Secure Mutual Proximity Zone Enclosure Evaluation

Sunoh Choi
Purdue University
West Lafayette, IN
choi39@purdue.edu

Gabriel Ghinita
Univ. of Massachusetts
Boston, MA
Gabriel.Ghinita@umb.edu

Elisa Bertino
Purdue University
West Lafayette, IN
bertino@cs.purdue.edu

ABSTRACT

Mobile users engage in novel and exciting location-based social media applications (e.g., geosocial networks, spatial crowdsourcing) in which they interact with other users situated in their proximity. In several application scenarios, users define their own *proximity zones* of interest (typically in the form of polygonal regions, such as a collection of city blocks), and want to find other users with whom they are in a mutual enclosure relationship with respect to their respective proximity zones. This boils down to evaluating two point-in-polygon enclosure conditions, which is easy to achieve for revealed user locations and proximity zones. However, users may be reluctant to share their whereabouts with their friends and with social media service providers, as location data can help one infer sensitive details such as an individual's health status, financial situation or lifestyle choices. In this paper, we propose a mechanism that allows users to securely evaluate mutual proximity zone enclosure on encrypted location data. Our solution uses homomorphic encryption, and supports convex polygonal proximity zones. We provide a security analysis of the proposed solution, we investigate performance optimizations, and we show experimentally that our approach scales well for datasets of millions of users.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Security, Experimentation

Keywords

Location Privacy, Homomorphic Encryption

1. INTRODUCTION

An increasing number of services that provide a geo-spatial dimension to user interaction are surfacing in today's online landscape. These range from scenarios such as snapshot queries sent to map services (e.g., GoogleMaps) to more complex types of interaction where users report their history of movement in return for personalized services, location-centric recommendations, etc. Most of these applications also involve a social media component, where users interact with social network friends based on their geographical proximity.

SIGSPATIAL'14, November 04 - 07 2014, Dallas/Fort Worth, TX, USA

http://dx.doi.org/10.1145/2666310.2666384.

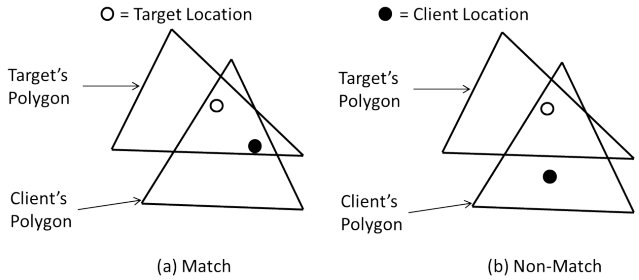

Figure 1: Mutual Proximity Zone Enclosure Detection

Location-based social networks (LBSN) may allow complex location-based interaction among users. However, many such providers are not trustworthy, and loose terms of service agreements allow them to share location data with various third parties for purposes that are often not in the best interest of users. Personal location data may allow an adversary to stage attacks ranging from physical assault and stalking, to inferring sensitive information about an individual's health status, financial situation or lifestyle choices. It is thus necessary to build a secure framework for sharing and processing location data, and cryptographic approaches are a promising direction to achieve this aim [8, 10, 27, 21].

Previous work [8, 27] has addressed effectively the scenario where a client finds securely the result to a query for a nearby point (e.g., nearest-neighbor queries). However, in practice, more complex types of interaction are necessary. In a typical scenario, each user may want to establish a *proximity zone* of interest, e.g., the down-town area of a city, or a region comprising of several city blocks. In this context, users are eager to find friends with whom they are *mutually* situated within each other's interest zones.

Each user will have as personal data two objects: a point location (current user location), and a proximity zone, in the form of a convex polygonal region. Users encrypt this information, and upload it to the LBSN *service provider (SP)*, e.g., Foursquare. At different times, the same user can act in two roles: as a query initiator, in which case the user is referred to as *client*, or as a *target user* (or simply, target)

of another user's *proximity zone query*. Figure 1 illustrates two cases where a particular target represents a match or a non-match for a given client query.

Users typically organize in groups, and they can share encryption keys within a group, which can be distributed using a different channel than the social network (e.g., through a secure connection established directly through their cell phone provider, encrypted email, etc.). The challenge is to design cryptographic techniques that allow users to evaluate securely proximity zone queries, i.e., determine whether two users are mutually enclosed within each other's proximity zones. This condition should be evaluated by the SP at the request of the client, but without requiring the targets to be directly involved in the protocol (i.e., in an "offline" setting). Due to the latter requirement, existing *interactive* techniques for secure polygon enclosure evaluation [2, 19] are not suitable.

Furthermore, users may not fully trust their friends, and even if they do, they may not want friends to always know where they are. Instead, only the outcome of the mutual proximity zone enclosure evaluation should be disclosed by the protocol, and no other information about their locations or zones, other than what can be derived from the evaluation outcome.

In this paper, we address the problem of secure mutual proximity zone enclosure evaluation. Specifically:

- We formulate the problem of secure mutual proximity zone enclosure evaluation, and we introduce a framework for solving it using homomorphic encryption and order-preserving encryption.
- We propose two protcols: *Client-in-Target-Zone (CTZ)* that allows a client to securely determine when her location is enclosed in the proximity zone of a target, and *Target-in-Client-Zone (TCZ)* through which the client learns if the target's location is within the client's proximity zone.
- We provide performance optimizations that allow the SP to filter targets situated far from the client's location, and thus reduce the amount of necessary CTZ and TCZ evaluation rounds.
- We provide a security analysis of the proposed scheme, and we conduct an extensive experimental evaluation which shows that our proposed technique scales well to datasets of millions of users.

The remainder of the paper is organized as follows: Section 2 provides necessary background on the system model and the building-block cryptographic primitives used. Section 3 introduces the proposed protocols for secure mutual proximity evaluation. Sections 4 provides the security analysis of our protocols. Section 5 presents the experimental evaluation results, followed by a survey of related work in Section 6 and concluding remarks in Section 7.

2. PRELIMINARIES

2.1 Cryptographic Building Blocks

Advanced Encryption Standard (AES) [1] is the most widely used symmetric cryptography technique. It supports only two operations: encryption (E_A) and decryption (D_A), both requiring knowledge of the same secret key.

The **Paillier cryptosystem** is an asymmetric *additive homomorphic* encryption scheme. Encryption, denoted by E_P, requires knowledge of the *public key* only, whereas decryption D_P requires the *private* key. The additive homomorphic property allows computation of the ciphertext of a sum of plaintexts directly from ciphertexts of individual terms, without decryption, and only with knowledge of E_P:

$$E_P(m_1 + m_2) = E_P(m_1) \times E_P(m_2)$$

In addition, it is possible to perform multiplication with a scalar value v under the ciphertext:

$$E_P(m)^v = E_P(m \times v)$$

Paillier encryption is *semantically secure*, i.e., an adversary who intercepts two ciphertexts cannot derive any relationship among the respective plaintexts. In particular, the same plaintext encrypted twice will result in different ciphertexts.

Mutable Order-Preserving Encryption (mOPE) [23] is a novel technique that allows secure and efficient comparison between numbers. The mOPE scheme in a client-SP setting works as follows: the client has the secret key of a symmetric cryptographic scheme, e.g., AES, and wants to store the dataset of ciphertexts at the SP in increasing order of corresponding plaintexts. The client and the SP engage in a protocol that builds a B-tree. The SP only sees the AES ciphertexts, but is guided by the client in building the tree structure. The algorithm starts with the client storing the first value, which becomes the tree root. Every new value stored at the SP is accompanied by an insertion in the B-tree.

The SP maintains a mOPE table that stores for each value its AES-encrypted label, as well as an *encoding* of the path from the tree root to the label of that value. The SP uses the encoding to evaluate order relationships among values. The mOPE tree structure is kept balanced due to the use of B-trees. The height of the tree is low, thus all search operations are efficient. To ensure the balanced property, when insertions are performed, it may be necessary to change the encoding of certain ciphertexts. Typically, mutability can be done very efficiently, and the complexity of the operation (i.e., the maximum number of affected values in the tree) is logarithmic in the number of stored values. As shown in [23], mOPE satisfies *IND-OCPA*, i.e., indistinguishability under ordered chosen-plaintext attack. The scheme does not leak anything besides order of values.

ElGamal cryptosystem. ElGamal [4] is a *multiplicative homomorphic* asymmetric encryption system that allows computation of encrypted products under the ciphertext:

$$E_G(m_1 \times m_2) = E_G(m_1) \times E_G(m_2)$$

ElGamal is not directly used in our method, but it is a building block for the GT protocol described next.

GT Protocol. The secure comparison *greater-than (GT)* protocol was proposed in [14] and it allows two parties Alice and Bob holding numbers x and y to determine which one is greater between the numbers, without revealing any information except for the comparison outcome. For an n-bit integer s, define the 0-encoding S_s^0 and the 1-encoding S_s^1 of s as two sets of binary strings, as follows:

$$S_s^0 = \{s_n s_{n-1} ... s_{i+1} 1 | s_i = 0, 1 \leq i \leq n\}$$
$$S_s^1 = \{s_n s_{n-1} ... s_i | s_i = 1, 1 \leq i \leq n\}$$

S_s^0 and S_s^1 contain a number of strings equal to the number of 0 and 1 bits in the representation of s, respectively. The GT protocol reduces integer comparison to the set intersection

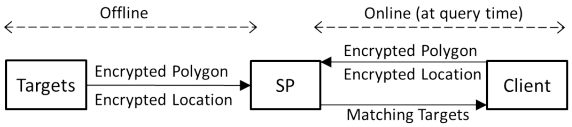

Figure 2: System Model

problem. Given values x and y, $x > y$ if and only if S_x^1 and S_y^0 have a common element. For example, let $x = 6 = 110_2$ and $y = 2 = 010_2$. Then S_x^1={1,11} and S_y^0={1,011}. Since $S_x^1 \cap S_y^0 = \{1\}$, it results that $x > y$. On the other hand, if $x = 2 = 010_2$ and $y = 6 = 110_2$, we have S_x^1={01} and S_y^0={111}. Since $S_x^1 \cap S_y^0 = \emptyset$, then $x \leq y$.

Alice, who holds $x = x_n x_{n-1}...x_1$ prepares a $2 \times n$ table $T[i,j], i \in \{0,1\}, 1 \leq j \leq n$, where $T[x_j, j] = E_G(1)$ and $T[\overline{x_j}, j] = E_G(r_j)$ for some random r_j. Bob, who holds y, receives the table from Alice and computes for each string $t = t_n t_{n-1}...t_i \in S_y^0$ the ciphertext product $c_t = T[t_n, n] \times T[t_{n-1}, n-1] \times ... \times T[t_i, i]$. All ciphertexts are sent to Alice who decrypts them and determines by the properties of ElGamal encryption that S_x^1 and S_y^0 have a common element if and only if one of the ciphertexts is the encryption of 1.

For example, consider $n = 3$, $x = 6 = 110_2$ and $y = 2 = 010_2$. Alice computes a 2×3 table $T[i,j]$

$$T = \{\{E_G(1), E_G(r_1), E_G(r_2)\}, \{E_G(r_0), E_G(1), E_G(1)\}\}$$

Next, Bob computes c_t for each string in its set $S_y^0 = \{1, 011\}$:

$$c_1 = T[1,3] = E_G(1)$$
$$c_{011} = T[0,3] \times T[1,2] \times T[1,1] = E_G(r_2) \times E_G(1) \times E_G(r_0)$$

When Alice decrypts the ciphertexts, she obtains 1 for the first ciphertext and determines that $x > y$.

2.2 System Model

Our system model, illustrated in Figure 2, comprises of three parties: *targets*, *client* (or querying user), and *service provider (SP)*. Note that, the same user can act as client or as target at different times. Targets periodically update their location and proximity zone with the SP. The SP is a location-centric service, e.g., the Foursquare LBSN.

Our goal is to enable the client to securely find targets with whom she is in a mutual proximity relationship. We must protect both the client's and the targets' locations and proximity zones privacy from the SP. In addition, we must protect each target's location and proximity zone from the client, i.e., the client must only be able to learn which targets satisfy the mutual proximity condition (and whatever information can be inferred from this result), but no target's location or proximity zone must be disclosed to the client.

The problem is challenging because the SP needs to check the proximity condition on encrypted data. Existing solutions such as [7, 2, 15, 19, 3] do allow secure polygon enclosure evaluation, but they are interactive in nature, hence all parties must actively participate in each query. This is not realistic in practice, as targets cannot participate in intensive computations all the time. We assume an *offline*, or asynchronous model, where the targets do not directly participate in mutual proximity detection.

Symbol	Definition
E_A,D_A	Encryption/Decryption using AES
E_P,D_P	Encryption/Decryption using Paillier
E_G,D_G	Encryption/Decryption using ElGamal
E_M,D_M	Encryption/Decryption using mOPE
GT	Secure integer comparison protocol [14]
CTZ	Client-in-Target-Zone Detection Protocol
TCZ	Target-in-Client-Zone Detection Protocol

Table 1: Summary of Notations

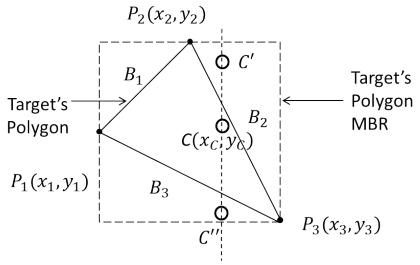

Figure 3: Polygon Enclosure Evaluation

The SP holds a (private) Paillier decryption key, and the corresponding public key is available to all users (i.e., targets and client). Paillier encryption is used to protect targets' data against the client, while at the same time allowing proximity test computations on ciphertexts. The SP is able to perform both E_P and D_P operations, whereas the targets and client only E_P. All users (i.e., targets and client) share a secret AES key, used to encrypt data stored at the SP, hence they can perform both E_A and D_A operations. AES encryption is used to protect the client and targets against the SP. The users also share a mOPE key to encrypt their coordinates, such that the SP is enabled to perform inequality tests on encrypted data. In addition, each client creates a public/private ElGamal key pair used to run the GT protocol with the SP. The client is able to perform E_G and D_G, whereas the SP can only perform E_G. Also, we note that all the discussed encryption functions assume a plaintext space of integers. However, this space is typically very large, and can accommodate very large fractional numbers (i.e., coordinates) converted to integers, with high precision, as in [7, 19].

We assume that the SP is honest but curious. The SP does not tamper with the location data received from the targets, it does not drop any messages, and runs the proximity detection protocol as designed. However, in addition to correctly executing the protocol, it attempts to learn the locations and proximity zones of the targets and the client. Also, we assume that there is no collusion between the SP and users. Table 1 summarizes the notations used.

3. SECURE MUTUAL PROXIMITY ZONE ENCLOSURE

In Section 3.1, we present a secure protocol to detect client enclosure in the target's zone (CTZ), followed in Section 3.2 by the dual protocol to test target enclosure in the client's zone (TCZ). In Section 3.3, we propose optimizations to reduce the number of CTZ and TCZ invocations. We summarize the complete approach in Section 3.4.

3.1 Client-in-Target-Zone (CTZ) Enclosure

Consider the example in Figure 3, where the target's zone is the polygon $P_1P_2P_3$ with edges B_1, B_2, B_3, and $C(x_C, y_C)$ is the client's location. Assume that C is inside the target

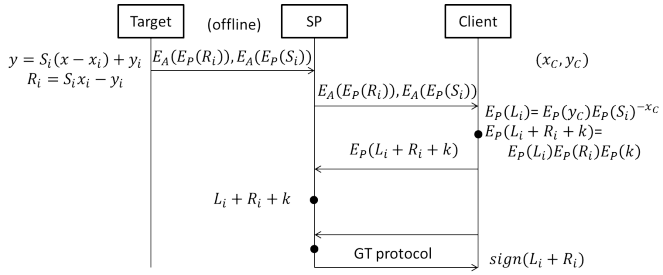

Figure 4: CTZ: placement relative to one line

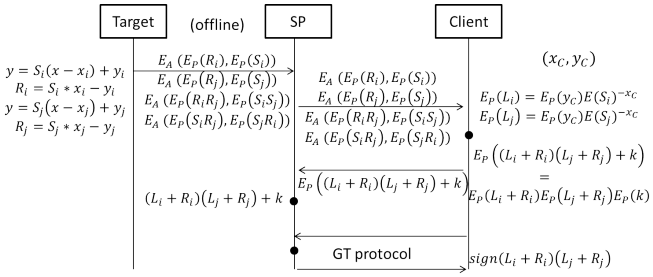

Figure 5: CTZ: placement relative to two lines

zone's minimum bounding rectangle (MBR)[1]. To find if the zone encloses the client, one can draw a vertical line through point C, and determine the two polygon edges that intersect the line, in this example B_2 and B_3 (for convex polygons, it is guaranteed that there will be two intersecting edges). The client is enclosed in the zone if C is above one of the edges (B_3 in the figure) and below the other (B_2). Any other relative placement (e.g., points C' and C'' in the diagram) means that the client is outside the target's zone. Next, we show how to securely perform these steps.

First, we show how to determine the placement of a point relative to a single line. For the line segment of the i^{th} polygon edge that starts at point $P_i(x_i, y_i)$ and has slope S_i, the line equation is:

$$y = S_i(x - x_i) + y_i$$

which can be also written as

$$y - S_i * x = -S_i * x_i + y_i$$

We denote by L_i and $-R_i$ the left and right hand of the equation, respectively:

$$L_i = y - S_i * x = -(S_i * x_i - y_i) = -R_i \quad (1)$$

If we plug in the client's coordinates in Eq. (1), the right-hand side R_i has a fixed value (independent of C), whereas the value of L_i depends on C. Each target will send to the SP the value of R_i for all its zone edges, doubly-encrypted first with Paillier, and then with AES encryption:

$$E_A(E_P(R_i)) = E_A(E_P(S_i * x_i - y_i))$$

as well as $E_A(E_P(S_i))$. These two items are periodically uploaded to the SP with every location update.

Upon receiving a query, the SP sends both encrypted items to the client who decrypts them using D_A, and then computes using the homomorphic property of Paillier encryption the encrypted left-hand side of Eq. (1) as follows[2]:

$$E_P(L_i) = E_P(y_C) \times E_P(S_i)^{-x_C} = E_P(y_C - S_i * x_C) \quad (2)$$

Furthermore, the client computes

$$E_P(L_i + R_i) = E_P(L_i) \times E_P(R_i) = E_P(y_C - (S_i * (x_C - x_i) + y_i))$$

If $L_i + R_i > 0$, it signifies that the client's point is above the line, otherwise it is below or on the line. Thus, only the sign of $L_i + R_i$ is necessary for evaluation. However, should the client send $E_P(L_i + R_i)$ to the SP, the latter may infer additional details about the target's location. To prevent such disclosure, we employ *additive blinding*, whereby the client selects a random number k and computes:

$$E_P(L_i + R_i) \times E_P(k) = E_P(L_i + R_i + k)$$

The blinded value is sent to SP, which decrypts it and obtains $L_i + R_i + k$. Next, the client and the SP execute the GT protocol [14], in order to learn which value is greater between k and $L_i + R_i + k$. This way, the client learns the sign of $L_i + R_i$, and hence whether its location is above the line, whereas the SP learns nothing. Figure 4 summarizes the protocol to securely evaluate the client's placement relative to a single line.

Recall that, secure polygon enclosure evaluation requires finding the relative placement of the client location with respect to two lines. For instance, in Figure 3 the client learns that its location C is situated above B_3 and below B_2, and concludes it is enclosed by the polygon. Given the i^{th} and j^{th} edges of a polygon and their respective lines $y = S_i * (x - x_i) + y_i$ and $y = S_j * (x - x_j) + y_j$, the client computes

$$E_P((L_i + R_i) \times (L_j + R_j)) = E_P(L_iL_j + L_iR_j + R_iL_j + R_iR_j) =$$

$$E_P(L_iL_j) \times E_P(L_iR_j) \times E_P(R_iL_j) \times E_P(R_iR_j), \ where$$

$$E_P(L_iL_j) = E_P(L_i)^{L_j} = E_P(L_i)^{y_C - S_j \times x_C} =$$

$$E_P(L_i)^{y_C} \times E_P(S_jL_i)^{-x_C}, \ and$$

$$E_P(S_jL_i) = E_P(S_j(y_C - S_ix_C)) = E_P(S_j)^{y_C} \times E_P(S_iS_j)^{-x_C}$$

Next, $E_P(L_iR_j)$ and $E_P(R_iL_j)$ can be computed as follows:

$$E_P(L_iR_j) = E_P(R_j)^{L_i} = E_P(R_j)^{y_C} \times E_P(S_iR_j)^{-x_C}$$

$$E_P(R_iL_j) = E_P(R_i)^{L_j} = E_P(R_i)^{y_C} \times E_P(S_jR_i)^{-x_C}$$

Every target sends periodically to the SP as location updates items $E_A(E_P(R_iR_j))$, $E_A(E_P(S_iS_j))$, $E_A(E_P(S_iR_j))$ and $E_A(E_P(S_jR_i))$, as well as $E_A(E_P(R_i))$, $E_A(E_P(S_i))$, $E_A(E_P(R_j))$, and $E_A(E_P(S_j))$, as shown in Figure 5.

Upon receiving a query, all the above values are sent by the SP to the client, who performs decryption D_A to obtain the Paillier ciphertexts and applies the homomorphic operations described above. Then, the client performs additive blinding to determine $E_P((L_i + R_i)(L_j + R_j) + k)$ and sends it to the SP. Next, the SP decrypts the ciphertext and obtains

[1]We defer discussion on how to securely determine MBR enclosure until Section 3.3.

[2]Although the plaintext space of Paillier encryption consists of positive integers, one can employ a binary encoding to represent negative numbers as in [7].

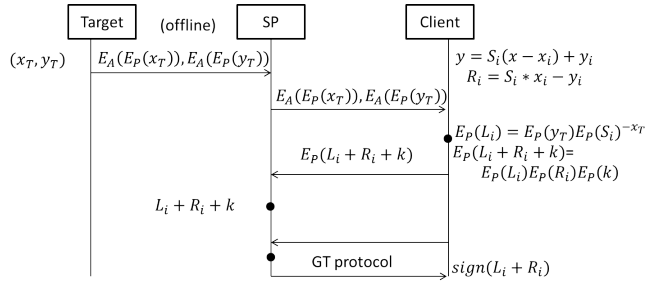

Figure 6: TCZ: placement relative to one line

$(L_i+R_i)(L_j+R_j)+k$, followed by the GT protocol execution between client and SP, after which the client determines the sign of $(L_i + R_i)(L_j + R_j)$. If the sign is negative, then the client location is enclosed by the target's polygon.

The last component of CTZ to be discussed is how to find the two target polygon edges intersecting the vertical line passing through the client location C. Denote by l the number of edges of the target's polygon. By taking in order the x-axis coordinates of the l polygon vertices, we obtain a set of l intervals. Finding which edges intersect the vertical line passing through x_C is the same as finding which two such intervals contain coordinate x_C. If the query point is inside an interval $[x_i, x_j]$, where $j = (i+1) \mod l$, then the following condition must be satisfied[3]:

$$x_i < x_C < x_j.$$

This condition can be efficiently evaluated with the help of mOPE encryption, reviewed in Section 2.1. The evaluation is done at the SP using the mOPE-encrypted values sent by the target and the client. Specifically, the target sends to the SP (in its periodic update) pairs $E_M(x_i), E_M(x_j)$ for all of its polygon sides, whereas the client sends $E_M(x_C)$ at query time.

The SP evaluates the outcome of the following conditions for each target polygon edge:

$$E_M(x_i) \leq E_M(x_C), E_M(x_C) \leq E_M(x_j)$$

and if both hold, then the edge is marked as one of the two intersecting edges. Note that, the client does not learn anything about the target polygon from this secure comparison process, because the comparison is performed entirely at the SP. However, with the above protocol the SP may learn which of the target's polygon edges contain x_C (i.e., the index of the intersecting edge in the polygon). This may allow the SP to find certain characteristics of the polygon, such as the fact that two consecutive edges intersect the vertical line, which may disclose a particular polygon shape. To prevent such additional disclosure, the target can randomly permute the mOPE-encrypted intervals before sending them to the SP.

3.2 Target-in-Client-Zone (TCZ) Enclosure

Next, we focus on the dual problem of determining securely whether a target's location is enclosed within the proximity zone of the client. Recall that we are solving this problem in the more challenging offline setting, where only the SP and the client C participate in the computation, whereas the targets do not. Denote by $T(x_T, y_T)$ the location of the target. Following the same steps as in the CTZ case, we first show how C can determine the relative placement of T with respect to one of C's polygon edges, then we extend this to relative placement with respect to two edges, and finally we show how to determine the two polygon edges intersecting the vertical line passing through T.

Let $y = S_i(x - x_i) + y_i$ be the line equation for the i^{th} edge of C's polygon. The client must determine whether T is above or below the line. Figure 6 illustrates the protocol: targets send periodically to SP $E_A(E_P(x_T))$ and $E_A(E_P(y_T))$. Upon receiving a query, the SP sends these encrypted items to the client, who computes:

$$E_P(R_i) = E_P(S_i * x_i - y_i)$$

$$E_P(L_i) = E_P(y_T - S_i * x_T) = E_P(y_T) \times E_P(x_T)^{-S_i}$$

$$E_P(L_i + R_i + k) = E_P(L_i) \times E_P(R_i) \times E_P(k)$$

where L_i and R_i have the same significance as in Eq. (1), and k is an additive blinding term. Then, the client sends $E_P(L_i + R_i + k)$ to SP which decrypts it and obtains $L_i + R_i + k$, following which the client and the SP engage in the GT protocol. The client determines whether $L_i + R_i > 0$, and consequently whether T is above the i^{th} polygon edge of the client.

The client must find the relative placement of the target with respect to two edges of the client's polygon. Furthermore, the client should learn only whether the target is above one edge and below the other, but should not learn the outcome of any individual edge test. Given the i^{th} and j^{th} polygon edges and their respective lines $y = S_i*(x-x_i)+y_i$ and $y = S_j * (x - x_j) + y_j$, the client computes:

$$E_P((L_i+R_i)(L_j+R_j)) = E_P(L_iL_j+L_iR_j+R_iL_j+R_iR_j) =$$

$$= E_P(L_iL_j) \times E_P(L_iR_j) \times E_P(R_iL_j) \times E_P(R_iR_j), \text{ where}$$

$$E_P(L_iL_j) = E_P((y_T - S_i * x_T)(y_T - S_j * x_T)) =$$

$$= E_P(y_T^2 - (S_i + S_j)x_T y_T + S_iS_jx_T^2)$$

$$= E_P(y_T^2) \times E(x_T y_T)^{-(S_i+S_j)} \times E(x_T^2)^{S_iS_j}$$

and $E_A(E_P(y_T^2))$, $E_A(E_P(x_T y_T))$, $E_A(E_P(x_T^2))$ are periodically uploaded to SP by each target, as shown in Figure 7. Next, the client determines

$$E_P(L_iR_j) = E_P(y_T - S_i*x_T)^{R_j} = (E_P(y_T) \times E_P(x_T)^{-S_i})^{R_j}$$

The client computes similarly $E_P(R_iL_j)$ and $E_P(R_iR_j)$ and uses additive blinding with random k to compute $E_P((L_i + R_i)(L_j + R_j) + k)$, which is sent to the SP. The SP decrypts the message using the Paillier private key and obtains $(L_i + R_i)(L_j + R_j) + k$. Next, the client initiates the GT protocol, at the end of which the client learns the sign of $(L_i + R_i)(L_j + R_j)$, and hence whether the target is inside the client's polygon or not. Meanwhile, the SP does not learn anything about the evaluation outcome.

The remaining component of TCZ is to find the two edges of the client's polygon that intersect the vertical line passing through T. Denote the set of x-axis projections of the l-sided client polygon edges as x'_i, x'_j, where $j = (i+1) \mod l$. The SP needs to determine the client polygon's edges for which

$$x'_i < x_T < x'_j.$$

[3]The condition assumes that $x_i \leq x_j$. Each target will ensure before encryption that each interval is represented such that the x coordinates are swapped if the condition does not hold.

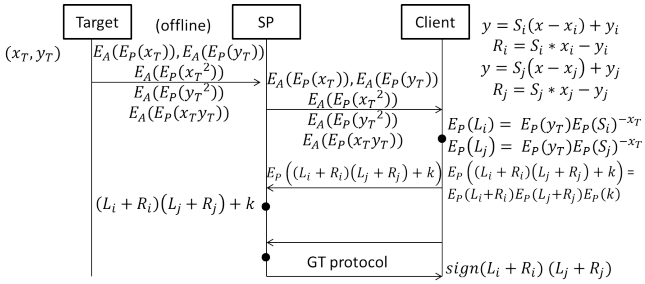

Figure 7: TCZ: placement relative to two lines

The evaluation is done at the SP using the mOPE-encrypted pairs $E_M(x_i'), E_M(x_j')$ sent by the client for all its polygon sides, as well as the value $E_M(x_T)$ sent by the target with its periodic update.

The SP evaluates the outcome of the following conditions for each client polygon edge:

$$E_M(x_i') \leq E_M(x_T), E_M(x_T) \leq E_M(x_j')$$

and if both hold, then the edge is marked as one of the two intersecting edges. As in the case of CTZ, to prevent additional disclosure, the client can randomly permute the mOPE-encrypted intervals before sending them to the SP.

3.3 MBR Filtering

The CTZ and TCZ protocols introduced so far make extensive use of Paillier and ElGamal encryption operations, which are computationally expensive. To reduce the performance overhead, we introduce as an optimization a *filtering* step that reduces the number of targets that need to be considered. Specifically, each target sends the SP, in addition to the encrypted items needed for CTZ and TCZ, an encryption of the minimum bounding rectangle (MBR) of the target's proximity zone polygon. If the client's location is not included in the MBR, then it will not be inside the polygon either. Furthermore, checking secure enclosure in a rectangle is considerably simpler and faster than polygon enclosure. We employ for this test the mOPE scheme [23] described in Section 2.1.

Each target determines the MBR of their proximity zone (as illustrated in Figure 8), encrypts the MBR lower left and upper right coordinates using mOPE, and sends them to the SP as $E_M(x_{LL})$, $E_M(y_{LL})$, $E_M(x_{UR})$ and $E_M(y_{UR})$. At query time, the client also encrypts its query point (x_C, y_C) using mOPE, and sends the SP $E_M(x_C)$ and $E_M(y_C)$. When the SP receives the query, it checks whether the MBR contains the query point through simple numerical comparisons:

$$E_M(x_{LL}) < E_M(x_C) < E_M(x_{UR})$$

$$E_M(y_{LL}) < E_M(y_C) < E_M(y_{UR})$$

If both conditions (i.e., four inequalities) hold, the SP will initiate the CTZ and TCZ protocols for the currently considered target, otherwise it will move to the next target. Due to the efficiency of mOPE, this operation is fast. Unfortunately, mOPE does not have homomorphic properties, and it does not permit evaluation of more complex tests than scalar inequality. Hence it is not suitable for CTZ and TCZ, where Paillier and ElGamal encryption are still necessary.

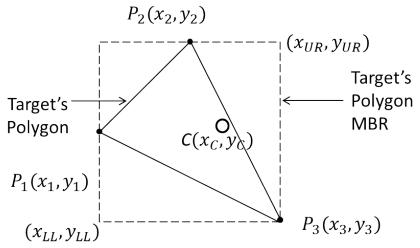

Figure 8: MBR Filtering

3.4 Complete Protocol

The pseudocode in Algorithm 1 shows the complete protocol for secure mutual proximity zone enclosure evaluation. We focus on the online component, i.e., query processing. The offline component is straightforward: each target periodically uploads to the SP the encrypted items specified in the Sections 3.1, 3.2 and 3.3.

The first step consists of target filtering. The client C sends her coordinates encrypted with mOPE to the SP, and the latter checks which target polygon MBRs (also encrypted with mOPE) enclose C's location. Targets that fail this check are eliminated from further processing. For each remaining target, the second step consists of performing CTZ. If the target passes the CTZ evaluation, the third and final step is to run the TCZ protocol. Any target that passes the TCZ test is part of the client's final result.

Within the CTZ protocol, C first determines (lines 1-2) which two of the target T's polygon sides intersect the vertical line passing through C's location. Recall that, since we assume convex polygons, it is guaranteed that there are always two such sides. C does not learn any information about the sides other than their indexes. Next, C computes the relative placement of its location with respect to these two sides (line 3). If the location is above one of the lines and below the other, then the CTZ test is successful. TCZ proceeds in a similar fashion.

Algorithm 1: Secure Mutual Proximity Zone Enclosure

$\mathcal{T}$ = set of targets
$R = \emptyset$ /* *set of results with matching targets* */
Client: Send encr. location C and polygon P_C to SP
SP : For $\forall T \in \mathcal{T}$
SP : If ($C \notin T.MBR$) /* *MBR filtering* */
SP : skip to next T
SP : If (CTZ($C.location$, $T.polygon$) = **false**)
SP : skip to next T
SP : If (TCZ($T.location$, $C.polygon$) = **false**)
SP : skip to next T
SP : $R = R \cup T$

Algorithm 2: CTZ (Location C, Target Polygon P_T)

SP $\Leftrightarrow$ **Client (interactive)**:
1. Let V be the vertical line passing through C
2. Find two sides B_1 and B_2 of P_T that intersect V
3. If (C is in between B_1 and B_2)
4. Return **true**
5. Return **false**

Algorithm 3: TCZ (Location T, Client Polygon P_C)

SP $\Leftrightarrow$ Client (interactive):

1. Let V be the vertical line passing through T
2. Find two sides B_1 and B_2 of P_C that intersect V
3. If (T is in between B_1 and B_2)
4. Return **true**
5. Return **false**

4. SECURITY DISCUSSION

Privacy of Client and Target Locations against SP. The SP must not learn the locations or proximity zones of either the client C or the target T. We discuss only the privacy of the client location, as a similar reasoning holds for the privacy of the target location.

First, in the MBR filtering step, due to the use of mOPE encryption, the SP learns only whether the client location is inside the MBR of the target polygon or not. The SP cannot pinpoint the client to a specific region within the MBR (or outside of it, in case the MBR enclosure test fails). Furthermore, the SP does not know what are the actual coordinates of the MBR. Since every MBR has the same number of edges, all encrypted MBR coordinates are indistinguishable to the SP, according to the IND-OCPA property of mOPE.

Second, in the CTZ protocol execution (or TCZ in the case of target locations), when finding the two target polygon edges that intersect the vertical line passing through C, the enclosure tests for all l intervals (one for each target polygon edge) are done on top of mOPE-encrypted ciphertexts. Since mOPE satisfies IND-OCPA, the SP cannot distinguish between the corresponding x-axis coordinates. All the client learns is that C's x_C coordinate is enclosed by two intervals, which is true for every possible client location and convex target polygon that passed the filtering step.

Third, when performing the placement test relative to two lines, the client uses additive blinding when computing the value $E_P((L_i+R_i)(L_j+R_j)+k)$. Therefore, the SP cannot learn the exact values of the left- and right-hand sides of Eq. (1), and does not learn anything about the polygon edges or the client location.

Privacy of Target Locations against the Client. During the mutual proximity zone protocol, the client C only receives items encrypted with Paillier encryption, which is semantically secure, hence no information can be derived from the ciphertexts. In addition, all evaluation results are binary outcomes (i.e., yes/no) that are obtained through the GT protocol. At the end of the protocol, the client only learns whether the target is in the mutual proximity zone enclosure relationship with the client, and nothing else.

Security of Additive Blinding. The final steps of the CTZ and TCZ protocols include an additive blinding operation with the purpose of protecting the exact value of L_i+R_i from the SP. When the client computes $E_P(L_i+R_i)$, it selects a random number k and then determines $E_P(L_i+R_i+k) = E_P(L_i+R_i) \times E_P(k)$. Then, the SP obtains the plaintext L_i+R_i+k and the client initiates the GT protocol.

We provide a brief security analysis of the additive blinding operation. Let $v = L_i+R_i$, and assume that v has n bits (i.e., all coordinates are mapped to a n-bit fixed precision representation). Denote v's value domain by $d_v = [v_l, v_u]$ where v_l and v_u are the domain boundaries. The cardinality of the value domain is $m = 2^n$. Consider a number k with n' bits and value domain $d_k = [k_l, k_u]$. The cardinality of d_k is $m' = 2^{n'}$. Then, the domain d_{v+k} of $v+k$ is $[v_l+k_l, v_u+k_u]$ and the size of the domain is $m'+m-1 = 2^{n'}+2^n+1$.

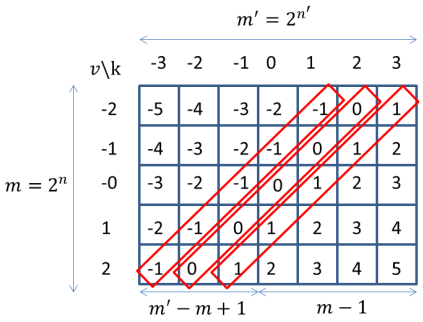

Figure 9: Additive Blinding: $c = v + k$

An example of additive blinding $v+k$ is illustrated in Figure 9, where $d_v = [-2, 2]$, $d_k = [-3, 3]$ and $d_{v+k} = [-5, 5]$. Denote $c = v + k$. When $c \in [v_u + k_l, v_l + k_u] = [-1, 1]$, c can be the result of blinding any value in d_v. Hence, the SP cannot learn anything about v given c. However, when c is outside the highlighted interval, c may not be obtained through blinding from any value in d_v. Hence, there may be some information that SP learns about v. Next, we quantify this probability and show it is negligible in practice.

In Figure 9, there are $m * m' = 2^{n+n'}$ elements in the table. The number of elements in the highlighted interval is $m * (m' - m + 1) = 2^n * (2^{n'} - 2^n + 1)$ and the number of elements outside the interval is $m(m-1) = 2^n(2^n - 1)$. The probability that an element is outside the interval is $m(m-1)/\{m*m'\} = (m-1)/m' = (2^n-1)/2^{n'} \approx 1/2^{n'-n}$. In practice, given that the plaintext space of the encryption functions used is at least 1024 bits, one can easily set $n = 256$ and $n' = 512$, leading to a probability of returning an additively blinded value which leaks information of $1/2^{256}$, hence negligible.

5. EXPERIMENTAL EVALUATION

We implemented a prototype of the proposed techniques for secure mutual proximity zone enclosure, namely: CTZ (Section 3.1), TCZ (Section 3.2) and the MBR filtering protocol (Section 3.3). We also implemented the GT protocol from [14] (Section 2.1) which is used as a building block in our CTZ and TCZ protocols. We developed the prototype using Java JDK 1.6, and we executed all experiments on a 3.4GHz Intel i7 CPU machine with 16GB RAM running Windows 8.

Experimental Settings. We use Paillier and ElGamal encryption, both with 1024-bit key strength, and 128-bit key AES encryption. We represent each coordinate using $n = 128$ bits, which determines the size of the tables created by the GT protocol. The client and target locations are randomly distributed in the domain $[0, 10^6]^2$. We consider a number of targets between 200,000 and 1 million. We generate proximity zones as convex polygons with number of edges varying from 3 to 7. The polygon extent influences the MBR filtering step, so we also vary the polygon size (measured as the resulting MBR size) from 500 × 500 to 4000 × 4000. Table 2 summarizes the parameter settings, with default values shown in boldface.

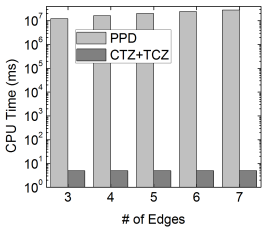

Figure 10: PPD vs CTZ+TCZ (100K targets)

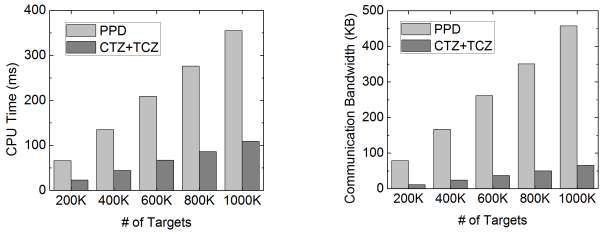

Figure 11: Performance vs. Number of Targets

We use as performance metrics CPU time at the SP and the client, and the amount of client-SP traffic generated. For each experiment, we averaged the results over $1,000$ random queries, and we used 10 distinct random seeds for each run.

5.1 Comparison with Benchmark

To the best of our knowledge, our technique is the first one to allow secure mutual proximity zone testing in the offline case (i.e., where targets do not participate in the protocol). We use as benchmark the *Private Proximity Detection (PPD)* method from [19], which requires both client and targets to be online. The protocol from [19] is an adaptation of our earlier work in [7] where Paillier encryption is used to compute the orientation of the client location with respect to all polygon edges of the target. When the edges are sorted in counter-clockwise direction, by plugging in the client x coordinate in the line equation of all edges, one can determine that the client is enclosed by the polygon if all y results (i.e., orientations) are negative. PPD needs to be run twice for each target, to check mutual proximity. Note that, PPD needs to perform orientation evaluations with respect to *all* edges of the polygon, as opposed to our method which only considers two edges, and is hence more efficient.

Figure 10 shows the CPU time for our method and PPD. Our method clearly outperforms PPD by close to seven orders of magnitude (the time axis is logscale). PPD is not practical even for $100K$ targets, as the time to process a single query is more than 2 hours. On the other hand, our method achieves a time of under $10msec$. Clearly, applying directly interactive methods like PPD or [2] is not suitable for secure mutual proximity detection.

Parameter	Values
Paillier Key	1024 bits
ElGamal Key	1024 bits
AES key	128 bits
Number of targets	200K, 400K, 600K, 800K, **1000K**
Polygon edges	3, 4, 5, 6, **7**
Polygon MBR Size	500×500, **1K×1K**, 2K×2K, 4K×4K

Table 2: Experimental Parameter Settings

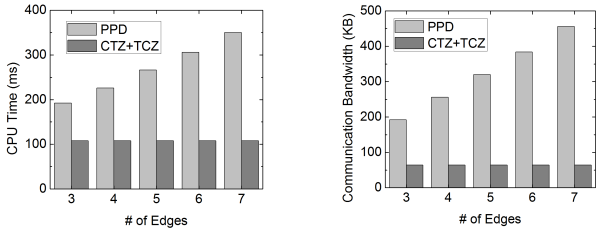

Figure 12: Performance vs. Number of Edges

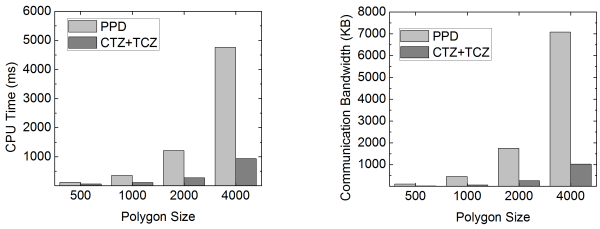

Figure 13: Performance vs. Polygon Size

One of the reasons why PPD is so slow is that it performs no filtering of targets. In contrast, we make use of the novel mOPE encryption to reduce the number of TCZ and CTZ rounds. For fairness, in the rest of this section, we adapt PPD to also include the filtering based on polygon MBRs. Figure 11 shows the relative performance of our method and PPD when varying the number of targets. Filtering improves a lot the performance of PPD, but it is still slower than our method by a factor of four on average. In addition, the communication bandwidth used is close to ten times higher for PPD, due to running GT for every polygon edge. As expected, all methods scale linearly with the number of targets. In the worst case, our method requires $100msec$ processing time, and 60KB communication.

Figure 12 shows the performance overhead when varying the number of edges in the proximity zone polygons. Note that, the PPD overhead grows linearly with the number of edges, as it needs to run orientation evaluation for each edge. On the other hand, our method's cost does not increase significantly with the number of edges (the only part that depends on edge count is finding the edges that intersect the vertical line, but that only requires inexpensive mOPE secure comparisons). Similarly, the GT protocol only has to be executed for two edges, hence the communication cost of our method is also constant.

In Figure 13 we vary the extent of the polygonal regions. Larger polygons decrease the effectiveness of the MBR filtering step, and the overall cost when polygon size grows is higher for both PPD and our method. However, we maintain our relative advantage over PPD. The results so far show the clear superiority of the proposed method over PPD, even when we enhance the latter with filtering. In the remainder of this section, we no longer consider PPD, and we focus on the evaluation of the different components of our secure protocol.

5.2 Performance Breakdown for CTZ/TCZ

We investigate the performance overhead breakdown for the three components of the proposed technique: filtering step, CTZ and TCZ. We vary both the number of targets,

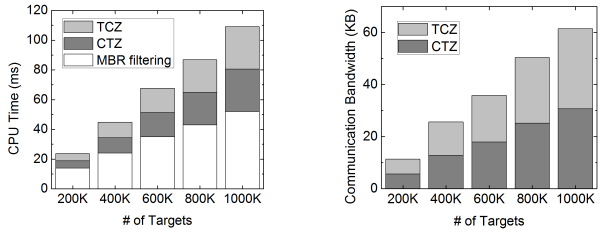

Figure 14: Breakdown vs. number of targets

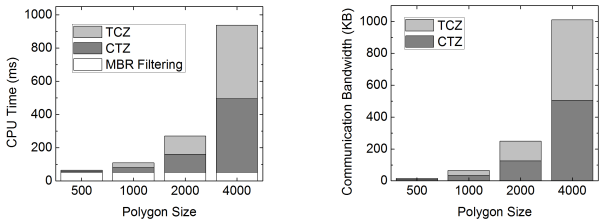

Figure 15: Breakdown vs. size of proximity zones

and the polygon size (recall that the number of edges does not significantly influences performance for our method). Figure 14 shows that the CPU time consists mostly of MBR filtering. As the number of targets grows, the proportion of filtering time decreases, reaching roughly half of the total time for 1 million targets. The remaining CPU time is equally spent on CTZ and TCZ. The communication cost is also equally spent between CTZ and TCZ. The filtering step only requires the client to send its encrypted coordinates to the SP once, which is a negligible amount of communication.

Figure 15 shows the performance breakdown as polygon size increases. For larger polygons (hence larger MBRs), the filtering step's effectiveness decreases considerably, as fewer targets are eligible for filtering. A considerably larger number of CTZ and TCZ evaluations are necessary, so almost all CPU time is split equally among these steps. Overall, the CPU time increases significantly with polygon size, and so does the communication cost, the latter exhibiting the same equal share among CTZ and TCZ. Nevertheless, the performance overhead remains practical, with under one second of CPU time and one megabyte of communication needed in the worst case. Note that, we did not use parallelism in our implementation, but using multiple cores is likely to result in close-to-linear speedup, due to the inherent parallelism of the protocol (each target is independently evaluated from the others).

6. RELATED WORK

Protecting location data is an important problem that has been addressed in a variety of settings. For instance, two approaches for location protection have been investigated in the context of private queries to location-based services (LBS). The objective here is to allow a querying user to retrieve her nearest neighbor among a set of public points of interest without revealing her location to the LBS. The first approach is to use *cloaking regions (CRs)* [9, 5, 18, 12]. Most CR-based solutions implement the spatial k-anonymity paradigm and assume a three-tier architecture where a trusted anonymizer sits between users and the LBS server and generates rectangular regions that contain at least k user locations. This approach is fast, but not secure in the case of outliers. The second approach uses private information retrieval (PIR) protocols [8, 7]. PIR protocols allow users to retrieve an object X_i from a set $X = \{X_1, X_2, \ldots, X_n\}$ stored by a server, without the server learning the value of i. The work in [8, 7] extends an existing PIR protocol [6] for binary data to the LBS domain and proposes approximate and exact nearest neighbor protocols. The latter approach is provably secure, but it is expensive in terms of computational overhead.

Privacy has also been addressed in spatial query outsourcing [13], where data points are encoded by the data owner according to a secret transformation. Users who know the transformation key map their 2-D queries to 1-D points and the query processing is done in the 1-D space. However, the mapping can decrease the result accuracy and the transformation may be vulnerable to reverse-engineering. The work in [25] uses a secret matrix transformation to hide the data points and the query is also transformed. When the server receives the transformed data and the transformed query, it determines which data point is nearest to the query. The method provides exact results, but the matrix transformation is vulnerable to chosen plaintext attacks, as shown in [27].

The solutions in [10, 11] propose the use of a secure index for NN queries. The secure index is given to the client, and the server has the secret key to decrypt each node in the secure index. When there are many data points, the size of the secure index is very large. The work in [22] proposes a scheme using oblivious transfer [20] and PIR, which is provably secure, but very expensive. Furthermore, it only handles point-point distances, but not polygonal regions.

In this paper, our goal is to determine whether two users are mutually situated in each other's proximity regions. In [16] a scheme using Paillier encryption is proposed to compute the distance of two users using the GT protocol [14]. It determines only whether the two users are within a certain distance. In [17] a grid-based scheme is proposed. When the server receives the regions of users, the server computes a minimum and maximum distance between the cloak regions to filter out far-away users. Then, the users refine the result in a peer-to-peer manner using a secure multi-party computation (SMC) protocol [26]. [24] suggests an adaptive grid-based scheme using vicinity regions. However, the technique incurs false positives. [21] proposes a scheme using equality testing. A user's location is approximately selected as one of the points in a grid. Then, by performing a secure equality test, it is determined whether two users are nearby.

Our solution is closely related to secure polygon enclosure. The work in [2] proposes a scalar product protocol and a vector dominance protocol using homomorphic encryption and secure multiparty computation [26]. Building upon the two, it then proposes a secure two-party point-in-polygon inclusion protocol. However, it is very expensive, and does not work in offline mode. The PPD method from [19] also assumes online targets, and it is clearly outperformed by our solution, as shown in Section 5. The recent work in [15] proposes a scheme to test mutual proximity zone enclosure. However, the targets must disclose the slope of each edge of their zones to the client, hence the security of the method is low, and it is vulnerable to chosen plaintext attacks.

7. CONCLUSIONS

In this paper, we proposed a secure mechanism for mutual proximity zone enclosure evaluation. Mobile users de-

fine their own zones of interest, and then are able to privately find friends with whom they are in a mutual proximity zone enclosure relationship, without having to disclose information about their location or proximity zones. Furthermore, our solution allows offline evaluation of mutual proximity, i.e., the target users need not be directly involved in the protocol, and only the querying user (client) and the SP run the secure protocol. We provided a security analysis of the proposed protocols for secure client-in-target-zone (CTZ) and target-in-client-zone (TCZ) evaluation, and we showed through experiments that the performance achieved is practical, with sub-second processing times for up to 1 million users. In future work, we plan to extend our cryptographic protocols to support more complex types of queries, such as top-k queries with respect to the amount of overlap of users' proximity zones. In addition, we will study techniques to accelerate protocol execution using parallelism and graphical processing units (GPUs).

Acknowledgments. The work reported in this paper has been partially supported by the Purdue Cyber Center and by the National Science Foundation under grant CNS-1111512.

8. REFERENCES

[1] NIST FIPS 197 - Advanced Encryption Standard (AES). *National Institute of Standards and Technology*, 2001.

[2] M. J. Atallah and W. Du. Secure multi-party computational geometry. In *WADS Conference Proceedings*, pages 165–179. LNCS, 2001.

[3] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino. Secure knn query processing in untrusted cloud environments. *IEEE Transactions on Knowledge and Data Engineering*, Jan 2014.

[4] T. Elgamal. A public-key cryptosystem and a signature scheme based on discrete logarithm. *IEEE Transactions on Information and Theory*, 1985.

[5] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS Conference Proceedings*, pages 620–629. IEEE, June 2005.

[6] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *Automata, Languages and Programming Conference Proceedings*, pages 803–815. LNCS, 2005.

[7] G. Ghinita, P. Kalnis, M. Kantarcioglu, and E. Bertino. Approximate and exact hybrid algorithms for private nearest-neighbor queries with database protection. *Geoinformatica*, Dec 2010.

[8] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: Anonymizers are not necessary. In *SIGMOD Conference Proceedings*, pages 121–132. ACM, June 2008.

[9] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys Conference Proceedings*, pages 31–42. ACM, 2003.

[10] H. Hu, J. Xu, C. Ren, and B. Choi. Processing private queries over untrusted data cloud through privacy homomorphism. In *ICDE Conference Proceedings*, pages 601–612. IEEE, April 2011.

[11] H. Hu, J. Xu, X. Xu, K. Pei, B. Choi, and S. Zhou. Private search on key-value stores with hierarchical indexes. In *ICDE Conference Proceedings*. IEEE, 2014.

[12] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE TKDE*, 2007.

[13] A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD Conference Proceedings*, pages 239–257. LNCS, 2007.

[14] H.-Y. Lin and W.-G. Tzeng. An efficient solution to the millionaires problem based on homomorphic encryption. In *ACNS*, pages 456–466. Springer, January 2005.

[15] X. Lin, H. Hu, H. P. Li, J. Xu, and B. Choi. Private proximity detection and monitoring with vicinity regions. In *MobiDE Conference Proceedings*, pages 5–12. ACM, June 2013.

[16] Y.-L. Luo, L.-S. Huang, and H. Zhong. Secure two-party point-circle inclusion problem. In *Journal of Computer Science and Technology*, pages 88–91. LNCS, Jan 2007.

[17] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, and S. Jajodia. Privacy-aware proximity based services. In *MDM Conference*, pages 31–40. IEEE, 2009.

[18] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB Conference Proceedings*, pages 763–774. VLDB, Sep 2006.

[19] B. Mu and S. Bakiras. Private proximity detection for convex polygons. In *MobiDE Conference Proceedings*, pages 36–43. ACM, June 2013.

[20] M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. In *Advances in Cryptology*, pages 573–590. LNCS, 1999.

[21] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *NDSS Conference Proceedings*, pages 1–17. ISOC, Feb 2011.

[22] R. Paulet, M. G. Kaosar, X. Yi, and E. Bertino. Privacy-preserving and content-protecting location based queries. In *IEEE Conference Proceedings*, pages 44–55. IEEE, 2012.

[23] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *Security and Privacy*, pages 1–20. IEEE, 2013.

[24] L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu. Private and flexible proximity detection in mobile social networks. In *MDM Conference Proceedings*, pages 75–84. IEEE, 2010.

[25] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis. Secure knn computaiton on encrypted databases. In *SIGMOD Conference Proceedings*, pages 1–12. ACM, June 2009.

[26] A. C. Yao. Protocols for secure computations. In *SFCS Conference Proceedings*. IEEE, 1982.

[27] B. Yao, F. Li, and X. Xiao. Secure nearest neighbor revisited. In *ICDE Conference Proceedings*, pages 733–744. IEEE, April 2013.