [7]. In this paper, we will tackle the challenge of supporting computations on location data, while preserving user privacy. To balance data privacy and utility, we propose the use of homomorphic encryption (HE) [8], which has been recognized as the "holy grail" of cryptography and cloud computing [9] owing to its ability to perform computations on encrypted data. This paper presents a practical construction of secure proximity computation using HE. Although the performance of this construction should be still substantially improved, this paper provides motivation for implementing such a real-world application using HE.

Our work focuses on the NLV2011 [10] Somewhat HE (SWHE) scheme, which is a practical construction of a well studied SWHE scheme — BV2011 [11], [12]. A SWHE scheme supports only limited number of computations before the ciphertexts become too noisy for decryption. Based on the NLV2011 scheme, we propose two methods for computing distance between users over encrypted coordinates. The first approach converts GPS coordinates into the Universal Transverse Mercator (UTM) coordinates, and computes Euclidean distance to approximate the actual distance on the Earth's surface. The second approach (also known as *spatial cloaking* [13]) uses a geo-hashing technique to transform GPS coordinates into their linear representation (as binary strings); the binary representation preserves the locality of points. Thus, given two binary strings each representing a point, the longer the common prefix between two binary strings the closer the two points are. Based on this property, we develop homomorphic string matching operations to compute an encrypted form of the common prefix over the encrypted coordinates. The encrypted common prefix is then used to compute a bounding box that gives the minimum and maximum distance of the two points, rather than a precise distance value. Throughout the entire process, precise coordinates are used for computation, but they are protected by the encryption. The final encrypted results are sent back to users who then decrypt the results with a private key. Currently, our prototype uses a single-key SWHE scheme, but we plan to extend it to incorporate multi-key schemes in our future work.

In summary, we make three main contributions in this paper: (i) to our best knowledge, the first practical construction of secure computation on location data using SWHE, (ii) a novel construction that combines a geo-hashing technique and SWHE scheme to achieve privacy preserving distance computation, and (iii) feasibility study of the proposed approaches and evaluation on platforms with different specifications.

The remainder of this paper is organized as follows. Section II gives a brief introduction to homomorphic encryption, in particular we discuss the construction of the NLV2011 scheme. This is followed by the design of two proposed methods of secure distance computation using SWHE in Section III. We present a performance evaluation of the proposed solutions in Section IV. Section V provides an overview of related work on HE and other privacy preserving techniques. Section VI concludes the paper and discusses possible future work.

II. Homomorphic Encryption

Homomorphic encryption (HE) [8] is a cryptographic technique that preserves privacy through encryption while supporting computations over encrypted data. In a nutshell, for messages m and m' we want the following properties to hold for encryption using a key k.

$$Enc(m,k) + Enc(m',k) = Enc(m+m',k),$$
$$Enc(m,k) * Enc(m',k) = Enc(m*m',k),$$

where applying addition $+$ and multiplication $*$ to ciphertexts has the same effect as applying these operations to plaintexts and then encrypting the results. Since all functions can be broken down into these basic operations, we could theoretically construct Fully Homomorphic Encryption (FHE) schemes that perform arbitrary computations on encrypted data. In this paper, we base our discussion on Ring-Learning With Errors (Ring-LWE) schemes, in particular NLV2011 [10]. The Ring-LWE assumption is stated as follows:

If we uniformly sample s and a_i from a ring $R_q = \mathbb{Z}_q[x]/(\Phi(x))$ and e_i from a Gaussian distribution χ, such that $b_i = a_i s + e_i$ for $i \in \mathbb{N}$, then the pairs (a_i, b_i) are *computationally* indistinguishable from (a_i, b'_i), where both a_i, b'_i are uniformly sampled from R_q. Here $\Phi(x) = x^n + 1$ is a cyclotomic polynomial and n is a power of 2.

A cryptographic system typically consists of three main components: key generation, encryption and decryption. Since we want to support computations over ciphertexts, we have homomorphic operations as well.

1) Key Generation: We focus on the asymmetric scheme, hence we need a secret key and the corresponding public key. For a secret key $SK = s$, we sample its coefficients from χ, denoted by $s \leftarrow \chi$, a random element $a_1 \in R_q$ and an error $e \leftarrow \chi$. We set the public key to be $PK = (a_0, a_1)$, where $a_0 = -(a_1 \cdot s + t \cdot e)$ and t is the modulus of the plaintext space. Note that s, a_0, a_1 and e are all elements of a ring of polynomials.

2) Encryption: Given a plaintext $m \in R_t = \mathbb{Z}_t[x]/(\Phi(x))$ and a public key $PK = (a_0, a_1)$, we construct an encryption function $Enc(m, PK) = (c_0, c_1) = (a_0 u + tg + m, a_1 u + th) \in (R_q)^2$, where u, g and h are noises sampled independently from the Gaussian distribution χ.

3) Decryption: While any fresh encryption will produce a ciphertext with two components $C = (c_0, c_1) \in (R_q)^2$, homomorphic multiplication (described below) will increase the number of elements in the ciphertext beyond two. Hence, we represent the ciphertext as $C = (c_0, \ldots, c_\xi) \in (R_q)^{\xi+1}$. The decryption function is defined as $Dec(C, SK) = \tilde{m} = \sum_{i=0}^{\xi} c_i s^i \in R_q$.

To understand its correctness, we show the following proof for a ciphertext with two elements $C = (c_0, c_1)$:

$$\begin{aligned}\sum_{i=0}^{1} c_i s^i &= (a_0 u + tg + m) + (a_1 u + th)s \\ &= -a_1 us - tue + tg + m + a_1 us + ths \\ &= t(-ue + g + hs) + m.\end{aligned}$$

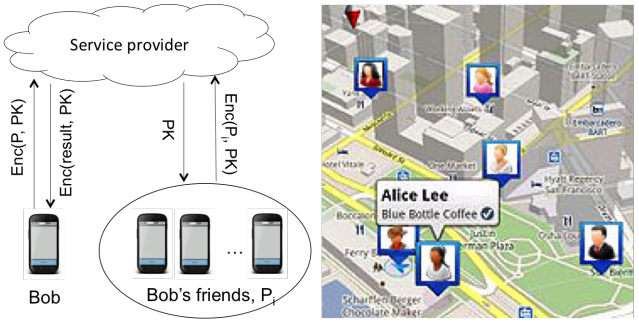


Fig. 1. Privacy preserving geosocial apps.

The coefficients of the resulting expression must be converted from $(0, q]$ to $(-q/2, q/2]$ in order to properly represent the error terms, since they are drawn from Gaussian distribution χ. We should then be able to decrypt the plaintext m by performing a $(mod\ t)$, given that the noise terms are small.

4) Homomorphic Operations: Given two ciphertexts $C = (c_0, \ldots, c_\xi)$ and $C' = (c'_0, \ldots, c'_\eta)$, the homomorphic addition is a straightforward component-wise addition.

$$C + C' = (c_0 + c'_0, \ldots, c_\xi + c'_\eta) \in (R_q)^{max(\xi,\eta)+1},$$

where we might need to pad ciphertexts by 0's in order to match the length of the longer ciphertext.

Homomorphic multiplication is more difficult, because of the growth of elements,

$$C * C' = (\hat{c}_0, \ldots, \hat{c}_{\xi+\eta}).$$

There is a technique, called *relinearization*, to reduce the number of ciphertext elements. In a nutshell, homomorphic multiplication introduces terms with s^i, for $i > 1$. Take the case of multiplying two ciphertexts of length two: $C = (c_0, c_1)$ and $C' = (c'_0, c'_1)$. We want $C * C' = mm' + te_{mult}$ so that we get back mm' $(mod\ t)$ where m and m' are the corresponding messages and e_{mult} is the error resulting from multiplying two ciphertexts. Interested readers can find more information in [10].

III. Secure proximity computation with SWHE

In the current system, we use the NLV2011 [10] scheme to implement all the required HE operations. To illustrate the working of the proposed system, we use the following application scenario, as shown in Fig. 1. Bob wants to compute his proximity from his friends subscribed to the service and uses encryption to hide the exact coordinates from the service provider.

A. Computing the Euclidean distance with UTM coordinates

The first approach is to directly compute the distance between two coordinates. However, GPS coordinates are commonly encoded in the WGS84 form (latitude and longitude, we ignore altitude for clarity), which require complex computations like the *haversine* formula [14]. Implementing these functions as HE operations will significantly increase the computation time, making the system impractical. Instead, we

map the WGS84 coordinates into the Universal Transverse Mercator (UTM) coordinate system so that the Euclidean distance of two UTM points is an approximation of the distance on the Earth surface. The UTM projection is a type of Cartesian coordinate system, which divides the Earth surface into 60 zones, each 6° of longitude in width. Within each zone, (x, y) refers to a point in the projection. Given two UTM points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, we approximate the distance using the Euclidean distance function $d(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. This projection typically works well if the two points are relatively close, but the error increases as the actual distance between the two points increases.

We compute the square of encryption of Euclidean distance $(C_x - C'_x)^2 + (C_y - C'_y)^2$, where all C_i terms are ciphertexts. To speed-up the computation, we apply the Chinese Remainder Theorem (CRT) to breakdown large integers. The result of this HE computation is sent back to the user, who will decrypt it with a private key and apply the square root operation to obtain the approximate distance in meters. Due to the use of the polynomial ring structure, the square-root operation is not supported in SWHE and this operation is applied by users after decryption [15].

B. Approximating distance with geo-hashing

For some applications, we do not need or want precise distance value. This subsection describes a method for answering the "nearby" query, or to compute an appropriate bounding box, which contains the two given points. We can compute the maximum and minimum distance between the two points from the bounding box, without revealing the actual geographical position of the two points.

This approach uses a geo-hashing technique known as *Z-order* curve [16], which can reduce the dimension of data while preserving locality of points. In this case, we are mapping two dimensional coordinates into an array of concatenated indexing keys. Figure 2 shows the first three levels of mapping of geographic regions to the corresponding indexing keys (represented in base 4, hence *quadkey*) in Microsoft's Bing Maps tile system. Every time we increase one level of detail, we divide each bounding box into four equal sub-boxes, with each assigned a new quadkey appended to the existing quadkey string, as illustrated. Essentially, in this linear representation the longer the common prefix between the quadkeys of two points, the closer they are. Also, the longer the indexing key, the more precise reference to the original point. We will discuss how this property can help to improve privacy in Section IV. The details of the transformation steps are described in [17]; for clarity, we focus our discussion on the string matching processes with homomorphic encryption.

Given two GPS coordinates in WGS84 encoding, we use the geo-hashing technique to compute the corresponding quadkeys, and subsequently convert them into a binary representation. As an example, for a GPS point $(48.9225, -78.75)$, the corresponding quadkey and binary key at level 5 are 03023 and 0011001011 respectively. Once the coordinates are transformed into their binary representations, we can apply efficient bitwise operations to compute the common prefix. This common prefix can then be used to compute the bounding box that contains the two points. Because of the property of the geo-hashing technique, the longer the prefix, the closer the two points, and the smaller the bounding box. The highlighted square (containing 130, 131, 132, 133), as shown in Fig. 2, is an example of a bounding box. The maximum distance between any two points in the box is the length of the diagonal of the bounding box. Thus given the common prefix, we can approximately compute how far apart the two points are without giving away their exact locations.

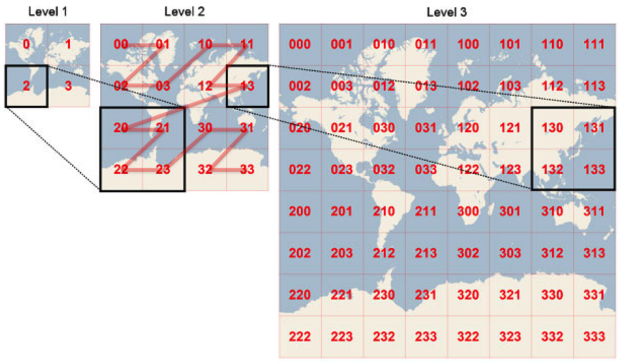


Fig. 2. Z-order curve used in the Microsoft's Bing Maps Tile System.

The main task is to find the common prefix of the two coordinates' binary representations. The logical way is to implement a homomorphic equality operation which compares the encrypted vectors and figures out the matching bits. However, because HE schemes use random noise to mask every encryption, it is difficult to implement this operation on the ciphertexts.

Let us consider the example of Bob and Bob's friend, Alice, when they perform a bitwise encryption on their coordinates, yielding a vector of ciphertexts $\vec{C} = [Enc(A[i], PK)]$ and $\vec{C'} = [Enc(B[i], PK)]$ for i being up to the specified level of detail. To compute the common prefix, we first apply a coordinate-wise XNOR, which is represented as $C_i * C'_i + \bar{C}_i * \bar{C}'_i$ in algebraic form. The coordinate-wise XNOR will return encryption of 1, if the corresponding bit values in the encrypted coordinates are the same, otherwise it will return encryption of 0x. Figure 3 shows in plaintext an example of the bitwise XNOR operation. This yields an intermediate result T.

To extract the common prefix we need a prefix mask $\check{T}$, which has all bits after the first 0 bit reset to 0. To convert from T to $\check{T}$, we introduce a prefix mask refinement process. For a T of n bits, this process involves $n - 1$ homomorphic multiplications $\check{T}[i] = T[i] \times \check{T}[i-1]$, as illustrated in Fig. 3. We do not change the value of the first bit, hence $\check{T}[0] = T[0]$. Due to the use of multiple levels of homomorphic

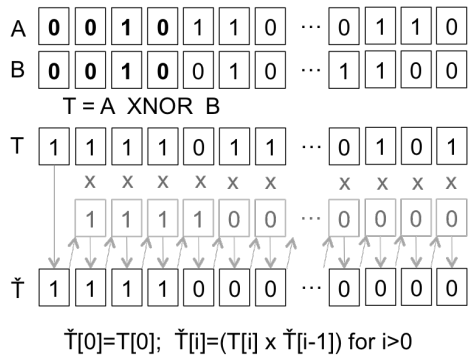


Fig. 3. Technique to extract the plaintext version of prefix mask $\check{T}$.

Device	RaspberryPi	ODROID-XU3	Amazon EC2
CPU	ARM1176JZF-S	Cortex-A15	Intel Xeon E5-2670
Clock	700 MHz single	2.0 Ghz quad	2.5 Ghz single
Memory	512 MB	2 GB	1 GB

Fig. 4. Hardware evaluation platforms.

TABLE I
PARAMETER SETTINGS FOR DIFFERENT EXPERIMENTS.

Parameter	SWHE test	Euclidean distance	Geo-hashing
t	2	[2, 3, 5, 7, ..., 53]	2
n	64	64	64
$\lceil \log_2 q \rceil$	128	128	846
L	1	1	43

Here t is the plaintext space modulus, n is the degree of the polynomial $\Phi(x)$, $\lceil \log_2 q \rceil$ is the bit-length of q and L is the level of multiplications required.

multiplications[1], the relinearization step is required to reduce the size of the ciphertext which increases the computation time. Once we obtain the prefix mask, we can compute the common prefix by single-level coordinate-wise homomorphic multiplications of $\check{T}$ and A or B.

Finally, the common prefix is used to compute the appropriate bounding box (as described in [17]), whose encryption will be returned as result to the user.

IV. Evaluation and Discussion

In this section, we describe the implementation of our SWHE framework, evaluation results and privacy analysis.

A. Implementation and evaluation platforms

The prototypes of the two approaches were implemented using our HE framework. We have implemented three recent SWHE schemes: NLV2011 [10], YASHE [18], and FV [19]. The framework includes implementations of all main functionalities of the individual schemes. These schemes are implemented in C++ with the support of polynomial operations from the Number Theory Library (NTL), which depends on the GNU Multiple Precision Arithmetic Library (GMP) to handle large integers. We verified the correctness of our implementation through extensive validation tests, and we compared our performance results with the data reported in the original papers.

We conducted our performance evaluations on three platforms: Raspberry Pi model B+, ODROID-XU3 (as shown in Fig. 4), and Amazon EC2 Cloud (the basic t2.micro type). All these platforms run Ubuntu 14.01 with standard installation of packages. As shown in Fig. 4, the specifications of these platforms represent three different classes of devices; from low-end smartphones to resource rich cloud environment. The ODROID-XU3 board is especially designed for the Android application development.

B. Evaluation results and discussion

In HE, parameter selection is a key process that determines the correctness, security and performance of the schemes. Table I shows the parameter settings for the three types of experiments, which consists of tests for the common SWHE operations and the two proposed approaches for secure distance computation. The focus of our evaluation is largely on the performance and feasibility.

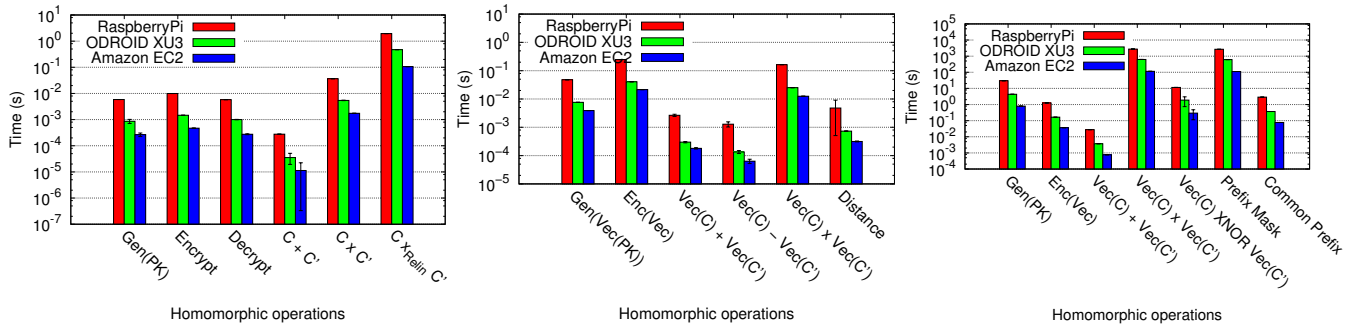
Each experiment was executed 1000 times and a timing framework was used to record and compute the average computation time of each operation as well as the standard error of the mean which was relatively small. Each experiment was also tested on the various platforms previously mentioned. In our application scenario, since most of the homomorphic operations are to be executed in the cloud environment, we mainly focused our attention on the performance of Amazon's EC2. In addition, we can expect better performance if we run our experiments on faster cloud service configurations, rather than the restricted service that was used here. The performance results of the other two platforms are presented to demonstrate that the proposed approaches are feasible on common off-the-shelf mobile devices.

1) Common homomorphic operations: Figure 5(a) shows the computation time of all the common homomorphic operations: keygen, encryption, decryption, addition, multiplication and multiplication with relinearization. As expected, the computation times depend mainly on the CPU clock rate of the respective platform. For the same experiment, Raspberry Pi takes a significantly longer time than the ODROID-XU3 or the Amazon EC2 environment. Note also that the y-axis values are in log-scale.

As shown in Fig. 5, our results are consistent with the expectation that homomorphic multiplication takes more time (even longer if the relinearization step is included) compared to other HE operations. However, it should be noted that even for the Raspberry Pi, the homomorphic multiplication of two ciphertexts with relinearization still takes less than a second.

2) Computing Euclidean distance: Figure 5(b) shows the computation time for each operation involved in the first approach. As expected, the component-wise vector addition and subtraction of ciphertexts take less time than the vector multiplication of ciphertexts. The vector-based operations used in this approach take longer than the basic homomorphic

[1]Product of one homomorphic multiplication is used in another homomorphic multiplication.



(a) Common homomorphic operations (b) Computing Euclidean distance (c) Geo-hashing approach

Fig. 5. Computation times measurement of different homomorphic operations: [Gen(PK)] public key generation; [Gen(Vec(PK))] public key vector generation; [Encrypt] encryption; [Decrypt] decryption; [C + C'] addition; [C x C'] multiplication; [C x_{Relin} C'] multiplication with relinearization; [Enc(Vec)] coordinate-wise vector encryption; [Vec(C) + Vec(C')] vector addition; [Vec(C) x Vec(C')] coordinate-wise vector multiplication; [Distance] Euclidean distance computation; [Vec(C) XNOR Vec(C')] coordinate-wise XNOR operation; [Prefix Mask] prefix mask vector computation; [Common Prefix] identifying the common prefix.

operations. The distance computation operation involves three vector-based homomorphic operations: subtraction, followed by a multiplication with relinearization and an addition, and results in the square value of the Euclidean distance. The square root of this value is computed after decryption. The overall computation time of the Euclidean distance is 0.21s in the Amazon's EC2 platform.

3) Approach using geo-hashing technique: As shown in Fig. 5(c), this approach has a longer computation time than the first approach because of the complexity involved in the computations. All the operations for this techniques take a longer time than the previous approach and the basic homomorphic operations because of the larger q value and the vector computation. The computation of the prefix mask is the most time consuming process in this approach due to the fact that it uses 43 levels of homomorphic multiplication with relinearization. The common prefix computation uses one level of multiplication and the computation time is approximately the same as that of coordinate-wise XNOR operation. The overall computation time for the geo-hashing approach is 232.36s in Amazon's EC2 platform. This is long, but can be significant improved by leveraging parallel processing techniques. We investigate the speed-up techniques as future work.

C. Privacy analysis and extension

The security of the system depends on the underlying HE scheme, but the privacy also depends on the system design. In the first approach, the accuracy can be improved by implementing the *haversine* formula using homomorphic operations which can provide the actual distance from the starting point. The privacy threat is that if we know the exact distance between two points, using one point as the center, the other point must be on the circle. The adversary can use heuristics, such as map information, to eliminate unlikely places to infer the exact location.

In contrast, the final result of the second approach is the top-left coordinate of the corresponding bounding box. Based on the geo-hashing technique, we know the two points must be within two distinct sub-areas of the given bounding box, but no other information is exposed. For some applications where an approximate distance is sufficient, this approach provides better privacy. In addition, the computation of the bounding box is based on the common prefix. Since we formulate the problem of distance computation as string matching, the system can handle string patterns that differ in length. Thus, we can introduce a privacy control feature with zero overhead where users decide the level of detail to be revealed to other users, which is protected by homomorphic encryption. When a query is received requesting the user's coordinates, the user can adjust the level of detail to be released by masking the binary vector before encryption depending on the relationship with the requester. For example, only the city for social encounters, but exact location for family.

V. Related work

There are different types of privacy preserving mechanisms [2], including access control, obfuscation, anonymity, cryptography and differential privacy.

We see that our second approach is closely related to the idea of spatial cloaking, which is a technique for transforming the user's exact location to a cloaked area (i.e., rectangle or circle). In [13], Hashem and Kulik proposed the idea of computing cloaked area by coordinating with nearby peers over the wireless ad-hoc networks. This distributed approach leverages the idea of crowdsourcing to compute a cloaked area, and it depends greatly on the availability of participants within close vicinity. Our solutions make use of the increasingly accessible cloud computing resource, thus making it more practical. Khoshgozaran and Shahabi [20] proposed the use of Hilbert curves (similar to our Z-order curve) and a one-way trapdoor function to transform the user's location into a cloaked area, which contains places-of-interest (POIs) related to the user's query. This solution computes the cloaked area that contains the user's location and a precomputed set of POIs stored in a look-up table. Our second approach is also a type of spatial cloaking, but the use of SWHE scheme ensures that the private data is only known to the user.

When considering private data release, there is also differential privacy [21], which introduces small controlled noises (typically from *Laplace* distribution) to make individual contributions to the dataset indistinguishable. For example, Mir et al. [22] applied differential privacy to privatize the human mobility model that was proposed in [23]. However, due to the noise that is introduced with each computation, the number of computations that one can perform on the synthesized data is limited. Also, unless we only maintain the synthesized data in storage, security breaches can be any company's worst nightmare [4]; such incidents have been reported frequently. In this regard, homomorphic encryption offers extra protection against an insider attack.

While performing arbitrary computation with FHE schemes is still far from practical [10], [24], many applications based on SWHE schemes have demonstrated their practicality in the field of biometric authentication [25], medical/genomic data analysis [15], [26], and private information retrieval [27], [4]. This paper shows how new SWHE schemes can be used to support secure proximity computations.

VI. Conclusion and future work

In pervasive and mobile computing, location data is one of the most commonly used, but sensitive, contextual information that drives many adaptive applications. Despite the potential benefit, the use of location data is limited, due to privacy concerns. In this paper, we explored a cryptographic approach to enable privacy-preserving computation of location data. Using distance computation as an example, we proposed methods to compute distance and to perform spatial cloaking over encrypted coordinate data. Security is guaranteed by the underlying hardness problem in homomorphic encryption. We prototyped the two approaches using a recent somewhat homomorphic encryption scheme. We conducted performance evaluations on mobile devices with different specifications. We argue that our work demonstrated a fully working system for supporting secure computation of location data.

As a future work, we would like to make a number of optimizations to improve performance: (i) incorporating Single-Instruction-Multiple-Data (SIMD) support in both approaches to parallelize the computation, (ii) exploring different techniques to compute the prefix mask in the geo-hashing approach and to avoid the use of relinearization after each homomorphic multiplication, and (iii) incorporating the homomorphic encryption schemes that support multiple keys.

References

[1] 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.

[2] C. Bettini and D. Riboni, "Privacy protection in pervasive systems: State of the art and technical challenges," *Pervasive and Mobile Computing*, vol. 17, Part B, pp. 159 – 174, 2015, 10 years of Pervasive Computing' In Honor of Chatschik Bisdikian.

[3] Cisco, Inc., "Cisco visual networking index: Global mobile data traffic forecast update, 2014–2019," May 2015.

[4] R. A. Popa and N. Zeldovich, "How to compute with data you can't see," IEEE Spectrum, July 2015.

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

[6] ATOCKAR, "Riding with the stars: Passenger privacy in the nyc taxicab dataset," Sept 2014. [Online]. Available: http://research.neustar.biz/2014/09/15/riding-with-the-stars-passenger-privacy-in-the-nyc-taxicab-dataset/

[7] A. Beresford and F. Stajano, "Location privacy in pervasive computing," *Pervasive Computing, IEEE*, vol. 2, no. 1, pp. 46–55, Jan 2003.

[8] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[9] D. Micciancio, "A first glimpse of cryptography's holy grail," *Commun. ACM*, vol. 53, no. 3, pp. 96–96, Mar. 2010.

[10] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of CCSW '11*, Chicago, IL, USA, 2011, pp. 113–124.

[11] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," in *Proceedings of FOCS '11*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 97–106.

[12] ——, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Proceedings of CRYPTO'11*. Santa Barbara, CA: Springer-Verlag, 2011, pp. 505–524.

[13] T. Hashem and L. Kulik, ""don't trust anyone": Privacy protection for location-based services," *Pervasive and Mobile Computing*, vol. 7, no. 1, pp. 44 – 59, 2011.

[14] MTL, "Calculate distance, bearing and more between latitude/longitude points," http://www.movable-type.co.uk/scripts/latlong.html.

[15] J. W. Bos, K. Lauter, and M. Naehrig, "Private predictive analysis on encrypted medical data," Tech. Rep. MSR-TR-2013-81, September 2013.

[16] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Comput. Surv.*, vol. 30, no. 2, pp. 170–231, Jun. 1998.

[17] J. Schwartz, "Bing maps tile system," https://msdn.microsoft.com/en-us/library/bb259689.aspx.

[18] J. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *Cryptography and Coding*, ser. LNCS, M. Stam, Ed. Springer Berlin Heidelberg, 2013, vol. 8308, pp. 45–64.

[19] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," https://eprint.iacr.org/2012/144/20120322:031216, March 2012.

[20] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *Proceedings of SSTD'07*. Boston, MA, USA: Springer-Verlag, 2007, pp. 239–257.

[21] C. Dwork, "Differential privacy," in *Proceeding of ICALP2006*, ser. LNCS, vol. 4052. Venice, Italy: Springer Verlag, July 2006, pp. 1–12.

[22] D. Mir, S. Isaacman, R. Caceres, M. Martonosi, and R. Wright, "Dp-where: Differentially private modeling of human mobility," in *Proceedings of BigData '13*. Santa Clara, CA, USA: IEEE, Oct 2013, pp. 580–588.

[23] S. Isaacman, R. Becker, R. Cáceres, M. Martonosi, J. Rowland, A. Varshavsky, and W. Willinger, "Human mobility modeling at metropolitan scales," in *Proceedings of MobiSys '12*. Low Wood Bay, Lake District, UK: ACM, 2012, pp. 239–252.

[24] T. Lepoint and M. Naehrig, "A comparison of the homomorphic encryption schemes fv and yashe," in *IACR Cryptology ePrint Archive, 2014:062*, 2014.

[25] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, "Secure pattern matching using somewhat homomorphic encryption," in *In Proceedings of CCSW '13*. Berlin, Germany: ACM, 2013, pp. 65–76.

[26] J. H. Cheon, M. Kim, and K. Lauter, "Homomorphic computation of edit distance," in *Workshop on Encrypted Computing and Applied Homomorphic Cryptography*. Isla Verde, Puerto Rico: ACM, January 2015.

[27] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, "Single-database private information retrieval from fully homomorphic encryption," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 5, pp. 1125–1134, 2013.