InnerCircle: A Parallelizable Decentralized Privacy-Preserving Location Proximity Protocol

Per Hallgren
Chalmers University of Technology, Sweden

Martín Ochoa
Siemens AG & Technische Universität München, Germany

Andrei Sabelfeld
Chalmers University of Technology, Sweden

***Abstract*—*Location Based Services* (LBS) are becoming increasingly popular. Users enjoy a wide range of services from tracking a lost phone to querying for nearby restaurants or nearby tweets. However, many users are concerned about sharing their location. A major challenge is achieving the privacy of LBS without hampering the utility. This paper focuses on the problem of *location proximity*, where principals are willing to reveal whether they are within a certain distance from each other. Yet the principals are *privacy-sensitive*, not willing to reveal any further information about their locations, nor the distance. We propose *InnerCircle*, a novel secure multi-party computation protocol for location privacy, based on partially homomorphic encryption. The protocol achieves precise fully privacy-preserving location proximity without a trusted third party in a single round trip. We prove that the protocol is secure in the semi-honest adversary model of Secure Multi-party Computation, and thus guarantees the desired privacy properties. We present the results of practical experiments of three instances of the protocol using different encryption schemes. We show that, thanks to its parallelizability, the protocol scales well to practical applications.**

I. Introduction

Location Based Services (LBS) are becoming increasingly popular. The ubiquity of mobile interconnected devices creates tremendous opportunities for services that utilize location information. A single online resource features 2206 companies within LBS at the time of writing [1]. Logistics companies make extensive usage of tracking the location of cargo throughout the land, sea, and air. Enforcement authorities use location tracking technology for devices carried by people and embedded in vehicles. Individual users enjoy a wide range of location-based services from tracking a lost phone to querying for nearby restaurants or nearby tweets.

Location Privacy Challenge Unfortunately, privacy-sensitive location information of end users is commonly sent to the LBS. The privacy of users is often neglected, perhaps not surprisingly as there are no readily available privacy-preserving solutions to the problem at hand. An illustrative *trilateration* attack on a dating service has been detailed by Include Security [31], revealing exact position of a chosen user. In a similar vein, the smartphone app *Girls around me* allowed users to find other users (profiled as female) who recently had checked in on Foursquare [4]. Deemed as a serious privacy violation, the app had since been banned from using the Foursquare API and removed from the app store. Recent research systematizes these attacks and identifies a number of LBS where it is possible to reveal the user' location even if some of the LBS approximate and obfuscate distance information [27], [22].

There is fundamental tension between utility and privacy: privacy can be achieved by keeping location information secret but this may result in rendering LBS useless. A major challenge is to address the privacy of LBS without hampering the utility of the services [19], [30].

Privacy in Location Proximity This paper focuses on the problem of *location proximity*, where principals are willing to reveal whether they are within a certain distance from each other. Yet the principals are *privacy-sensitive*, not willing to reveal any further information about their locations, nor the exact distance. Both *mutual location proximity*, when the result of a proximity check is shared between the two participating principals, and *one-way location proximity*, when only one principal finds out whether the other principal is in the proximity, are important scenarios.

Mutual location proximity is useful for collision prevention for vehicles, vessels, and aircraft when revealing their exact location is undesirable. Another example is discovering friends in the vicinity [33], [32], without finding out the location of the friends or distances among them. *One-way location proximity* is of interest for discovery of nearby people (e.g., doctors and police officers) without giving out the principal's location. The asymmetric case of friend discovery also falls under this category with one principal checking if there are friends nearby without the friends learning the result of the proximity check.

The current practice is that a third party is trusted to handle principals' locations for scenarios as above. However, privacy concerns call for avoiding trust to third parties. In line with the efforts on decentralizing certificate authorities [7], [21] and the Internet itself [34], our goal is a *decentralized* solution for the privacy-preserving proximity problem.

Figure 1 illustrates the general scenario. Principal *Alice* (A) wants to know if principal *Bob* (B) is within a certain distance. While *Bob* is allowed to find out that *Alice* is interested in knowing if he is in her proximity, the goal is that *Bob* learns nothing else about *Alice*. Similarly, the goal is that *Alice* learns nothing about *Bob*'s location other than knowing if he is in the proximity.

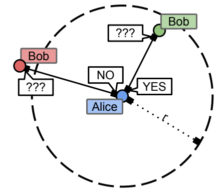

Fig. 1. Privacy-preserving location proximity

Decentralized Privacy-Preserving Location Proximity Motivated by the challenge of location privacy for the proximity problem, we set out to provide unhampered functionality without compromising privacy through means of *secure multi-party computation (SMC)*, where participants can jointly compute a function based on private inputs.

Attacker Model Our assumption is that A and B are *honest but curious*, in the sense they follow the format of the protocol but may log all messages and attempt to infer some further knowledge. We do not protect against attacks parties provide fake coordinates. These attacks are orthogonal, and can be mitigated by using tamper-resistant location devices or strategies such as *location tags* [25].

Single- vs. Multi-run security As is common [35], [9], [24], [33], [32], this paper focuses on the security of one run of the protocol. Re-running a precise proximity protocol may allow trilaterating the location of the users, as in the Include Security attack [31]. In general, multi-run security is a difficult problem which calls for additional countermeasures, and is a worthwhile subject to future studies. Collusion, where multiple parties run a single run of the protocol is in our setting, from *Bob*'s point of view, equivalent to one principal re-running the protocol. We remark that our framework readily provides multi-run security for one-way location proximity when the protocol-initiating principal is statically positioned (e.g., a user stationed at a coffee shop looking for nearby friends). In this case the static principal's input into the protocol is supplied once and for all runs, breaking a necessary prerequisite for trilateration.

Discretization degree Our goal is to provide exact proximity result given a chosen metric and unit measurement, rather than approximating the result. An approach which is not precise up to the unit of the coordinate system can have both false negatives and false positives. Consider Figure 2, which visualizes the worst case of an approach where A 's proximity is approximated by the gray cell, the approach considers the top-right B nearby, but the bottom-left B far. Even if A 's position is in the center of such a grid, one can only exclude either false positives or false negatives, but not both. Narayanan et al. [25] discuss how a grid-based approach can be useful for multi-run security. However a grid-based solution has weaknesses to a multi-run attacker when crossing cell boundaries [6].

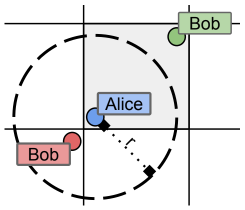

Fig. 2. Grid-based testing

As most of the related work, we assume a Euclidian plane, which is a a reasonable local approximation for most applications. Approximations that take the curvature of the Earth into consideration can be performed in our setting, as outlined in the full version of the paper [14].

Parallelizability As efficiency is one of our goals, it is desirable to make use of parallelizability. Generic SMC solutions are not readily suitable to parallel execution. In contrast, we set out to design a protocol that can benefit from parallelization.

Contributions The paper proposes *InnerCircle*, a novel decentralized protocol for privacy-preserving location proximity. In contrast to most of the related work, we fully dispense with any third parties. Moreover we require only one round trip using a parallelizable algorithm. Further, we do not degrade the protocol by approximating the principals' positions by grid blocks. Instead, we offer full precision for the chosen coordinate system and the Euclidean metric.

The protocol's key phases are distance and proximity calculation (see Section II). Essentially, A provides an encrypted (under her public key) aggregate that allows B to homomorphically compute the encrypted distance. Then a novel technique for homomorphically computing "less than" without any roundtrips between the participants is used to estimate proximity within a public range r. As discussed in Section IV, implementations of the same functionality using state-of-the-art generic SMC approaches with a one roundtrip protocol are significantly less efficient than our solution for a wide range of practical applications. In Section III we discuss the security of *InnerCircle* for the standard definitions for semi-honest adversaries in SMC [12].

We report on practical experiments with a prototype implementation in Section IV. Asymptotic complexity results are reported in the full version [14]. This allows us to study the performance of the protocol for various homomorphic ciphers under different key sizes and different proximity ranges. We empirically show the effectiveness of the parallelization strategy by running our benchmarks on a multi-core machine with different configurations. Our evaluation indicates that the protocol scales well to practical applications.

II. Protocol Description

This section describes *InnerCircle* in detail, for an unspecified additively homomorphic encryption scheme. The protocol consists of one roundtrip, where *Alice* sends one message to *Bob*, to which he responds with a boolean value encoded inside a list of randomized ciphertexts telling *Alice* whether she is close to *Bob* or not. First, *Alice* uses IC_A to construct the content of her message to *Bob*. *Bob* then creates the proximity result through a procedure IC_B, which defines the second message of the protocol. IC_B makes use of two procedures L_2 and $\mathsf{lessThan}$, computing the distance and proximity result, respectively. The proximity result is encoded within an encrypted and shuffled list, which contains exactly one zero (after decryption) if the distance is less than the queried range r, and no zero otherwise. Finally, a procedure inProx is defined to show how *Alice* converts the answer array into a boolean value.

The protocol description (Figure 3) is given in pWHILE [3]. For the convenience of the reader a few constructs used in the paper are outlined here, but for details the reader is directed to [3]. $a \leftarrow b$ means assigning a value b to a variable a, while $a \xleftarrow{\$} [0..n]$ means assigning a random value between 0 and n to a. $L :: l$ denotes appending the item l to the list L.

Additively Homomorphic Encryption Schemes A homomorphic cryptosystem allows to evaluate some functions of plaintexts while only holding knowledge of their corresponding ciphertexts and the public key. This feature is central for the construction of *InnerCircle*, further it is required that the cryptosystem is semantically secure. For a standard definition of semantic security see [14].

In the following, k and K is the private/public key pair of *Alice*. For the purpose of this paper, let the plaintext space

$\mathcal{M}$ be isomorphic to the ring $(\mathbb{Z}_m, \cdot, +)$ for some m and the ciphertext space $\mathcal{C}$ such that encryption using public key K is a function $E_K : \mathcal{M} \rightarrow \mathcal{C}$ and decryption using a private key k is $D_k : \mathcal{C} \rightarrow \mathcal{M}$. The vital homomorphic features which will be used later in the paper is addition function $\oplus$, a unary negation function $\neg$, a multiplication function $\odot$ and a randomizing function $\mathcal{R}$, as defined in Equations (1-4). For readability, these functions as well as the encryption and decryption functions E and D are not indexed with the keys used in the operation, however it is assumed that K is available when the respective function is computed and that k is available to *Alice* for decryption. The $\ominus$ symbol is used in the following to represent addition by a negated term, that is, $c_1 \oplus \neg c_2$ is written as $c_1 \ominus c_2$.

$$E(m_1) \oplus E(m_2) = E(m_1 + m_2) \tag{1}$$
$$\neg E(m_1) = E(-m_1) \tag{2}$$
$$E(m_1) \odot m_2 = E(m_1 \cdot m_2) \tag{3}$$
$$\mathcal{R}(E(m_1)) = \begin{cases} E(0) & \text{if } m_1 = 0 \\ E(l) & \text{otherwise} \end{cases} \tag{4}$$

Where $m_1 \in \mathcal{M}$, $m_2 \in \mathcal{M}$ and l is a (uniform) random element in $\mathcal{M} \setminus \{0\}$.

A. Distance Calculation

This section explains how *InnerCircle* makes use of additive homomorphism to compute the distance between two principals without sending unencrypted coordinates to either party. Zhong et al. [35] present three protocols which all include a step in which the distance between two principals is computed homomorphically. This process has been distilled into the following formulae.

The euclidean distance between two points (x_A, y_A) and (x_B, y_B) is computed as $d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$. The square of d is thus:

$$D = d^2 = x_A^2 + x_B^2 + y_A^2 + y_B^2 - (2x_A x_B + 2y_A y_B)$$

By the additive (Equation (1)), negation (Equation (2)) and multiplicative (Equation (3)) properties of the cryptosystem, D can be computed homomorphically as shown in Equation (5), separating the principals respective input.

$$\begin{aligned} E(D) = & E(x_A^2 + y_A^2) \oplus E(x_B^2 + y_B^2) \\ & \ominus ((E(2x_A) \odot x_B) \oplus (E(2y_A) \odot y_B)) \end{aligned} \tag{5}$$

A procedure $\mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$ through which *Bob* computes the (encrypted) distance given three ciphertexts $a_0 = E(x_A^2 + y_A^2)$ and $a_1 = E(2x_A)$ and $a_2 = E(2y_A)$ from *Alice* and his own coordinates (x_B, y_B) is now defined as:

```
Procedure $\mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$ :
$E(D) \leftarrow a_0 \oplus E(x_B^2 + y_B^2) \ominus ((a_1 \odot x_B) \oplus (a_2 \odot y_B))$ ;
return $E(D)$;
```

B. Proximity Calculation

As *Bob* is unwilling to disclose his position and his distance to *Alice*, he must compute the proximity result so that it reveals no such information. This section details how this is accomplished in *InnerCircle*. The procedure to compute the privacy-preserving proximity result consists of two parts, where the first part makes use of the obfuscation method used in the *Pierre* protocol [35].

First the squared distance is subtracted by each value from 0 to the threshold r^2. The result is randomized using $\mathcal{R}()$ and stored in a list. Each separate list element is thus a random number except for when the subtraction results in a zero (in which case the list element contains a zero). The number of zeroes in the list is either zero or one. Second, the content of the list must also be shuffled, to make sure that the position in the list which produces a zero is not leaked to *Alice* as a part of the result. Note that since the encryption scheme is semantically secure, *Bob* can not deduce any information about the plaintext while constructing this list. This is formalized as a generic procedure $\mathsf{lessThan}(x, y)$ below, which homomorphically computes an intermediate value that can be used by *Alice* to decide whether x is less than y without learning x. This function is used to compute whether D is less than r^2 in the final protocol:

```
Procedure $\mathsf{lessThan}(x, y)$ :
$L \leftarrow [\,]$;
for $i \leftarrow 0$ to $i \leftarrow y - 1$ :
    $l \leftarrow \mathcal{R}(x \ominus E(i))$ ;
    $L \leftarrow L :: l$;
return $\mathsf{shuffle}(L)$;
```

$\mathsf{shuffle}$ returns a uniformly shuffled copy of its input. The following lemma follows directly from the definition of $\mathcal{R}$ and the $\mathsf{shuffle}$ function:

Lemma 1: If $x, y \geq 0$ then the output of $\mathsf{lessThan}(x, y)$ is determined by the inputs as follows. *Case $x \geq y$:* A list drawn uniformly at random from the set of all lists of length y such that all elements are different from zero. *Case $x < y$:* A list drawn uniformly at random from the set of all lists of length y such that all elements are different from zero *except exactly one*.

Note that for this lemma to hold, it is crucial that $\mathcal{R}$ randomizes its input uniformly, which is not the case for all instantiations of the cipher (for a discussion see [14]).

The list returned from $\mathsf{lessThan}$ are sent by *Bob* to *Alice*. *Alice* decrypts the list and can conclude that if any element is equal to zero, it means that $D < r^2$, which in turns means that she and *Bob* are within r of each other.

C. Protocol

The protocol is formally described in Figure 3. *Alice* must send three separate ciphertexts to *Bob* as depicted through IC_A in Figure 3-1. From this *Bob* computes the distance between *Alice* and himself, encrypted under *Alice*'s public key. *Bob* applies the $\mathsf{lessThan}$ algorithm to the distance, in effect making the response binary, using IC_B seen in Figure 3-2. Finally, *Alice* sees either noise or a zero, encrypted using her public key, after she analyzes the result using a simple procedure inProx described in Figure 3-3.

III. Privacy Considerations

This section details some of *InnerCircle's* requirement and its privacy properties. Authentication is outside the scope of

```
1. Proc. $\mathsf{IC}_A(x_A, y_A)$ :
     $a_0 \leftarrow E(x_A^2 + y_A^2)$;
     $a_1 \leftarrow E(2x_A)$;
     $a_2 \leftarrow E(2y_A)$;
     return $a_0, a_1, a_2$;

3. Procedure $\mathsf{inProx}(L)$ :
     for $i \leftarrow 0$ to $i \leftarrow r^2$ :
         if $D(L[i]) = 0$ then :
             return 1
     return 0

2. Proc. $\mathsf{IC}_B(x_B, y_B, a_0, a_1, a_2)$:
     $D \leftarrow \mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$ ;
     $L \leftarrow \mathsf{lessThan}(D, r^2)$ ;
     return $L$
```

Fig. 3. The procedures of *InnerCircle*

this paper, but can easily be solved by using e.g. SSL with mutual certificate authentication.

Intuitively, the goal of a privacy-preserving proximity protocol is to allow *Alice* to learn whether she is in the proximity of *Bob*, without either of the parties having to disclose their position or the exact distance between them, and preventing third parties from learning anything from the protocol execution. This is precisely captured by the standard definitions of secure multi-party computation in the semi-honest adversarial model [12], [23], which guarantee that involved parties learn only a *functionality*, jointly computed from the parties private inputs. For simplicity of exposition, in the following we assume that parameters such as public keys and the proximity threshold r are considered to be previously known by both parties.

Formally, let $x_1, \dots, x_p$ be the inputs of the p parties. Then a function that specifies the intended output f_i for each party $f(x_1, \dots, x_p) = (f_1(x_1, \dots, x_p), \dots, f_p(x_1, \dots, x_p))$ is called the *functionality*.

Definition 1 (Privacy, [23]): Given a deterministic functionality f, a protocol π computes it privately in the semi-honest adversarial model if there exist probabilistic algorithms S_i for $i = 1, \dots, p$ such that:

$$\{S_i(x_i, f_i(x_1, \dots, x_p))\} \overset{c}{\equiv} \{view_i^\pi(x_1, \dots, x_p)\}$$

Where $view_i^\pi(x_1, \dots, x_p) = (r_i, x_i, m_1, \dots, m_t)$, r_i represents the coin tosses made by party i in a normal execution of the protocol for inputs $x_1, \dots, x_p$. $m_1, \dots, m_t$ are the messages observed by party i during the execution of the protocol and $\overset{c}{\equiv}$ denotes computational indistinguishability of distributions.

In other words, a protocol is secure with respect to its functionality if we can completely simulate what a party would see in a normal run of the protocol just using its input and the intended output to that party. From this it immediately follows that the parties can not learn more than their inputs and their intended outputs. For a detailed recap of *Negligible functions* and *Indistinguishability*, we refer the reader to [14].

Therefore to show that our protocol is secure in the semi-honest adversary setting, we need to construct so-called *simulators* for each party in the computation. These simulators are non-deterministic algorithms that by construction only receive the private inputs of a given party (and not the others) and its intended output as defined by the functionality, and such that their resulting output is computationally indistinguishable from a real protocol run.

Instantiation for Proximity Testing

In the case of proximity protocols, the desired functionality is: $f((x_A, y_A), (x_B, y_B), (x_C, y_C)) = (d, \lambda, \lambda)$, where λ is an empty string (*Bob* and third parties learn nothing) and:

$$d = \begin{cases} 1 & \text{if } (x_A, y_A) \in \mathsf{prox}(r, (x_B, y_B)) \\ 0 & \text{otherwise} \end{cases}$$

Here $\mathsf{prox}(r, (x_B, y_B))$ is a connected set whose area is a function of r and which contains (x_B, y_B). This can be a disc of radius r centred in (x_B, y_B), as depicted in Figure 1, a square of side r as in [24], or a hexagon as in [25].

Theorem 1 (Privacy guarantee): InnerCircle computes the location proximity functionality f privately according to Definition 1.

A key observation is that the view of *Alice* can be simulated based on the value of d independently of the exact coordinates of *Bob*. The views of *Bob* and *Claire* are essentially easy to simulate due to the semantic security of the encryption mechanism. The proof of the theorem can be found in [14].

Towards automatic verification The above statements are amenable to semi-automatic verification. Similar proofs for semi-honest adversaries were constructed in [2]. Although automatic verification is outside the scope of this paper, our definitions and proof strategy is an initial step in this direction.

IV. Implementation

This section reports on an evaluation of prototypes of *InnerCircle* written in Python and compares the results against alternative approaches that yield the same functionality.

General state-of-the-art SMC frameworks are able to implement any protocol, the motivation behind creating a special-purpose solution is to improve performance over these. Therefore, our benchmarks focus on comparing processing time. Generic and efficient implementations of SMC are openly available, which facilitate our experiments.

Three cryptosystems are used, which, under certain assumptions on the inputs, respect the conditions of Section II: these are Paillier [26] and two variants of ElGamal [10]. The first variant of ElGamal is referred to as ElGamal$_\mathbb{Z}$ and uses a group in the integers, the second one is referred to as ElGamal$_{ECC}$ and uses a group in elliptic curve cryptography [16]. Details about instantiating the protocol using these three schemes can be found in [14]. For Paillier, the protocol is secure only under some input restrictions. As Paillier uses an RSA modulus $n = p \cdot q$ with p and q prime, the used coordinates must be less than both q and p to be secure against a curious *Alice*. For an honest *Alice* on earth, this should hold, as discussed in [14]. The results are shown in Figure 4. Details and further experiments can be found in [14].

An example of a location-based service is when users want to get notified when a set of principals arrive at a location, but only want to use the feature for user-specified periods of time , along the lines of Glympse [11]. If this service grants a precision of 10 meter when the area polled is 500 meters wide, it could use *InnerCircle* for better results than when using any competitor. Using TASTY, the time taken to execute the protocol is 411 ms, while using *InnerCircle* with ElGamal$_\mathbb{Z}$ gives a response time of 211 ms.

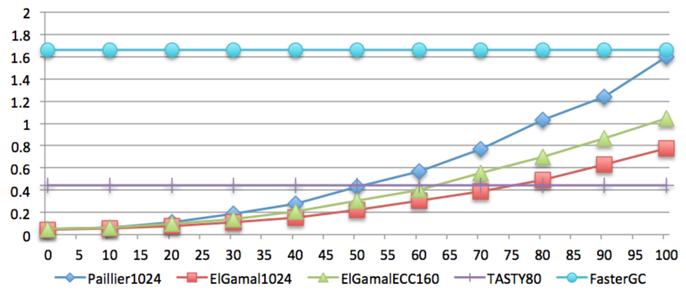

Fig. 4. Time consumption for different algorithms and values of r using 80 bits of security

V. Related Work

The two main solutions within SMC are homomorphic encryption and garbled circuits. Section IV compared *InnerCircle* to FasterGC by Huang et al. [15], which is a prominent work in the field of garbled circuits. The benchmarks show that for this particular functionality garbled circuits on their own do not catch up to an approach based on homomorphic encryption. This has been shown also for privacy-preserving face recognition [28]. In general it is hard to determine which approach is suitable for a specific application [20], [18]. Our comparison to TASTY (which uses both Garbled Circuits and homomorphic encryption, see Section IV) shows that an implementation based merely on homomorphic encryption such as ours can be better for practical applications.

Orthogonal to SMC, there is a large body of work in the overall area of privacy for location-based services. We refer the readers to the surveys by Krumm [19] and Terrovitis [30] for an overview. The following focuses on the most closely related work on the proximity problem.

A recent work by Costantino et al. [5] solves a different problem with similar methods. In this work, the setting is a network where people with similar interests want to share information. To compute similarity, interest integer vectors of size n are compared in each dimension This is a generalisation of the proximity problem to multiple dimensions. The paper discusses the utility of variations of such metrics, where not all values are compared, but only a random subset.

An important source of inspiration for this work is the *Louis* and *Pierre* protocols by Zhong et al. [35]. The *Louis* protocol computes precise distances using additive homomorphism in the same manner as described in Section II-A, but uses a third party to check whether the principals are within r from each other. The *Pierre* protocol obfuscates specific distance comparisons similarly to Section II-B, however, it does not incorporate a general inequality method and maps the principals coordinates to a grid.

There are several published works about testing proximity privately by concealing locations through cloaking the users position within a partition of the plane, called a *granule* [9], or a set of granules. However, these approaches lead to false positives and false negatives. In some cases over 66% of reported positives can be false [24]. We discuss the most prominent approaches in relation to *InnerCircle*.

Šeděnka and Gasti [29] homomorphically compute distances using the UTM projection, ECEF (Earth-Centered Earth-Fixed) coordinates, and using the Haversine formula. Haversine and ECEF are both useful when considering the curvature of the earth. Results using these three distance functions are combined with both the inequality function from Erkin et l. [8], and using garbled circuits using a technique from a work by Kolesnikov et al. [17]. *InnerCircle* is less resource-consuming both in terms of bandwidth and processing time when r is not large, and requires fewer round trips to complete the protocol. As argued above, small values of r makes sense for many practical applications of proximity testing. Being a recent and prominent work, an effort to compare the performance of this work and *InnerCircle*, which showed that it has similar performance to TASTY. The results are expanded in [14].

TABLE I. Comparison of proximity protocols

Protocol	Precise	Decentralized	Fully Privacy-preserving	Single Round-trip
Narayanan 2 [25]				
Narayanan 1,3 [25]		X		X
Pierre[35]		X		X
Louis[35]	X		X	
Lester[35]	X	X		X
Hide&Crypt[9]				
C-Hide&Hash[24]	X		X	
FriendLocator[33]			X	
VicinityLocator[33]	X		X	
PP-[HS,UTM,ECEF][29]	X	X	X	
InnerCircle	X	X	X	X

The *Hide&Crypt* protocol by Mascetti et al. [9] consists of two steps. The first step is a filtering done between a third party and the initiating principal. The next step uses a more fine grained granularity, and is executed between the two principals. In both steps, the granule which a principal is located is sent to the other party. *C-Hide&Hash*, also by Mascetti et al. [24], is a centralized protocol, where the principals do not need to communicate pairwise but otherwise share many aspects with *Hide&Crypt*.

FriendLocator by Šikšnys et al. [33] presents a centralized protocol where clients map their position to different granularities, similarly to *Hide&Crypt*, but instead of refining via the second principal each iteration is done via the third party. VicinityLocator, also by Šikšnys et al. [32] is an extension of FriendLocator, which allows the proximity of a principal to be represented not only in terms of squares, but instead can have any shape.

Narayanan et al. [25] present three protocols for location proximity. The first two make use of private equality testing to find whether three hexagons are overlapping, however with the second protocol being centralized. The third makes use of private set intersection to compute whether the two principals has an overlap in *location tags*, which makes it very hard for attackers to spoof their location, but rely severely on how much data can be collected about the environment (through WiFi, GPS, Bluetooth, etc).

In Table I each protocol is classified as precise, decentralized, fully privacy-preserving and the number of round-trips needed to conclude the protocol. Justifications for Table I can be found in the full version of the paper[14]. By *precise* is meant that granted enough computational power, the protocol can give proximity verdicts without false positives and false negatives, down to the precision of the coordinate system. A *decentralized* protocol does not rely on a third party. A *fully privacy-preserving proximity protocol* conforms to the security definitions of Section III.

It is concluded that *InnerCircle* is the only ad-hoc current protocol to uphold all four properties. Note that due to the restricted availability of prototypes implementing the related work, we were not able to benchmark the protocols in Table I, and we restricted ourselves to the comparison presented previously against generic SMC (which has most features in common with our approach).

VI. Conclusions

We have proposed InnerCircle, a parallelizable protocol which achieves fully privacy-preserving location proximity without a trusted third party in a single round trip.

Our experiments show that compared to other solutions InnerCircle excels when the queried radius is small and when many threads can be executed in parallel. The former is a common case in the scenario of geofencing [13]. The latter two means that the usability of the protocol will be boosted by future hardware development. The key size necessary to preserve privacy inherently grows over time, and processors gain most of their processing capacity from more cores, rather than from an increase in CPU frequency.

Our future work is on protection against stronger attackers that tamper with the message format and re-run the protocol.

Acknowledgments Thanks are due to Jorge Cuellar for generous feedback, to Daniel Hedin for insights on cryptographic games, to Dan Boneh for pointers to related work, and to Enrico Lovat for optimization ideas. This work was funded by the European Community under the ProSecuToR and WebSand projects and the Swedish agencies SSF and VR.

References

[1] AngelList. Location based services startups, Mar. 2015. https://angel.co/location-based-services.

[2] G. Barthe, G. Danezis, B. Gregoire, C. Kunz, and S. Zanella-Beguelin. Verified computational differential privacy with applications to smart metering. In *CSF*, 2013.

[3] G. Barthe, B. Grégoire, and S. Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *POPL*, 2009.

[4] D. Coldewey. "Girls Around Me" creeper app just might get people to pay attention to privacy settings. TechCrunch, March 2012.

[5] G. Costantino, F. Martinelli, and P. Santi. Investigating the privacy versus forwarding accuracy tradeoff in opportunisticinterest-casting. *IEEE Trans. Mob. Comput.*, 13(4):824–837, 2014.

[6] J. Cuéllar, M. Ochoa, and R. Rios. Indistinguishable regions in geographic privacy. In *SAC*, 2012.

[7] P. Dewan and P. Dasgupta. P2P reputation management using distributed identities and decentralized recommendation chains. *IEEE Trans. Knowl. Data Eng.*, 22(7):1000–1013, 2010.

[8] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *PETS*, 2009.

[9] D. Freni, C. R. Vicente, S. Mascetti, C. Bettini, and C. S. Jensen. Preserving location and absence privacy in geo-social networks. In *CIKM*, 2010.

[10] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[11] Glympse Inc. Glympse. https://www.glympse.com/, 2015. [Online; accessed 01-February-2015].

[12] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.

[13] A. Greenwald, G. Hampel, C. Phadke, and V. Poosala. An economically viable solution to geofencing for mass-market applications. *Bell Labs Technical Journal*, 16(2):21–38, 2011.

[14] P. Hallgren, M. Ochoa, and A. Sabelfeld. InnerCircle: A Parallelizable Decentralized Privacy-Preserving Location Proximity Protocol. http://www.cse.chalmers.se/~andrei/innercircle-full.pdf, 2015. Full version.

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

[16] N. Koblitz. Elliptic curve cryptography. *Mathematics of Computation*, 48(177), 1987.

[17] V. Kolesnikov, A. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS*, 2009.

[18] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. *Journal of Computer Security*, 21(2):283–315, 2013.

[19] J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[20] R. L. Lagendijk, Z. Erkin, and M. Barni. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Process. Mag.*, 30(1):82–105, 2013.

[21] N. Leavitt. Internet security under attack: The undermining of digital certificates. *IEEE Computer*, 44(12):17–20, 2011.

[22] M. Li, H. Zhu, Z. Gao, S. Chen, L. Yu, S. Hu, and K. Ren. All your location are belong to us: breaking mobile social networks for automated user location tracking. In J. Wu, X. Cheng, X. Li, and S. Sarkar, editors, *MobiHoc*, 2014.

[23] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.

[24] 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 J.*, 20(4):541–566, 2011.

[25] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

[26] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

[27] G. Qin, C. Patsakis, and M. Bouroche. Playing hide and seek with mobile dating applications. In *SEC*, 2014.

[28] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *ICISC*, 2009.

[29] J. Sedenka and P. Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In *ASIACCS*, 2014.

[30] M. Terrovitis. Privacy preservation in the dissemination of location data. *SIGKDD Explorations*, 13(1):6–18, 2011.

[31] M. Veytsman. How i was able to track the location of any tinder user, February 2014. Web resource: http://blog.includesecurity.com/.

[32] L. Šikšnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu. Private and flexible proximity detection in mobile social networks. In *Mobile Data Management*, 2010.

[33] L. Šikšnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen. A location privacy aware friend locator. In *SSTD*, 2009.

[34] X. Zhang, H. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen. SCION: scalability, control, and isolation on next-generation networks. In *S&P*, 2011.

[35] G. Zhong, I. Goldberg, and U. Hengartner. Louis, lester and pierre: three protocols for location privacy. In *PETS*, 2007.