POSTER: Location Privacy Using Homomorphic Encryption

Peizhao Hu(✉) and Siyu Zhu

Department of Computer Science, Rochester Institute of Technology, Rochester, USA
ph@cs.rit.edu

Users' location data has become important contextual information that is used by many popular geosocial applications (such as Facebook) to notify users when a friend is within specified vicinity, to recommend like-minded users who are within a given geographic proximity, or to deliver targeted ads. While service providers want "noise-free" location data to enable value-added social features or targeted ads, and need solutions that offer provable data security, users want robust control during or after releasing location data. There are various categories of techniques in preserving location privacy [13], but many existing solutions fall short in achieving the needs of robust and dynamic control from the user and preserving high data granularity [1]. As an example, one type of solutions focuses on masking users' coordinates using *spatial cloaking* [6], anonymity or obfuscation. These techniques typically require a trusted server that will see all location data. Also, since data is not encrypted these solutions fail to protect users' private data in situations like security breaches or insider attacks, which are more than just occasional incidents [11].

This paper investigates solutions that (i) meet the needs of users and service providers discussed above, (ii) can benefit from the trend of computation outsourcing to support resource constrained mobile devices, and (iii) offer measurable security protection to users' private data. Specifically, we investigate a mix of spatial cloaking that offers flexible control of location masking, and Homomorphic Encryption (HE) that provides the ability for service providers to perform computations on encrypted location data without decryption [5]. Many HE schemes base their security on the *Learning-With-Errors* (LWE) problems that can be reduced to the *average-case* lattice-based cryptography problems, such as *Shortest Vector Problem* (SVP) [12]. A quantum reduction from LWE to standard SVP problem was constructed [12], which suggests that it is unlikely that we can efficiently solve the LWE problem in polynomial time [8,10]. This property offers promising post-quantum cryptographic assurance to users' location data. Regarding the encryption scheme, our prototype implementation used the NLV2011 [9] Somewhat HE (SWHE) scheme. NLV2011 is a practical construction of a well studied SWHE scheme—BV2011 [2]. An SWHE scheme supports a limited number of computations before the ciphertexts become too noisy for decryption.

To compute proximity information securely, the common first attempt is to construct homomorphic operations to evaluate a distance function, such as

R. Deng et al. (Eds.): SecureComm 2016, LNICST 198, pp. 758–761, 2017.
DOI: 10.1007/978-3-319-59608-2_45

Euclidean distance. Due to the complexity of implementing the *haversine* function in HE schemes, our earlier attempt [7] focused on projecting the WGS84 coordinates onto the Cartesian coordinates, such as the Universal Transverse Mercator coordinate system, and developing homomorphic functions to calculate the Euclidean distance. However, the transformation between coordinate systems introduce undesirable noise, which increases as two points become further apart. Thus, this approach limits user's ability to control the granularity of location data. Also, given a point if we can calculate the extract distance to another point, then we know the other point must be on circumference. By triangulation technique, we can easily determine the location of a point. Hence, this approach leads to serious privacy issues. Instead, we focus on the idea of spatial cloaking, which dynamically compute an appropriate region that encloses the two given points. The size of this region (or level of details) can be controlled by the user. Due to space limitation, two approaches for homomorphic proximity computation will be discussed below. Interested readers can refer to additional papers[1] [7].

Spatial Cloaking using User Preferences. Usually, spatial cloaking solutions mask coordinates with multidimensional data access methods [4], such as *Z-order* or *Hilbert* curve. These geohashing techniques reduce the dimensionality of coordinate data while preserving locality of points. We used the *Z-order* curve to geo-hash the two dimensional coordinates into an array of concatenated indexing keys. Each key represents the position of the point at a particular level of detail. The indexing keys are represented in base 4, hence *quad-key*. Every time we increase one level of detail, we divide each bounding box into four equal sub-boxes, with each assigned a new quad-key appended to the existing quad-key string, as illustrated. Essentially, in this representation the longer the common prefix between the quad-keys of two points, the closer they are. Also, the longer the indexing key provides a more precise reference to the original coordinates.

Given GPS coordinates in WGS84 encoding, we can compute quad-keys. As an example, given the coordinates (43.584474, –77.675472) the quad-keys at level 5 is 03023. We transform the quad-keys into binary-keys, as 0011001011. Due to space constraints, we only show the indexing keys at a limited level. The maximum level of detail in the form of quad-key is 22 (or 44 level in binary-keys), which corresponds to the exact GPS coordinates. Ideally, users will share the complete quad-keys with a server and update it periodically, rather than sending different versions of the masked keys to reduce communication overhead. Users specify the privacy preferences as a list of prefix masks to control the granularity of location data depending on the friendship with another user. By applying different masks, the server can generate different masked areas for the different requesters; hence achieving the level of privacy as desired by users. However, because we can invert the binary-keys to coordinates, a trusted server is required.

To avoid using a trusted server, we encrypt both the binary-keys and the masks, and construct homomorphic operations to select an appropriate mask

[1] http://cs.rit.edu/~ph/research.

and apply it to the binary-keys. Using homomorphic encryption, users can share periodic updates of their coordinates in full level of detail while preserving privacy. We encrypt each element in the binary-keys as a separate ciphertext. Hence, the encryption of Alice's binary-keys yields a vector of ciphertexts. This is common in many existing works [3] to simplify the prototype implementation. When a user B or a third-party application requests another user A's location, the server performs coordinate-wise homomorphic multiplications using the appropriate masks, $Enc(A) \otimes Enc(M_B)$ or $Enc(A) \otimes Enc(M_{\#})$, yielding the encrypted results that are sent back to the corresponding information requesters. In this cases, since we only need homomorphic multiplications with depth one, no noise reduction step is needed; hence this approach should be relatively efficient, as shown in our results.

Computing Common Prefix of Two Geo-hashing Codes. With homomorphic encryption we can achieve spatial cloaking with untrusted servers. Extending from the simple masking operation, we explore the possibility of computing the appropriate proximity information in the form of a common prefix (CP), if given $Enc(A)$ and $Enc(B)$. The server will apply the corresponding masks as defined in user's preference before computing the common prefix, because individual users' privacy preferences have higher priority. To simplify the discussion, we assume users are happy with using the location data in full level of detail to compute the common prefix. The resulting encrypted common prefix can then be used to compute a bounding box that contains the location of Alice and Bob. Since the maximum distance between any two points in the bounding box is the length of the diagonal, we know the upper bound of how far apart the two points are without giving away their exact locations. In addition, we can use this property to hide users' mobility trajectories within an area; hence, this approach does not only preserve the privacy of coordinates but also the mobility patterns of users.

To find the common prefix, we apply homomorphic operations to compute a common prefix mask of two binary-keys. The logical way is to implement a homomorphic equality operator which compares the encrypted vectors and figures out the matching bits. However, because HE schemes are nondeterministic due to the use of random noise in every encryption, different encryptions of the same value generate ciphertexts that are different. It is difficult to implement an equality operator that compares the ciphertexts. In literature, the equality operator [3] for vectors $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_n)$ is implemented as arithmetic circuits, $EQU(X, Y) = \wedge_{i=1}^{n}(1 \oplus x_i \oplus y_i)$, where $\wedge$ and $\oplus$ are bitwise AND and XOR. However, this operation can only tell whether the two vectors are identical but cannot dynamically compute the common prefix with an appropriate level of detail. In this regard, this approach is similar to the work on private proximity test [13].

In this paper, we construct a special arithmetic circuit to compute the common prefix. Given two encrypted binary vectors A and B, we first apply a coordinate-wise XNOR, which returns encryption of 1, if the corresponding bit values in the encrypted coordinates are the same, otherwise it will return encryption of 0. We then perform a prefix mask *purification* step in which bit value

after the left most 0 is reset to 0. This process requires consecutive homomorphic multiplications which increase the multiplicative depth. Due to the use of multiple levels of homomorphic multiplications, the relinearization step is required to reduce the size of the ciphertext. We study the characteristics of this algorithm and compare it to other homomorphic operations, such as the equality operator *EQU* [3]. We found that they share similar computation time profile because they both rely on consecutive homomorphic multiplications.

References

1. Bettini, C., Riboni, D.: Privacy protection in pervasive systems: state of the art and technical challenges. Pervasive Mob. Comput. Part B **17**, 159–174 (2015)
2. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (Standard) LWE. In: Proceedings of FOCS 2011, pp. 97–106. IEEE Computer Society, Washington, DC, USA (2011)
3. Cheon, J.H., Kim, M., Lauter, K.: Homomorphic computation of edit distance. In: Workshop on Encrypted Computing and Applied Homomorphic Cryptography, Isla Verde, Puerto Rico, ACM, January 2015
4. Gaede, V., Günther, O.: Multidimensional access methods. ACM Comput. Surv. **30**(2), 170–231 (1998)
5. Gentry, C.: Computing arbitrary functions of encrypted data. Commun. ACM **53**(3), 97–105 (2010)
6. Hashem, T., Kulik, L.: "Don't trust anyone": privacy protection for location-based services. Pervasive Mob. Comput. **7**(1), 44–59 (2011)
7. Hu, P., Mukherjee, T., Valliappan, A., Radziszowski, S.: Homomorphic proximity computation in geosocial networks. In: Proceeding of BigSecurity 2016, An INFOCOM 2016 Workshop, San Francisco, CA, USA, April 2016
8. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009)
9. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of CCSW 2011, pp. 113–124, Chicago, IL, USA (2011)
10. Peikert, C.: A decade of lattice cryptography. IACR Cryptology ePrint Archive (2015/939), p. 939 (2015)
11. Popa, R.A., Zeldovich, N.: How to compute with data you can't see. In: IEEE Spectrum, July 2015
12. Regev, O.: The learning with errors problem (invited survey). In: Proceedings of CCC 2010, pp. 191–204, Cambridge, MA, June 2010
13. Saldamli, G., Chow, R., Jin, H., Knijnenburg, B.: Private proximity testing with an untrusted server. In: Proceedings of WiSec 2013, Budapest, Hungary, ACM, April 2013