TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/11 pp270–280
Volume 21, Number 3, June 2016

Private Proximity Detection for Convex Polygons

Bin Mu and Spiridon Bakiras*

Abstract: Proximity detection is an emerging technology in Geo-Social Networks that notifies mobile users when they are in proximity. Nevertheless, users may be unwilling to participate in such applications if they are required to disclose their exact locations to a centralized server and/or their social friends. To this end, *private* proximity detection protocols allow any two parties to test for proximity while maintaining their locations secret. In particular, a private proximity detection query returns only a boolean result to the querier and, in addition, it guarantees that no party can derive any information regarding the other party's location. However, most of the existing protocols rely on simple grid decompositions of the space and assume that two users are in proximity when they are located inside the same grid cell. In this paper, we extend the notion of private proximity detection, and propose a novel approach that allows a mobile user to define an arbitrary convex polygon on the map and test whether his friends are located therein. Our solution employs a secure two-party computation protocol and is provably secure. We implemented our method on handheld devices and illustrate its efficiency in terms of both computational and communication costs.

Key words: proximity detection; secure computations; location privacy

1 Introduction

The emergence of Geo-Social Networks (GeoSNs), such as Foursquare (https://foursquare.com/), facilitates the development of novel applications that combine social networking features with location-based services. In particular, a GeoSN enhances the traditional social networking graph with spatial information, by allowing users to "check in" at arbitrary geographic locations. Mobile users may then utilize this information to identify their social *friends* that are spatially close (e.g., in order to meet at a nearby coffee shop). This is typically called a *proximity detection* query.

A trivial way to process such queries is to store all location information at the GeoSN server, in plaintext format. Alternatively, users may choose to bypass the server and exchange their locations (in plaintext), on-demand, in a peer-to-peer manner. Clearly, both methods might reveal a lot of information about an individual's lifestyle to the GeoSN server and/or his friends. If the leaked information is more than what the user is willing to disclose, he may be discouraged from registering with the GeoSN. Therefore, to protect privacy, GeoSN queries should not disclose any additional information regarding the location of a user, besides the information that can be derived from the query result.

To this end, *private* proximity detection protocols allow any two parties to test for proximity while maintaining their locations secret. In particular, a private proximity detection query returns only a boolean result to the querier and, in addition, it guarantees that no party can derive any information regarding the other party's location. Nevertheless, the current state-of-the-art private proximity detection protocols[1, 2] rely on simple grid decompositions of the space and assume

• Bin Mu is with the Department of Computer Science, Graduate Center, City University of New York, New York, NY 10016, USA. E-mail: bmu@gc.cuny.edu.
• Spiridon Bakiras is with the Department of Computer Science, Michigan Technological University, Houghton, MI 49931, USA. E-mail: sbakiras@gmail.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2016-02-05; accepted: 2016-03-07

that two users are in proximity when they are located inside the same grid cell.

However, this approach may not be sufficient in certain situations. Assume, for example, that Alice is enjoying her free time in the Central Park area of Manhattan. She wants to know whether Bob is also visiting the park (the area marked with the dashed line in Fig. 1), in order to pursue some outdoor activities together. Traditional proximity detection queries are only able to discover Bob within a (roughly) circular area around Alice, as shown in Fig. 1. A naïve solution to overcome this limitation is to enlarge the search range, so that it encloses the entire Central Park area. Clearly, this approach is not optimal, as it may lead to a large number of false positives. An alternative method is to leverage the existing space partitioning protocols, i.e., allow Alice to search for Bob in multiple grid cells, instead of just one. While this is a viable solution, it has two major shortcomings. First, to achieve an acceptable accuracy level, the underlying grid has to be very fine, thus leading to high query processing costs. Second, the fine grid allows Alice to search for Bob on a very small area, by submitting identical grid cells in the proximity detection query.

To this end, this paper extends the notion of private proximity detection, and introduces a novel approach that allows a mobile user to define an arbitrary convex polygon on the map and test whether her friends are located therein. Returning to the example of Fig. 1, Alice would simply specify the coordinates of the four rectangular edges in order to detect Bob's presence in Central Park. Our solution employs a secure two-party computation protocol that is based solely on public key homomorphic encryption operations. In addition, it offers the highest level of privacy to the involved parties, since (1) Alice only learns the query result and (2) Bob only learns the number of edges in Alice's polygon. We implemented our method on handheld devices, and illustrate its efficiency in terms of both computational and communication cost.

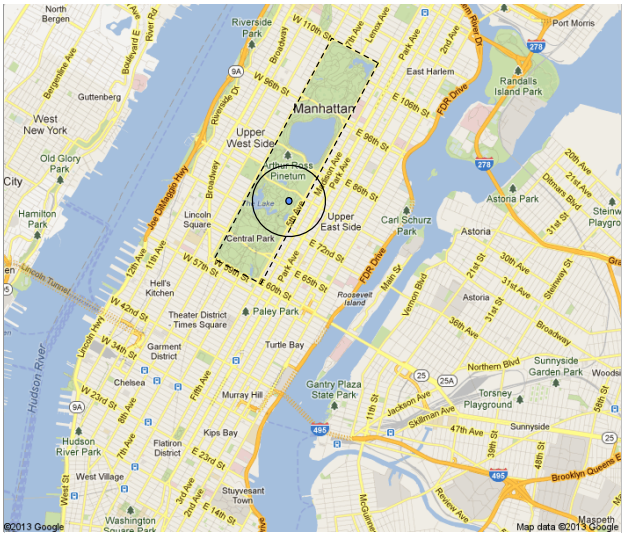


Fig. 1 Motivating example.

In summary, the main contributions of our work are the following:

- We introduce a generalized proximity detection query that incorporates arbitrary convex polygons instead of fixed tessellations.
- We provide a secure and efficient solution based on public key homomorphic encryption.
- We extend the basic approach, in order to enable the target of the proximity detection query to control her location disclosure.
- We show how to leverage the basic method to handle arbitrary concave polygons.
- We implement the cryptographic primitives of our protocol on handheld devices and present real query processing times.

The rest of the paper is organized as follows. Section 2 describes the cryptographic primitives utilized in our methods and summarizes previous work on private proximity detection. Section 3 presents the formal definition of the new proximity detection query and describes the underlying threat model and security. Section 4 introduces the details of our basic protocol and Section 5 presents two extensions to that protocol. Section 6 illustrates the results of our implementation on mobile devices and Section 7 concludes our work.

2 Background

Section 2.1 introduces the cryptographic primitives utilized in our methods and Section 2.2 surveys the related work on private proximity detection.

2.1 Preliminaries

Homomorphic encryption. Most public key cryptosystems in the literature are partially homomorphic, i.e., they facilitate the evaluation of one algebraic operation (either addition or multiplication) directly on the ciphertext space. In our work, we utilize *additively* homomorphic encryption, which allows for the following computations. First, given the encryptions $E(m_1)$ and $E(m_2)$ of two plaintext

messages m_1 and m_2, we can compute the encryption of $(m_1 + m_2)$ by multiplying the two ciphertexts:

$$E(m_1 + m_2) = E(m_1) \cdot E(m_2).$$

Second, any message m can be multiplied with a *plaintext* constant c as follows:

$$E(c \cdot m) = E(m)^c.$$

In our implementation, we utilize two different additively homomorphic cryptosystems, namely the Paillier[3] and ElGamal[4] cryptosystems. The major advantage of Paillier's scheme (Fig. 2) is that it can decrypt arbitrarily large plaintexts very efficiently. However, all operations are computed in modulo n^2 arithmetic, where n^2 is typically a 2048-bit number. As a result, the basic cryptographic operations, such as modular multiplication and exponentiation, are relatively expensive. On the other hand, the modulus size in ElGamal's scheme (Fig. 3) is typically 1024 bits, so it is much more efficient in terms of computational cost. The limitation of the ElGamal cryptosystem is that it can only decrypt small plaintext values, because the decryption function involves the computation of a discrete logarithm, which is a very difficult problem in cryptography. Both schemes produce ciphertexts of size 256 bytes, so they are comparable in terms of communication cost.

Note that, both cryptosystems are semantically secure, i.e., it is infeasible to derive any information about a plaintext, given its ciphertext and the public key that was used to encrypt it. The security of Paillier's scheme is based on the decisional composite residuosity assumption, while the security of ElGamal's scheme is based on the decisional Diffie-Hellman assumption.

Paillier cryptosystem

Key generation

1. Choose two large primes p and q of equal length, and compute the RSA modulus $n = pq$
2. The *public* key is n
3. The *private* key is $\varphi(n) = (p-1)(q-1)$

Encryption

1. Let m be the private message
2. Choose r uniformly at random from $\mathbb{Z}_n^*$
3. Compute ciphertext $c = (mn+1)r^n \bmod n^2$

Decryption

1. Compute $m = \frac{(c^{\varphi(n)} \bmod n^2)-1}{n} \cdot \varphi(n)^{-1} \bmod n$

Fig. 2 The Paillier cryptosystem.

ElGamal cryptosystem

Key generation

1. Instantiate a cyclic group G of prime order p, with generator g (G, g, and p are public knowledge)
2. Choose a *private* key x, uniformly at random from $\mathbb{Z}_p^*$
3. Publish the *public* key $h = g^x$

Encryption

1. Let m be the private message
2. Choose r, uniformly at random from $\mathbb{Z}_p^*$
3. Compute ciphertext $(c_1, c_2) = (g^r, h^{r+m})$

Decryption

1. Compute $h^m = c_2 \cdot (c_1^x)^{-1}$
2. Solve the discrete logarithm to retrieve m

Fig. 3 The ElGamal cryptosystem.

Secure two-party computation. A secure two-party computation protocol[5] enables two parties, Alice and Bob, to jointly compute a function based on their respective inputs, without having to reveal their input to the other party. In other words, the two parties will only learn the result of the computation and nothing else. Yao's *garbled circuit* technique[6] is a generic two-party computation protocol that can evaluate securely any function f, given its Boolean circuit representation. In particular, Bob first generates an *encrypted* version of the circuit that incorporates his own input, and sends it to Alice. Then, Alice and Bob engage in a series of Oblivious Transfer (OT)[7] executions that allow Alice to retrieve securely the keys corresponding to her input bits. Finally, Alice evaluates the circuit and learns the result of the function. Even though the actual circuit evaluation can be very efficient[8], the large number of OT invocations is a performance bottleneck in terms of both computational and communication cost.

Consequently, researchers have looked into more specialized, i.e., application dependent, protocols that are typically built around homomorphic encryption. One such example is *private equality testing*, which is used extensively in previous work[1, 2]. In this protocol, Alice and Bob hold a pair of values a and b, respectively, and want to know if the two values are equal. Alice encrypts her input with her public key and sends $E(a)$ to Bob. Next, Bob utilizes the properties of homomorphic encryption to produce $E(r \cdot (a-b))$, where r is a random number that masks the result so that it is infeasible for Alice to derive any information regarding the value $(a-b)$. Finally, Alice decrypts the

the result and infers that $a = b$ if and only if the result is zero. The security of this protocol has been proven in Refs. [2, 9].

2.2 Related work

Ruppel et al.[10] utilized a symmetric key cipher that encrypts locations by applying a distance-preserving transformation. A set of friends share a common key and use it to encrypt their location prior to uploading it to the server. Due to the distance-preserving property of the transformation, the server can determine whether any two friends are within a given proximity threshold. Clearly, this approach leaks some location information, as the server learns the actual distances among all users. Furthermore, if a user colludes with the server and reveals the shared key, all user locations are compromised. *Longitude*[11] is a similar approach, but the underlying transformation does not disclose the exact distances (i.e., it results in a loss of accuracy).

Most private proximity detection algorithms in the literature employ a tessellation method (typically a regular grid) to partition the space into a fixed number of cells. In this way, they reduce the proximity detection problem into an *equality testing* problem: identify whether the two parties are located inside the same or nearby cells. *FriendLocator*[12] and *VicinityLocator*[13] assume that the two parties share a secret key and use it to encrypt (with a deterministic symmetric cipher) the *ids* of certain cells nearby their location. The encrypted values are uploaded to the server, who can determine (by matching the ciphertexts) whether the two users lie in the same or adjacent cells of the grid. Clearly, both schemes are vulnerable to collusions with the server, since the party that colludes can learn the approximate location of the other party.

In *C-Hide&Seek*[14], every user shares his secret key with his friends, and uses this key to encrypt his up-to-date location. When another user issues a proximity request, the server simply forwards all the encrypted locations that it currently stores. Therefore, the proximity detection is performed at the querier, which enables him to identify (with a simple brute force approach) the approximate locations of all his friends. *C-Hide&Hash*[14] assumes a similar location update procedure as *C-Hide&Seek*. However, for proximity detection, it employs a secure computation protocol between the server and the querier. Nevertheless, due to the shared keys among the users, this scheme is also vulnerable to collusions with the server.

Zhong et al.[1] proposed three schemes, namely *Louis*, *Lester*, and *Pierre* that are based on secure computations. The main idea in all protocols is to compute the distance between two parties using the properties of homomorphic encryption. First, *Louis* computes the actual distance, but requires a trusted third-party that only returns the result (i.e., true or false) of the proximity detection query. *Lester* does not require a third-party, but instead masks the actual distance d in a way that its computation time increases linearly with d (i.e., d is retrieved efficiently only when its value is relatively small). Finally, *Pierre* utilizes a regular grid to discretize the users' locations. It then employs a secure two-party computation protocol to determine whether the users lie within the same or adjacent cells in the grid. Narayanan et al.[2] partitioned the space with three overlapping tessellations, in order to improve the accuracy of the proximity detection. When two parties want to test for proximity, they employ a secure computation protocol (similar to *Pierre*) to identify whether they are located in the same cell of at least one tessellation.

Among all the aforementioned protocols, *Pierre*[1] and Narayanan et al.'s[2] provide the strongest privacy guarantees, i.e., the querier only learns the proximity result, while all remaining parties learn nothing. However, as mentioned in Section 1, these schemes are not directly applicable to our problem setting, because they can not handle arbitrary polygonal shapes.

More similar to our work are protocols for the *secure point inclusion* problem. This problem was first introduced by Atallah and Du[15] as part of a collection of protocols for secure multiparty computational geometry. The two-party protocol for the secure point inclusion problem leverages a number of basic sub-protocols (such as scalar product and vector dominance) that are also proposed in Ref. [15]. Nevertheless, these protocols are computationally expensive and lack formal security proofs. Thomas[16] solved the secure point inclusion problem for star-shaped polygons, while Ye et al.[17] addressed convex polygons. However, these methods are not secure in our problem setting, because the proximity result is computed by the party that owns the fixed point, and has to be transmitted to the querier in an additional round. This property enables collusions among the participating entities that may leak location information regarding the querier's polygon. On the other hand, our method avoids such collusions, because the proximity result is computed at the querier.

3 Problem Definition

Alice (the querier) holds a convex polygon P consisting of N vertices $p_0, p_1, \ldots, p_{N-1}$. The vertices are labeled in a counterclockwise order, as shown in Fig. 4. The coordinates of vertices p_i are denoted as (p_i^x, p_i^y). Bob holds a single point $q = (q^x, q^y)$ that represents his current location. Alice wants to know whether Bob is located inside or on the boundary of P. The privacy guarantees provided by our protocol are the following:

- Alice learns only the result of the proximity detection query (a boolean value). Bob's exact location remains secret.
- Bob does not learn the query result. Furthermore, the only information he can derive about Alice's polygon is the number of edges N. The location, shape, and size of P remain secret.

We assume that both parties can be the adversaries in this protocol. Alice's goal is to pinpoint Bob in an area smaller than the one that can be inferred from the outcome of the protocol. On the other hand, Bob wants to deduce any additional information regarding Alice's polygon, besides the number of edges. Finally, we assume that both parties run in polynomial time and are "semi-honest", i.e., they will follow the protocol correctly, but will try to gain any advantage by analyzing the information exchanged during the protocol execution.

Note that Bob may have his own privacy requirement, namely that he does not want to be found within an area smaller than a certain threshold. We address this issue in our enhanced protocol in Section 5. Furthermore, it is possible for Alice to locate Bob with a brute-force attack, i.e., she can initiate a sequence of proximity detection queries that cover the entire space where Bob might be located. A straightforward solution here is for Bob to decline successive queries that are not sufficiently apart in time.

4 Private Proximity Detection

4.1 Geometric solution

Our proximity detection query can be solved by performing N point orientation computations[18]. Given an ordered triple of points $\langle p_i, q, p_{i+1}\rangle$ on the plane, we say that they have (Fig. 5):

- *Positive* orientation if their angle is a counterclockwise turn;
- *Negative* orientation if their angle is a clockwise turn; and
- *Zero* orientation if they are collinear.

Given the coordinates of the three points, the orientation is computed by the sign of the following determinant:

$$\theta_i = \begin{vmatrix} 1 & p_i^x & p_i^y \\ 1 & q^x & q^y \\ 1 & p_{i+1}^x & p_{i+1}^y \end{vmatrix} = q^x(p_{i+1}^y - p_i^y) + q^y(p_i^x - p_{i+1}^x) + (p_i^y p_{i+1}^x - p_i^x p_{i+1}^y) \quad (1)$$

Therefore, the following algorithm computes the correct proximity result:

(1) For $i \in \{0, 1, \ldots, N-1\}$ and $j = (i+1) \bmod N$, compute the orientation θ_i of point q with respect to the line segment $\overline{p_i p_j}$. The vertices are visited in a counterclockwise order, as shown in Fig. 4.

(2) If $\theta_i \leqslant 0$, $\forall i \in \{0, 1, \ldots, N-1\}$, return *true*; otherwise, return *false*.

The correctness of this algorithm (for convex polygons) follows from the point orientation property. That is, the orientation result determines the half-plane where point q is located if you infinitely extend the line segment $\overline{p_i p_j}$ in both directions. In other words, q lies in the intersection of the N half-planes. When all point orientations are negative, this area is equal to the convex polygon.

4.2 Secure protocol

We assume that, prior to the protocol execution, Alice has published her public keys for the Paillier and ElGamal cryptosystems. In what follows, we use $E_{\mathrm{P}}(\cdot)$ to denote encryption under Paillier's scheme and $E_{\mathrm{G}}(\cdot)$ to denote encryption under ElGamal's scheme. The protocol consists of three major steps. First, Bob

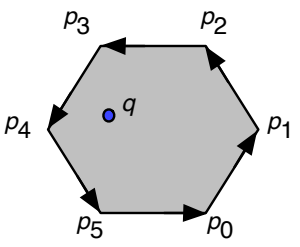


Fig. 4 Proximity detection example.

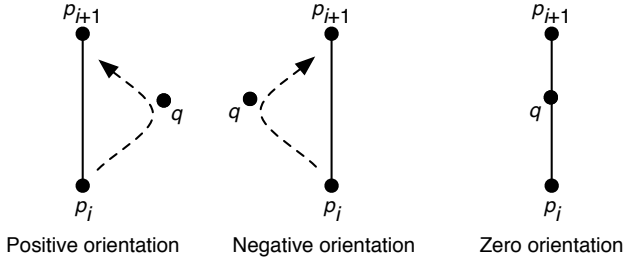


Fig. 5 Point orientation.

computes the encryptions of all θ_i under Alice's public key. Next, Alice and Bob engage in a series of secure comparison protocols that allow Bob to compute the encryptions of the signs for all θ_i. Finally, Bob merges these results into a single message that he sends to Alice. The detailed protocol is shown in Fig. 6.

Initially, Alice uses her Paillier public key to encrypt (for each edge) her own input, as dictated by Eq. (1). She then sends a total of $3N$ ciphertexts to Bob (Step 1). Bob cannot decrypt any of these ciphertexts because he does not have Alice's public key. However, he is able to incorporate his own input, using the properties of additively homomorphic encryption (Step 2). Note that, the final result in Step 2 is not the encryption of θ_i, as it is shown in Eq. (1). The last term adds the value -1 to the result, in order to produce the encryption of $(\theta_i - 1)$. The reason behind this approach is to enforce the points that lie on a polygon edge to produce a negative orientation result, instead of zero.

Next, Alice and Bob employ a secure comparison protocol (which is introduced in the next section) to compute an encrypted representation of the sign of each θ_i (Step 3). Specifically, the protocol allows Bob to compute the encryption of σ_i, where $\sigma_i = 0$ if and only if $\theta_i < 0$ (otherwise, it has a fixed value $t > 0$). During the protocol execution, no party can derive any information regarding the actual value or the sign of θ_i. Bob then combines all orientation results into one ciphertext, i.e., $E_{\mathrm{P}}(\sigma) = E_{\mathrm{P}}(\sigma_0 + \sigma_1 + \cdots + \sigma_{N-1})$. However, he can not send this value to Alice because, if $\sigma > 0$, Alice can figure out the number of edges that produced a positive orientation and, thus, she can eliminate some areas from the search space. Therefore, Bob multiplicatively masks the aggregate result (Steps 3 and 4), by computing $E_{\mathrm{P}}(r \cdot \sigma)$. Finally, Bob sends the masked result to Alice who decrypts it with her private key (Step 5). If the decrypted value is zero, q is located inside (or on the boundary of) P. Otherwise, it is impossible for Alice to determine how many point orientations were positive.

PPD_Convex

Input: Alice has polygon P with N vertices $p_0, p_1, \ldots, p_{N-1}$
Bob has point q

Output: **true** if q in P, **false** otherwise

1. For $i \in \{0, 1, \ldots, N-1\}$ and $j = (i+1) \bmod N$, Alice sends to Bob $E_{\mathrm{P}}(p_j^y - p_i^y)$, $E_{\mathrm{P}}(p_i^x - p_j^x)$, and $E_{\mathrm{P}}(p_i^y p_j^x - p_i^x p_j^y)$
2. For $i \in \{0, 1, \ldots, N-1\}$ and $j = (i+1) \bmod N$, Bob computes $E_{\mathrm{P}}(\theta_i) = E_{\mathrm{P}}(p_j^y - p_i^y)^{q^x} \cdot E_{\mathrm{P}}(p_i^x - p_j^x)^{q^y} \cdot E_{\mathrm{P}}(p_i^y p_j^x - p_i^x p_j^y) \cdot E_{\mathrm{P}}(-1)$
3. Alice and Bob engage in a series of secure comparison protocols and Bob computes, $\forall i \in \{0, 1, \ldots, N-1\}$, $E_{\mathrm{P}}(\sigma_i)$, where $\sigma_i > 0$ if $\theta_i \geqslant 0$ and $\sigma_i = 0$ if $\theta_i < 0$
4. Bob chooses r uniformly at random from $\mathbb{Z}_n^*$, computes the masked result $E_{\mathrm{P}}(r \cdot \sigma) = [\prod_{i=0}^{N-1} E_{\mathrm{P}}(\sigma_i)]^r$, and sends it to Alice
5. Alice decrypts $E_{\mathrm{P}}(r \cdot \sigma)$ with her private key and, if $r \cdot \sigma = 0$, she returns **true**; otherwise, she returns **false**

Fig. 6 The private proximity detection protocol.

4.3 Secure comparison protocol

In Step 3 of the PPD_Convex protocol, Bob holds the encryptions of all point orientation results and wants to compute the encryptions of their corresponding signs. For this task, we borrow a secure comparison protocol (Fig. 7) that is introduced by Erkin et al.[19] as part of their privacy preserving face recognition protocol.

Suppose we use ℓ-bit numbers to represent the orientation. In Step 1, Bob computes the encryption of $s = (\theta_i + 2^\ell)$, which is an $(\ell+1)$-bit number whose Most Significant Bit (MSB) determines the sign of θ_i: if it is 1, $\theta_i \geqslant 0$; otherwise, $\theta_i < 0$. The value of the

Sec_Comp

Input: Bob has $E_{\mathrm{P}}(\theta_i)$ under Alice's public key
ℓ is the max bit-size of θ_i

Output: Bob computes $E_{\mathrm{P}}(\sigma_i)$ where $\sigma_i > 0$ if $\theta_i \geqslant 0$ and $\sigma_i = 0$ if $\theta_i < 0$

1. Bob computes $E_{\mathrm{P}}(s) = E_{\mathrm{P}}(\theta_i + 2^\ell) = E_{\mathrm{P}}(\theta_i) \cdot E_{\mathrm{P}}(2^\ell)$
2. Bob generates a uniformly random $(k + \ell + 1)$-bit number r (where $k = 100$), computes $E_{\mathrm{P}}(s + r) = E_{\mathrm{P}}(s) \cdot E_{\mathrm{P}}(r)$, and sends it to Alice
3. Alice decrypts the message, computes $a = (s + r) \bmod 2^\ell$, and sends $E_{\mathrm{P}}(a)$ to Bob
4. For $i \in \{0, 1, \ldots, \ell-1\}$, Alice sends to Bob $E_{\mathrm{G}}(a_i)$, i.e., the ElGamal encryption of the i-th bit of a_i
5. Bob computes $b = r \bmod 2^\ell$
6. For $i \in \{1, 2, \ldots, \ell-1\}$, Bob sets $E_{\mathrm{G}}(w_i)$ equal to $E_{\mathrm{G}}(a_i)$ if $b_i = 0$, or $E_{\mathrm{G}}(1) \cdot E_{\mathrm{G}}(a_i)^{-1}$ if $b_i = 1$ (the i-th bit of b)
7. Bob chooses $d \in \{1, -1\}$ and, for $i \in \{0, 1, \ldots, \ell-1\}$, he computes $E_{\mathrm{G}}(c_i) = E_{\mathrm{G}}(a_i) \cdot E_{\mathrm{G}}(b_i)^{-1} \cdot E_{\mathrm{G}}(d) \cdot [\prod_{j=i+1}^{\ell-1} E_{\mathrm{G}}(w_j)]^3$
8. For $i \in \{0, 1, \ldots, \ell-1\}$, Bob chooses v_i uniformly at random from $\mathbb{Z}_p^*$, computes $E_{\mathrm{G}}(v_i \cdot c_i) = E_{\mathrm{G}}(c_i)^{v_i}$, and sends a permuted version of the results to Alice
9. Alice decrypts all messages and, if one of them is zero, she sets $\delta = 1$; otherwise $\delta = 0$. She sends $E_{\mathrm{P}}(\delta)$ to Bob
10. If $d = -1$, Bob sets $E_{\mathrm{P}}(\delta) = E(1) \cdot E_{\mathrm{P}}(\delta)^{-1}$
11. Bob computes $E_{\mathrm{P}}(\sigma_i) = E_{\mathrm{P}}(s) \cdot [E_{\mathrm{P}}(a) \cdot E_{\mathrm{P}}(b)^{-1} \cdot E_{\mathrm{P}}(\delta)^{2^\ell}]^{-1}$

Fig. 7 The secure comparison protocol.

MSB can be inferred from the result of $[s-(s \bmod 2^\ell)]$, which is 0 if MSB $= 0$ and 2^ℓ if MSB $= 1$. Therefore, Alice and Bob engage in a series of steps to securely compute $(s \bmod 2^\ell)$. Initially (Step 2), Bob generates a uniformly random number r in order to additively mask s, i.e., he sends the encryption of $(s+r)$ to Alice. Alice decrypts the message, reduces it modulo 2^ℓ, and sends the encryption of $a = [(s+r) \bmod 2^\ell]$ back to Bob (Step 3).

Bob now needs to subtract $b = (r \bmod 2^\ell)$ from a, in order to compute $(s \bmod 2^\ell)$. However, this is not sufficient, as the subtraction may cause the result to underflow when $a < b$. Instead, the correct approach is to securely compute the outcome of the above comparison ($\delta = 1$ if true, $\delta = 0$ if false) and then compute the desired result:

$$(s \bmod 2^\ell) = a - b + \delta \cdot 2^\ell.$$

Consequently, we are left with an instance of Yao's millionaire problem, i.e., we need to determine whether a (held by Alice) is smaller than b (held by Bob). Both inputs are ℓ-bit numbers, so we use index $i \in \{0, 1, \ldots, \ell-1\}$ to represent the individual bits. First (Step 4), Alice sends to Bob the encryptions of all her bits a_i. Note that, at this point, Alice switches to the more computationally efficient ElGamal scheme. Bob then chooses a random value d (either 1 or -1) and computes the following encryptions (Steps 6 and 7), where $w_j = a_j \oplus b_j$:

$$c_i = a_i - b_i + d + 3 \sum_{j=i+1}^{\ell-1} w_j.$$

Suppose that $d = 1$. If $a \geqslant b$, then all c_i will be non-zero. On the other hand, if $a < b$, then exactly one c_i will be zero (at the most significant bit position where the corresponding bits differ). If $d = -1$ the situation is identical, except that the zero value occurs when $a \geqslant b$. Next, Bob multiplicatively masks the individual c_i by raising them to a random power v_i. He also permutes the encryptions and sends them back to Alice (Step 8).

Alice decrypts all the c_i and checks whether one of them is zero. If this is the case, she sets $\delta = 1$; otherwise she sets $\delta = 0$. Note that, Alice does not know the value of d that Bob has selected, so she can not determine whether an underflow has occurred. Finally, she switches back to Paillier's cryptosystem and sends the encryption of δ to Bob (Step 9). If $d = -1$, Bob adjusts the value of δ (Step 10) and eventually computes (Step 11) the encryption of

$$\sigma_i = s - (a - b + \delta \cdot 2^\ell),$$

where $\sigma_i = 0$, if $\theta_i < 0$ and $\sigma_i = 2^\ell$, if $\theta_i \geqslant 0$.

4.4 Security

We will prove the security of the PPD_Convex protocol (which also includes the Sec_Comp protocol) for semi-honest adversaries, following the simulation paradigm[5]. In particular, we need to show that, for each party, we can simulate the distribution of messages that the party receives, given only the party's input and output in this protocol. This is true because, if we can simulate each party's view from only their respective input and output, the messages themselves reveal no additional information.

First, Alice's input consists of N vertices and her output is $r \cdot \sigma$. In Step 2 of Sec_Comp, Alice receives the encryption of a uniformly random number from Bob. The simulator knows Alice's public key, so it can simply generate the encryption of a random $(k+\ell+1)$-bit number. Furthermore, in Step 8 of Sec_Comp, Alice receives ℓ numbers that are either all random or there is a single one with value zero. The simulator knows how the protocol works, so it can either generate ℓ random encryptions, or $(\ell-1)$ random encryptions plus an encryption of zero. Finally, in Step 4 of PPD_Convex, Alice receives the encrypted result from Bob. The simulator knows Alice's output and can, thus, produce the corresponding ciphertext (an encryption of either zero or a random number).

In Bob's case, the input is a point q and there is no output. In Step 2 of PPD_Convex, Bob receives $3N$ encryptions from Alice. Here, the simulator can simply generate $3N$ encryptions of zero. Given the assumption that the underlying encryption scheme is semantically secure, Bob can not distinguish these ciphertexts from the ones that are produced by Alice's real input. Similarly, Bob receives a number of encryptions in Steps 3, 4, and 9 of Sec_Comp. This is also simulated by multiple encryptions of zero.

5 Protocol Extensions

In this section, we present two improvements over our basic protocol. Section 5.1 introduces a method that enables Bob to control his location disclosure during protocol execution and Section 5.2 describes how to handle concave query polygons.

5.1 Limiting Bob's location disclosure

While the PPD_Convex protocol protects Bob's privacy when he lies outside Alice's proximity region, it may

potentially reveal sensitive information when he is located inside that region. The reason is that Alice could define a polygon that surrounds a very small area on the map, such as a hospital or a church, without proving to Bob that the area is larger than his preferred privacy threshold. To this end, we propose an enhanced version of the basic protocol that allows Bob to (blindly) compute the area of the proximity polygon, before agreeing to participate in the remainder of the protocol. The key observation is that in Step 1 of the basic protocol (Fig. 6) Bob has all the information he needs to compute the encrypted value of the polygon's area. Specifically, the area w of a convex polygon with N edges, when the edges (i, j) are processed in a counterclockwise order, is equal to

$$w = -\frac{\sum_{i=0}^{N-1}(p_i^y\, p_j^x - p_i^x\, p_j^y)}{2}.$$

Figure 8 shows the enhanced version of our protocol that enables Bob to participate in a proximity detection query, only if the area of the query is larger than a threshold W (Bob's location disclosure threshold). At Step 2, Bob computes $E_{\mathrm{P}}(2w)$, which he then masks (with random values r' and r'') in order to make it infeasible for Alice to manipulate the result (Step 3). He then sends the masked result to Alice, who decrypts it and returns the result back to Bob (Step 4). Next, Bob removes the random values and computes the polygon area w. If this area is larger than his privacy threshold W, the protocol continues; otherwise, Bob informs Alice that he is not willing to participate in this instance of the proximity detection query (Step 5).

Nevertheless, if Alice and Bob simply follow the remaining steps of the basic protocol (Steps 2–5 of PPD_Convex), Alice can cheat by forcing Bob to compute a value w that is larger than the actual query area. In particular, instead of sending ciphertext $E_{\mathrm{P}}(p_i^y\, p_j^x - p_i^x\, p_j^y)$ for each of the N edges (Step 1), Alice sends to Bob $E_{\mathrm{P}}(p_i^y\, p_j^x - p_i^x\, p_j^y - v)$, where v is a positive integer. In this way, the area w' that Bob computes is actually equal to $w + N \cdot v/2$. Clearly, Alice can make the query area appear arbitrarily large to Bob. Furthermore, Alice can still get the correct query result (for the smaller area w) by adding v in Step 3 of the secure comparison protocol (Fig. 7). That is, Alice computes $a = (s + r + v) \bmod 2^{\ell}$ and sends $E_{\mathrm{P}}(a)$ to Bob.

To mitigate this problem, we need to modify the protocol so that it is difficult for Alice to get a correct proximity result if she tries to cheat. To this end, when Bob computes the encrypted orientation value $E_{\mathrm{P}}(\theta_i)$ for a polygon edge, he multiplies it with a random value r_i that is chosen uniformly in the range $[1, d]$ (Step 6 of Fig. 8). Note that this does not affect the correctness of the protocol, as we only care about the sign of the result. However, this makes it difficult for Alice to remove the effect of v in the orientation result, since she has to (correctly) guess r_i and add $r_i \cdot v$ when computing a in the secure comparison protocol. Therefore, for a polygon with k edges, the probability that Alice computes the correct query result is $1/d^k$. Moreover, even if Alice guesses all k random values correctly, she still has no way of knowing whether the query result is correct. Consequently, the value of d can be very low, which is important since the bit length of $r_i \cdot \theta_i$ affects the performance of the secure comparison protocol.

PPD_Convex_Enhanced

Input: Alice has polygon P with N vertices $p_0, p_1, \ldots, p_{N-1}$
Bob has point q

Output: **true** if q in P, **false** otherwise

1. For $i \in \{0, 1, \ldots N-1\}$ and $j = (i+1) \bmod N$, Alice sends to Bob $E_{\mathrm{P}}(p_j^y - p_i^y)$, $E_{\mathrm{P}}(p_i^x - p_j^x)$, and $E_{\mathrm{P}}(p_i^y\, p_j^x - p_i^x\, p_j^y)$
2. Bob computes $E_{\mathrm{P}}(2w) = E_{\mathrm{P}}(-\sum_{i=0}^{N-1}(p_i^y\, p_j^x - p_i^x\, p_j^y))$ $= \prod_{i=0}^{N-1} E_{\mathrm{P}}(p_i^y\, p_j^x - p_i^x\, p_j^y)^{-1}$, where $j = (i+1) \bmod N$
3. Bob chooses r', r'' uniformly at random from $\mathbb{Z}_n^*$, computes the masked result $E_{\mathrm{P}}(r'(r'' + 2w)) = [E_{\mathrm{P}}(r'') \cdot E_{\mathrm{P}}(2w)]^{r'}$, and sends it to Alice
4. Alice decrypts the masked result and sends to Bob $r'(r'' + 2w)$
5. Bob retrieves w and, if $w \geqslant W$, the protocol continues; otherwise, Bob informs Alice that he does not wish to participate
6. For $i \in \{0, 1, \ldots, N-1\}$ and $j = (i+1) \bmod N$, Bob computes $E_{\mathrm{P}}(r_i \cdot \theta_i) = [E_{\mathrm{P}}(p_j^y - p_i^y)^{q^x} \cdot E_{\mathrm{P}}(p_i^x - p_j^x)^{q^y} \cdot E_{\mathrm{P}}(p_i^y\, p_j^x - p_i^x\, p_j^y) \cdot E_{\mathrm{P}}(-1)]^{r_i}$, where r_i is chosen randomly in $[1, d]$
7. Alice and Bob engage in a series of secure comparison protocols and Bob computes, $\forall i \in \{0, 1, \ldots, N-1\}$, $E_{\mathrm{P}}(\sigma_i)$, where $\sigma_i > 0$ if $r_i \theta_i \geqslant 0$ and $\sigma_i = 0$ if $r_i \theta_i < 0$
8. Bob chooses r uniformly at random from $\mathbb{Z}_n^*$, computes the masked result $E_{\mathrm{P}}(r \cdot \sigma) = [\prod_{i=0}^{N-1} E_{\mathrm{P}}(\sigma_i)]^r$, and sends it to Alice
9. Alice decrypts $E_{\mathrm{P}}(r \cdot \sigma)$ with her private key and, if $r \cdot \sigma = 0$, she returns **true**; otherwise, she returns **false**

Fig. 8 The enhanced protocol.

5.2 Handling concave polygons

In certain cases, the querier may need to define a concave polygon, in order to better approximate

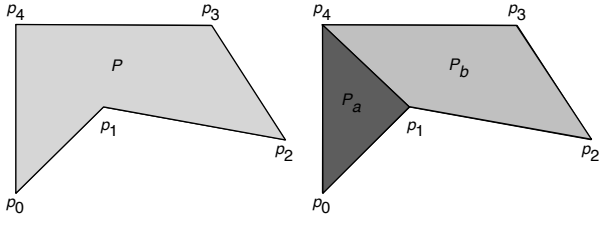
the underlying proximity region. The PPD_Convex protocol of Fig. 6 is not applicable in this scenario, as it may produce some false negatives. Nevertheless, using well-known algorithms for the optimal convex decomposition problem[20] in computational geometry, we can decompose any concave shape into the minimum number of convex polygons. An example is shown in Fig. 9, where P is decomposed into convex polygons P_a and P_b.

A straightforward algorithm to evaluate private proximity detection queries would then be to invoke PPD_Convex multiple times, and have Bob return all results (permuted) back to Alice. Alice would then decrypt the results and infer that Bob lies within P if and only if there is a zero value among the plaintexts. Furthermore, the permutation prevents Alice from identifying the exact polygon where Bob is located. While this approach would work in most cases, it is not secure if Bob lies on an edge that is shared between two convex polygons. In the example of Fig. 9, if Alice decrypts two zero values she can be certain that Bob lies somewhere along the line segment $\overline{p_1 p_4}$.

However, with a minor adjustment in our basic protocol, we can ensure that, when Bob lies on a common edge, only one of the neighboring polygons returns true in the proximity detection query. In particular, Bob has to modify the computation of the point orientation θ_i for one of the two polygons, such that it returns false when the point lies on that specific edge. This is done trivially, by removing $E_{\mathrm{P}}(-1)$ in Step 2 of the PPD_Convex protocol. In this way, a point that lies on that edge will produce a value of 0, instead of -1, and the query will return false. Note that, this modification also works for the enhanced version of our protocol.

6 Implementation Results

In this section, we present our results from an actual implementation of the PPD_Convex protocol on iOS devices. The implementation of the cryptographic primitives is written in C, and leverages the GMP (http://gmplib.org/) multiple precision arithmetic library and the OpenSSL (http://www.openssl.org/) cryptographic library. In particular, we cross compiled both libraries for the ARM architecture and incorporated them in our app. We deployed the app on two devices (an iPhone 6 and an iPad air) and connected the devices over a WiFi network using a secure SSL connection.



(a) Concave polygon (b) Convex decomposition

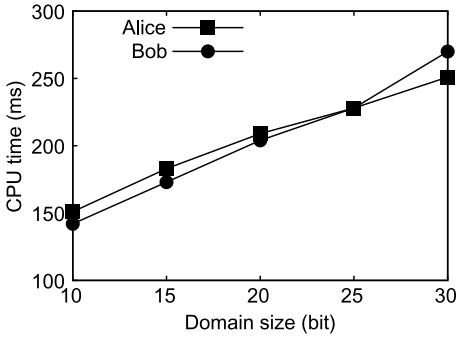
Fig. 9 Optimal convex decomposition.

Before running the actual protocol, we created a benchmark program to test the performance of the two homomorphic encryption schemes. Specifically, we deployed the benchmark app on the iPhone 6 device and measured the cost of the basic cryptographic operations. The results are shown in Table 1. The two modular exponentiation entries correspond to the size of the exponent, which is the deciding factor for the cost of this operation. Small exponents are involved when a party multiplies its plaintext input into an existing ciphertext, e.g., as Bob does in Step 2 of protocol PPD_Convex. Large exponents are normally necessary during a multiplicative masking operation, such as the one in Step 4 of PPD_Convex. The advantage of ElGamal's scheme is very clear in this table (as explained in Section 2.1), and justifies the usage of two different cryptosystems in the same protocol.

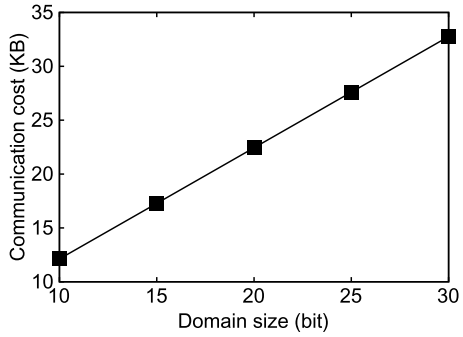
Next, we investigate the impact of the domain size (i.e., the number of bits required to store one coordinate) on the performance of a single point orientation computation. Figure 10 illustrates the CPU time required at the two parties, as well as the overall communication cost. Both costs grow linearly with the domain size, because the bit-size ℓ of the orientation result increases. This, in turn, increases the cost of the

Table 1 Cost of cryptographic primitives. (ms)

Paillier cryptosystem	
Encryption	17.14
Decryption	15.69
Modular exponentiation (small exponent)	0.52
Modular exponentiation (large exponent)	16.15
Modular multiplication	0.017
ElGamal cryptosystem	
Encryption	1.40
Decryption	0.96
Modular exponentiation (small exponent)	0.28
Modular exponentiation (large exponent)	1.38
Modular multiplication	0.01

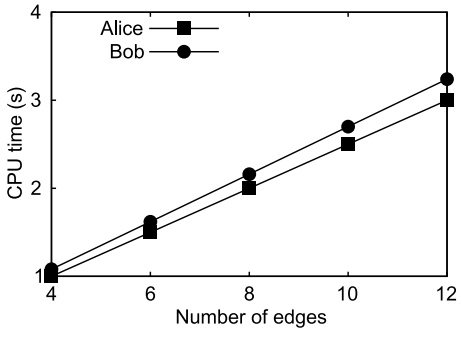


(a) CPU time

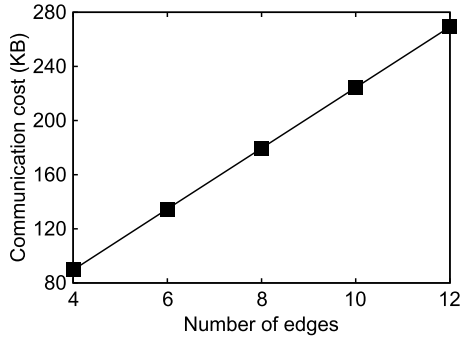


(b) Communication cost

Fig. 10 Cost vs. domain size (for a single edge).



(a) CPU time



(b) Communication cost

Fig. 11 Cost vs. number of polygon edges.

Sec_Comp protocol that has a linear complexity with respect to ℓ. Nevertheless, the CPU time is affected less than the communication cost, due to the dominant effect of the Paillier operations that are not influenced by the domain size. Also, Bob's CPU time increases more sharply than Alice's, because Bob needs to perform a lot of public key operations in Steps 6 and 7 of the Sec_Comp protocol.

Figure 11 shows the CPU time and the communication cost as a function of the number of edges N in the proximity region (for a domain size of 30 bits). Clearly, both costs scale linearly with N, since the proximity detection query involves exactly N point orientation computations. We expect that, in a real application, a rectangular region would probably be the most common query type. In this case, the query could be answered in around 2 s and incur a communication cost of 90 KB. We believe that this is an acceptable cost for privacy preserving query processing on handheld devices.

7 Conclusion

Traditional private proximity detection protocols are very restrictive in the definition of the proximity region. In particular, they typically constrain users to select (at most) a few cells from a fixed grid decomposition of the space. To this end, this paper extends the notion of private proximity detection, by allowing users to define regions of arbitrary convex shapes. We propose a novel solution, based on a secure two-party computation protocol that is provably secure. With a slight modification to our basic protocol, we show that it is also possible to protect the privacy of the query target, by enabling her to abort the protocol if the area of the query is below a desired threshold. Furthermore, we outline a solution that leverages our basic protocol to handle arbitrary concave polygons. Finally, we implement our protocol on handheld devices and illustrate its applicability in a real-life application.

Acknowledgment

This research was supported by the National Science Foundation CAREER Award IIS-0845262.

References

[1] G. Zhong, I. Goldberg, and U. Hengartner, Louis, Lester and Pierre: Three protocols for location privacy, in *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2007, pp. 62–76.

[2] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, Location privacy via private proximity

testing, in *Proceedings of NDSS*, 2011.

[3] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in *EUROCRYPT*, 1999, pp. 223–238.

[4] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[5] Y. Lindell and B. Pinkas, Secure multiparty computation for privacy-preserving data mining, *Journal of Privacy and Confidentiality*, vol. 1, no. 1, pp. 59–98, 2009.

[6] A. C.-C. Yao, How to generate and exchange secrets, in *Foundations of Computer Science, 1986, 27th Annual Symposium on*, 1986, pp. 162–167.

[7] M. Naor and B. Pinkas, Computationally secure oblivious transfer, *Journal of Cryptology*, vol. 18, no. 1, pp. 1–35, 2005.

[8] Y. Huang, D. Evans, J. Katz, and L. Malka, Faster secure two-party computation using garbled circuits, in *USENIX Security Symposium*, 2011.

[9] H. Lipmaa, Verifiable homomorphic oblivious transfer and private equality test, in *ASIACRYPT*, 2003, pp. 416–433.

[10] P. Ruppel, G. Treu, A. Küpper, and C. Linnhoff-Popien, Anonymous user tracking for location-based community services, in *Location- and Context-Awareness*. Springer Berlin Heidelberg, 2006, pp. 116–133.

[11] S. Mascetti, C. Bettini, and D. Freni, Longitude: Centralized privacy-preserving computation of users' proximity, in *Secure Data Management (SDM)*. Springer Berlin Heidelberg, 2009, pp. 142–157.

[12] L. Siksnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen, A location privacy aware friend locator, in *Advances in Spatial and Temporal Databases*. Springer Berlin Heidelberg, 2009, pp. 405–410.

[13] L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu, Private and flexible proximity detection in mobile social networks, in *2010 Eleventh International Conference on Mobile Data Management*, 2010, pp. 75–84.

[14] 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, *VLDB Journal*, vol. 20, no. 4, pp. 541–566, 2011.

[15] M. J. Atallah and W. Du, Secure multi-party computational geometry, in *Algorithms and Data Structures*. Springer Berlin Heidelberg, 2001, pp. 165–179.

[16] T. Thomas, Secure two-party protocols for point inclusion problem, *International Journal of Network Security*, vol. 9, no. 1, pp. 1–7, 2009.

[17] Y. Ye, L. Huang, W. Yang, and Y. Zhu, Efficient protocols for point-convex hull inclusion decision problems, *Journal of Networks*, vol. 5, no. 5, pp. 559–567, 2010.

[18] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, 2008.

[19] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, Privacy-preserving face recognition, in *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2009, pp. 235–253.

[20] J. M. Keil, Decomposing a polygon into simpler components, *SIAM Journal of Computing*, vol. 14, no. 4, pp. 799–817, 1985.

Spiridon Bakiras received the BS degree in electrical and computer engineering from the National Technical University of Athens in 1993, the MS degree in telematics from the University of Surrey in 1994, and the PhD degree in electrical engineering from the University of Southern California in 2000. Currently, he is an associate professor in the Department of Computer Science at Michigan Tech. Before that, he held teaching and research positions at the City University of New York, the University of Hong Kong, and the Hong Kong University of Science and Technology. His current research interests include database security and privacy, mobile computing, and spatiotemporal databases. He is a member of the ACM and a recipient of the U.S. National Science Foundation (NSF) CAREER award.

Bin Mu is currently a PhD student at the Graduate Center of the City University of New York (CUNY). He received the BS degree in computer science from Nankai University, China in 2007, the MS degree in geological information systems from Peking University, China in 2010, and MS in computer science from CUNY in 2014. His major research areas include privacy in location based services and privacy-preserving image search.