Paper **One**

InnerCircle
A Decentralized Privacy-Preserving Location Proximity Protocol

Per A. Hallgren, Martìn Ochoa and Andrei Sabelfeld

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.

Published in the 13th Conference on Privacy, Security and Trust
PST 2015, Izmir, Turkey, July 21-23, 2015
Revised version

1 Introduction

Location Based Services (LBS) are becoming increasingly popular. A single online resource features 2206 companies within LBS at the time of writing [25]. The ubiquity of mobile interconnected devices creates tremendous opportunities for services that utilize location information. 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 [37], 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 [33, 23].

There is fundamental tension between utility and privacy. Privacy can always 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, 36].

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 [39, 38], 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 [8, 21] and the Internet itself [41], 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.

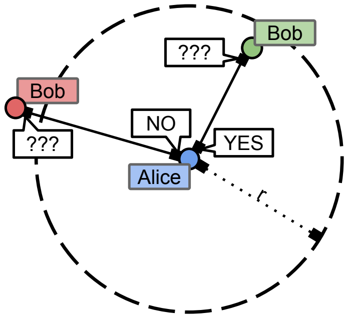


Fig. 1. Privacy-preserving location proximity

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.

While practical privacy-preserving techniques often make use of pragmatic approaches to obfuscate location data, they often fall short to provide rigorous privacy guarantees such as the ones provided using SMC techniques. Bridging the gap between rigorous and practical is an important motivation for our work.

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* [28].

Single- vs. Multi-run security As is common [42, 10, 27, 39, 38], 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 [37]. 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. [28] 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].

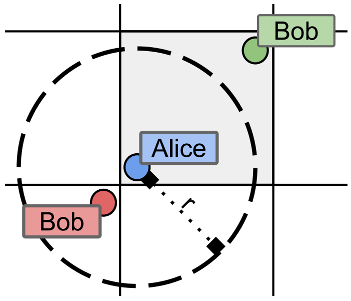


Fig. 2. Grid-based testing

As most of the related work, we assume a Euclidean 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 Appendix 3.

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 2). 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 4, 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 3 we discuss the security of *InnerCircle* for the standard definitions for semi-honest adversaries in SMC [13].

We report on practical experiments with a prototype implementation in Section 4. The implementation 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. Asymptotic complexity results are reported in Appendix 6. Our evaluation indicates that the protocol scales well to practical applications.

2 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 [2]. For the convenience of the reader a few constructs used in the paper are outlined here, but for details the reader is directed to [2]. $a \leftarrow b$ 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 Appendix 1.

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 $E_K : \mathcal{M} \to \mathcal{C}$ and decryption using a private key k is $D_k : \mathcal{C} \to \mathcal{M}$. 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 $c_1 \ominus c_2$.

$$E(m_1) \oplus E(m_2) = E(m_1 + m_2) \tag{1}$$

$$\neg E(m_1) = E(-m_1) \tag{2}$$

$$E(m_1) \odot m_2 = E(m_1 \cdot m_2) \tag{3}$$

$$\mathcal{R}(E(m_1)) = \begin{cases} E(0) \text{ if } m_1 = 0 \\ E(l) \quad \text{otherwise} \end{cases} \tag{4}$$

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

2.1 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. [42] 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.

$$E(D) = E(x_A^2 + y_A^2) \oplus E(x_B^2 + y_B^2) \ominus ((E(2x_A) \odot x_B) \oplus (E(2y_A) \odot y_B)) \quad (5)$$

A procedure $\mathsf{L}_2\,(x_B, y_B, a_0, a_1, a_2)$ through which *Bob* computes the encrypted and squared distance given three ciphertexts $a_0 = E(x_A^2 + y_A^2)$, $a_1 = E(2x_A)$ 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 $E(D)$;
```

2.2 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 [42].

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 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 shuffle$(L)$;
```

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

Lemma 1. *If $x, y \geq 0$ then the output of 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 Appendix 4).

The list returned from 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.

2.3 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 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.

3 Privacy Considerations

This section details some of *InnerCircle's* requirement and its privacy properties. Authentication is outside the scope of this paper, but can easily be solved by using e.g. SSL with mutual certificate authentication.

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

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

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

Fig. 3. The procedures of *InnerCircle*

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 [13, 24], 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 $x_1, \ldots, x_p$ be the inputs of the p parties. Then a function f that specifies the intended output f_i for each party is called the *functionality*:

$$f(x_1, \ldots, x_p) = (f_1(x_1, \ldots, x_p), \ldots, f_p(x_1, \ldots, x_p))$$

Definition 1 (Privacy, [24]). *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 $i = 1, \ldots, p$ such that:*

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

Where $view_i^\pi(x_1, \ldots, x_p) = (r_i, x_i, m_1, \ldots, m_t)$, r_i represents the coin tosses made by party i in a normal execution of the protocol for inputs $x_1, \ldots, x_p$. $m_1, \ldots, m_t$ 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 Appendix 1.

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 [27], or a hexagon as in [28].

Theorem 1 (Privacy guarantee). InnerCircle *computes the location proximity testing 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 Appendix 2.

Towards automatic verification The above statements are designed to be amenable to semi-automatic verification. Similar proofs for semi-honest adversaries were constructed in [1]. Although automatic verification is outside the scope of this paper, our definitions and proof strategy is an initial step in this direction.

4 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.

Results from two representative works for generic SMC are provided. We compare to ABY by Demmler et al. [7] to show how InnerCircle performs in relation to an implementation using only garbled circuits [40] (in the following called ABY_{Y}), and to compare against a hybrid system utilizing garbled circuits and arithmetic secret sharing (which we refer to as ABY_{AY}). We also compare with the TASTY tool by Henecka et al. [15] to show a comparison to a system that makes use of both garbled circuits and homomorphic encryption.

For TASTY, the Euclidean distance using homomorphic encryption and the comparison using garbled circuits. With ABY, the comparison is again done using garbled circuits, but we provide benchmarks for distance calculation using both secret sharing and garbled circuits separately.

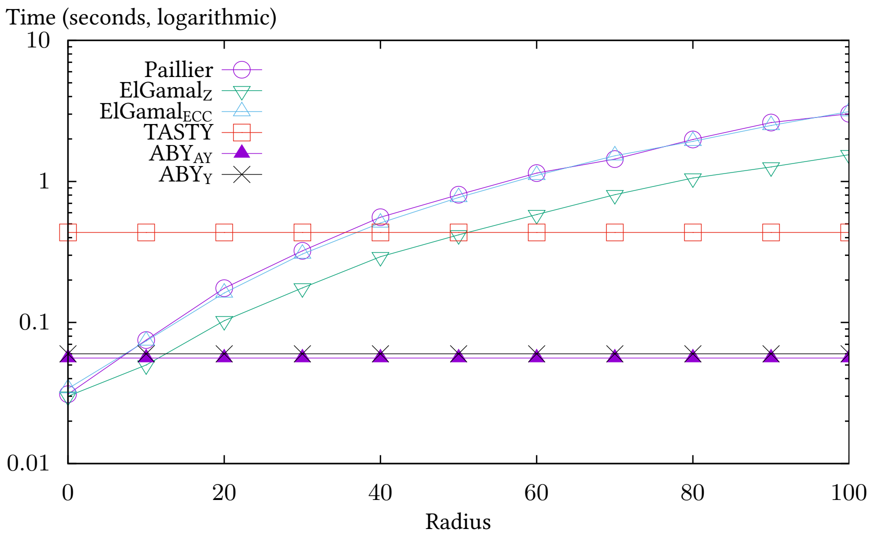


Fig. 4. Time consumption for different algorithms and values of r using 80 bits of security

Three cryptosystems are used for the *InnerCircle* implementation: Paillier [31] and two variants of ElGamal [11]. The first variant of ElGamal is referred to as

ElGamal$_{\mathbb{Z}}$ and uses a group in the integers, the second 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 Appendix 4. 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 Appendix 4.

Figure 4 and Figure 5 show the time taken to complete the protocol using different key sizes. The x-axis shows r, and the y-axis shows time in seconds. Details and further experiments can be found in Appendix 5. For an evaluation of the communication cost, see Appendix 5.2.

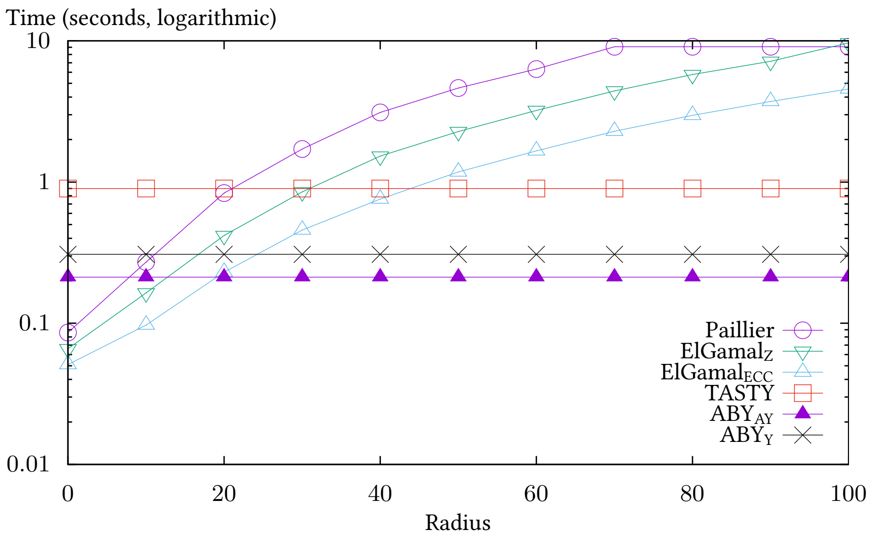


Fig. 5. Time consumption for different algorithms and values of r using 112 bits of security

For $r < 10$ and small key sizes ElGamal$_{\mathbb{Z}}$ is the most efficient implementation as seen in Figure 4. Note that due to the optimization regarding the sum of squares, the results are considerably less than quadratic in r. The benefit of using elliptic curves increases drastically when the key size is increased, as seen in Figure 5. For 112 bits of security, we see that ElGamal$_{ECC}$ is the most efficient implementation up to around $r = 20$. This indicates that *InnerCircle* may be a better choice also for larger thresholds when key sizes are very large. As much of InnerCircle's performance comes from parallel (see further details in Appendix 5), we can argue that InnerCircle may become a more and more viable option in the future, when larger key sizes are required at the same time as more cores are available in CPUs. The figures presented here are for eight threads run in parallel.

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 [12]. If this service needs 112 bits of security and a precision of 10 meter when the area polled is 100 meters wide, it could use *InnerCircle* for better results than when using any competitor. Using the best competitor, ABY with arithmetic sharing and garbled circuits, the time taken to execute the protocol is 212 ms, while using *InnerCircle* with ElGamal$_{ECC}$ gives a response time of 97 ms.

5 Related Work

The two main solutions within SMC are homomorphic encryption and garbled circuits. Section 4 compared *InnerCircle* to ABY by Demmler et al. [7] and TASTY by Henecka et al. [15], which is a prominent work in the field of garbled circuits. The benchmarks show that for this particular functionality the particular requirements of the application dictates whether to make use of garbled circuits or an approach based solely on homomorphic encryption. Similar results have been shown for privacy-preserving face recognition [34]. In general it is hard to determine which approach is suitable for a specific application [20, 18].

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 [36] 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 generalization 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. [42]. The *Louis* protocol computes precise distances using additive homomorphism in the same manner as described in Section 2.1, 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 2.2, 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* [10], 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 [27]. We discuss the most prominent approaches in relation to *InnerCircle*.

Šeděnka and Gasti [35] 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 al. [9], 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 Appendix 3.

The *Hide&Crypt* protocol by Mascetti et al. [10] 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. [27], 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. [39] presents a centralized protocol where principals 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. [38] 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. [28] 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).

Table 1. Comparison of proximity protocols

Protocol	Precise	Decentralized	Fully Privacy-preserving	Single Round-trip
Narayanan 2 [28]				
Narayanan 1,3 [28]		X		X
Pierre[42]		X		X
Louis[42]	X		X	
Lester[42]	X	X		X
Hide&Crypt[10]				
C-Hide&Hash[27]	X		X	
FriendLocator[39]			X	
VicinityLocator[39]	X		X	
PP-[HS,UTM,ECEF][35]	X	X	X	
InnerCircle	X	X	X	X

In Table 1 each protocol is classified as precise, decentralized, fully privacy-preserving and the number of round-trips needed to conclude the protocol. 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 3.

The *Hide&Crypt*, FriendLocator and *Pierre* protocols and the first two protocols by Narayanan et al. all shrink the search space when the precision is increased (when the granule size is decreased), and they are therefore not precise. The *Lester* and *Pierre* protocols, as well as the first and third protocol by Narayanan et al. are decentralized. All other make use of a third party to calculate the proximity. *Hide&Crypt* and FriendLocator are not fully privacy-preserving, as they always leak the granule where the principal is located, either to the other principal or to the third party. The *Pierre* protocol leaks whether the principal is located in the same, a diagonal, or a touching granule, and is therefore not fully privacy-preserving. Further, *Lester* is not fully privacy-preserving as the adversary can learn the exact distance between the principals. None of the protocols by Narayanan et al. are fully privacy-preserving, as the initiator gains information about which granule the responder resides in.

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 1, and we restricted ourselves to the comparison presented previously against generic SMC (which has most features in common with our approach).

6 Conclusions

We have proposed InnerCircle, a parallelizable protocol achieving 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, key sizes are large, and when many threads can be executed in parallel. The former is a common case in the scenario of geofencing [14]. 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. G. Barthe, G. Danezis, B. Grégoire, C. Kunz, and S. Z. Béguelin. Verified computational differential privacy with applications to smart metering. In *2013 IEEE 26th Computer Security Foundations Symposium, New Orleans, LA, USA, June 26-28, 2013*, pages 287–301. IEEE Computer Society, 2013.

2. G. Barthe, B. Grégoire, and S. Z. Béguelin. Formal certification of code-based cryptographic proofs. In Z. Shao and B. C. Pierce, editors, *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, pages 90–101. ACM, 2009.
3. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.
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 S. Ossowski and P. Lecca, editors, *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*, pages 1463–1469. ACM, 2012.
7. D. Demmler, T. Schneider, and M. Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. The Internet Society, 2015.
8. 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.
9. Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In I. Goldberg and M. J. Atallah, editors, *Privacy Enhancing Technologies*, volume 5672 of *Lecture Notes in Computer Science*, pages 235–253. Springer, 2009.
10. D. Freni, C. R. Vicente, S. Mascetti, C. Bettini, and C. S. Jensen. Preserving location and absence privacy in geo-social networks. In J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, and A. An, editors, *CIKM*, pages 309–318. ACM, 2010.
11. T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.
12. Glympse Inc. Glympse. `https://www.glympse.com/`, 2015. [Online; accessed 01-February-2015].
13. O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
14. 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.
15. W. Henecka, S. Kögl, A. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: tool for automating secure two-party computations. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Com-*

munications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010, pages 451–462. ACM, 2010.

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 J. A. Garay, A. Miyaji, and A. Otsuka, editors, *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, volume 5888 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 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. A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001.
23. 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, *The Fifteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc'14, Philadelphia, PA, USA, August 11-14, 2014*, pages 43–52. ACM, 2014.
24. Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008:197, 2008.
25. A. LLC. Location based services startups, March 2014.
26. M. Lochter and J. Merkle. Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. RFC 5639 (Informational), Mar. 2010.
27. 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.
28. A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011.* The Internet Society, 2011.
29. NIST. Recommendation for key management | part 1: General sp 800-57. Web resource: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf.
30. E. N. of Excellence in Cryptology II. Ecrypt ii, September 2012. Web resource: http://www.ecrypt.eu.org/documents/D.SPA.20.pdf.

31. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
32. K. Peng, C. Boyd, E. Dawson, and B. Lee. An efficient and verifiable solution to the millionaire problem. In C. Park and S. Chee, editors, *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, volume 3506 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2004.
33. G. Qin, C. Patsakis, and M. Bouroche. Playing hide and seek with mobile dating applications. In N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. A. E. Kalam, and T. Sans, editors, *ICT Systems Security and Privacy Protection - 29th IFIP TC 11 International Conference, SEC 2014, Marrakech, Morocco, June 2-4, 2014. Proceedings*, volume 428 of *IFIP Advances in Information and Communication Technology*, pages 185–196. Springer, 2014.
34. A. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In D. Lee and S. Hong, editors, *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*, volume 5984 of *Lecture Notes in Computer Science*, pages 229–244. Springer, 2009.
35. J. Sedenka and P. Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In S. Moriai, T. Jaeger, and K. Sakurai, editors, *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 99–110. ACM, 2014.
36. M. Terrovitis. Privacy preservation in the dissemination of location data. *SIGKDD Explorations*, 13(1):6–18, 2011.
37. M. Veytsman. How I was able to track the location of any Tinder user, February 2014. Web resource: http://blog.includesecurity.com/.
38. L. Šikšnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu. Private and flexible proximity detection in mobile social networks. In T. Hara, C. S. Jensen, V. Kumar, S. Madria, and D. Zeinalipour-Yazti, editors, *Mobile Data Management*, pages 75–84. IEEE Computer Society, 2010.
39. L. Šikšnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen. A location privacy aware friend locator. In N. Mamoulis, T. Seidl, T. B. Pedersen, K. Torp, and I. Assent, editors, *SSTD*, volume 5644 of *Lecture Notes in Computer Science*, pages 405–410. Springer, 2009.
40. A. C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.
41. X. Zhang, H. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen. SCION: scalability, control, and isolation on next-generation networks. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 212–227. IEEE Computer Society, 2011.

42. G. Zhong, I. Goldberg, and U. Hengartner. Louis, lester and pierre: Three protocols for location privacy. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, volume 4776 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2007.

Appendix

1 Security Concepts

In the following we recall briefly some fundamental concepts from SMC.

1.1 Negligible functions

Definition 2. *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is said to be negligible if*

$$\forall\, c \in \mathbb{N}.\ \exists\, n_c \in \mathbb{N}.\ \forall_{n \geq n_c}\ |\epsilon(n)| \leq n^{-c}$$

That is, ϵ decreases faster than the inverse of any polynomial.

1.2 Indistinguishability

Definition 3. *The two random variables $X(n, a)$ and $Y(n, a)$ (where n is a security parameter and a represents the inputs to the protocol) are called computationally indistinguishable and denoted $X \overset{c}{\equiv} Y$ if for a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ the following function is negligible:*

$$\delta(n) = |\Pr[\mathcal{A}(X(n, a)) = 1] - \Pr[\mathcal{A}(Y(n, a)) = 1]|$$

1.3 Semantic Security

Furthermore, recall that a public key encryption scheme E is semantically secure or IND-CPA secure if the advantage of any PPT adversary of winning the game IND-CPA in Figure 6 is negligible. This is an important feature that we will use in the following. If a cryptosystem is IND-CPA secure, it is also *multiple message* IND-CPA [3]. *Multiple message* IND-CPA is formalized by the game MM-IND-CPA in Figure 6, where k is polynomially bounded by the security parameter.

2 Proof of Theorem 1

This section shows that *InnerCircle* computes the functionality specified in Section 3 privately in the semi-honest adversary setting. To do so, simulators for *Alice* and *Bob* S_A S_B, as well as the simulator for a third party S_C, are constructed for the views of the three parties.

```
Game IND-CPA :
$(m_0, m_1) \leftarrow \mathcal{A}_1$;
$b \xleftarrow{\$} \{0,1\}$;
$b' \leftarrow \mathcal{A}_2(E_K(m_b))$;
return $b = b'$
```

```
Game MM-IND-CPA :
$((m_0^0, ..., m_k^0), (m_0^1, ..., m_k^1)) \leftarrow \mathcal{A}_1$;
$b \xleftarrow{\$} \{0,1\}$;
$b' \leftarrow \mathcal{A}_2(E_K(m_0^b), ..., E_K(m_k^b))$;
return $b = b'$
```

Fig. 6. IND-CPA and MM-IND-CPA defined in pWhile

```
$\mathbf{View}_A(x_A, y_A, x_B, y_B)$ :
$a_0 \leftarrow E(x_A^2 + y_A^2)$;
$a_1 \leftarrow E(2x_A)$;
$a_2 \leftarrow E(2y_A)$;
$L \leftarrow \mathsf{IC}_B(a_0, a_1, a_2, x_B, y_B)$;
return $a_0, a_1, a_2, L$;
```

```
$\mathbf{S}_A(x_A, y_A, d)$ :
$a_0 \leftarrow E(x_A^2 + y_A^2)$;
$a_1 \leftarrow E(2x_A)$;
$a_2 \leftarrow E(2y_A)$;
$L \leftarrow \mathbf{Sim}\mathsf{IC}_B(d)$;
return $a_0, a_1, a_2, L$;
```

Fig. 7. Simulator and view of *Alice*

View of Alice For the view of *Alice*, it must be shown that there exist a simulator S_A such that:

$$\{S_A(x_A, y_A, d)\} \overset{c}{\equiv} \{view_A^{\pi}(x_A, y_A, x_B, y_B)\}$$

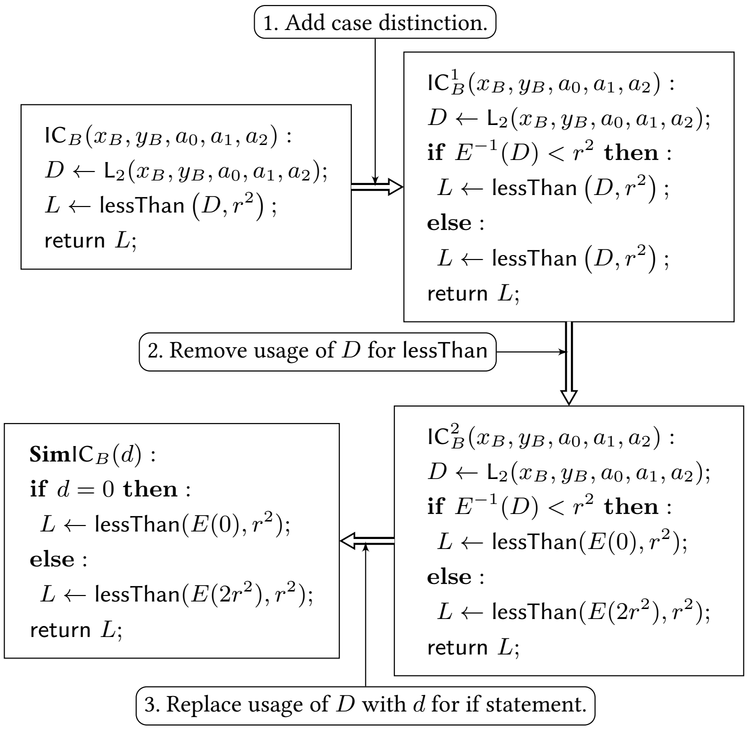
Note that the view of *Alice* consists of the messages sent by her (encryption of her coordinates) plus the vector coming back from *Bob*. This view is formalized in Figure 7. To show is that it has the same output distribution as the one of the simulator S_A, defined also as a probabilistic program.

Our proof strategy is a step-by-step program transformation starting from the view of *Alice*, where we show that each intermediate step preserves the distribution of the original probabilistic program and results in S_A. Since except for line 4 the two programs in Figure 7 are syntactically identical, what remains is to show that there is exists a simulator of IC_B which takes only d as input and has an indistinguishable output distribution. The transformations necessary are shown in Figure 8. Note that the reduction in step 2 follows directly from Lemma 1.

View of Bob For the view of *Bob*, it must be shown that there exist a simulator S_B such that:

$$\{S_B(x_B, y_B)\} \overset{c}{\equiv} \{view_B^{\pi}(x_A, y_A, x_B, y_B)\}$$

Note that *Bob*'s is made entirely of encrypted messages with *Alice*'s private key. *Bob*'s view and a simulator are defined in Figure 9. Since the programs are syntac-


1. Add case distinction.
2. Remove usage of D for lessThan
3. Replace usage of D with d for if statement.

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

```
$\mathsf{IC}^1_B(x_B, y_B, a_0, a_1, a_2)$ :
$D \leftarrow \mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$;
**if** $E^{-1}(D) < r^2$ **then** :
  $L \leftarrow \mathsf{lessThan}\left(D, r^2\right)$ ;
**else** :
  $L \leftarrow \mathsf{lessThan}\left(D, r^2\right)$ ;
return $L$;
```

```
$\mathsf{IC}^2_B(x_B, y_B, a_0, a_1, a_2)$ :
$D \leftarrow \mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$;
**if** $E^{-1}(D) < r^2$ **then** :
  $L \leftarrow \mathsf{lessThan}(E(0), r^2)$;
**else** :
  $L \leftarrow \mathsf{lessThan}(E(2r^2), r^2)$;
return $L$;
```

```
**Sim**$\mathsf{IC}_B(d)$ :
**if** $d = 0$ **then** :
  $L \leftarrow \mathsf{lessThan}(E(0), r^2)$;
**else** :
  $L \leftarrow \mathsf{lessThan}(E(2r^2), r^2)$;
return $L$;
```

Fig. 8. Simulating IC_B

```
**View**$_B(x_A, y_A, x_B, y_B)$ :
$a_0, a_1, a_2 \leftarrow \mathsf{IC}_A(x_A, y_A)$;
$D \leftarrow \mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$;
$L \leftarrow \mathsf{lessThan}\left(D, r^2\right)$ ;
return $a_0, a_1, a_2, L$;
```

```
$\mathbf{S}_B(x_B, y_B)$ :
$a_0 \leftarrow E(0)$;
$a_1 \leftarrow E(0)$;
$a_2 \leftarrow E(0)$;
$D \leftarrow \mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$;
$L \leftarrow \mathsf{lessThan}\left(D, r^2\right)$ ;
return $a_0, a_1, a_2, L$;
```

Fig. 9. Simulator and View of *Bob*

tically equivalent after the assignation of a_0, a_1, a_2 the goal is to show that the distribution of these variables is indistinguishable in both cases. This follows directly from the MM-IND-CPA property of the used cipher, since adversaries without the key are by hypothesis unable to distinguish tuples of encrypted messages.

View of Claire Here, what needs to be shown is that there exist a simulator S_B such that:

$$\{S_C()\} \overset{c}{\equiv} \{view_C^{\pi}(x_A, y_A, x_B, y_B)\}$$

Similarly as for *Bob*, *Claire* only sees encrypted messages between *Alice* and *Bob* under *Alice*'s private key. Her view and simulator are thus defined in Figure 10. As in the case of *Bob*, the fact that the distribution of the used variables is indistinguishable follows directly from MM-IND-CPA, since in both cases the view of C consists merely of encrypted messages under the key of A.

```
$\mathbf{View}_C(x_A, y_A, x_B, y_B)$ :
$a_0, a_1, a_2 \leftarrow \mathsf{IC}_A(x_A, y_A)$;
$D \leftarrow \mathsf{L}_2(x_B, y_B, a_0, a_1, a_2)$;
$L \leftarrow \mathsf{lessThan}\left(D, r^2\right)$ ;
return $a_0, a_1, a_2, L$;
```

```
$\mathbf{S}_C()$ :
$a_0 \leftarrow E(0)$;
$a_1 \leftarrow E(0)$;
$a_2 \leftarrow E(0)$;
$L \leftarrow \mathsf{lessThan}\left(a_0, 0\right)$ ;
return $a_0, a_1, a_2, L$;
```

Fig. 10. Simulator and View of *Claire*

3 Comparison Šeděnka and Gasti

Efforts to benchmark the recent work on privacy-preserving location proximity by Šeděnka and Gasti [35] were unfortunately hampered by missing details in the protocol descriptions, as there is no implementation openly available. A side effect is that the resulting benchmarks achieved by us are about 4 times faster than the results of the benchmarks reported in the original paper, with a similar machine. Therefore, we have instead thoroughly benchmarked the underlying comparison scheme, which was defined by Erkin et al. [9]. Our implementation of Erkin et al.'s comparison scheme shows to be more efficient than the novel lessThan protocol when the queried radius is larger than 40 when using the ElGamal$_{ECC}$protocol instance with 112 bits of security. Šeděnka and Gasti also provide a novel homomorphic distance calculation, which we benchmark to be negligibly slower (by 0.1 ms) than the one used by InnerCircle. The novel distance calculation function is more precise (with respect to the distance on the earth curvature) at larger distances, and could be used in InnerCircle as well.

4 Concrete Protocol Instances

The following shows how three selected cryptosystems can be used in the context of *InnerCircle*, in particular, with respect to the assumptions made in Section 2. A discussion on the computational complexity of the protocol for each cryptoscheme is also provided. Two variants of ElGamal are presented, the first is referred to as ElGamal$_{\mathbb{Z}}$ and uses a cyclic group in the integers, the second one is referred to as ElGamal$_{ECC}$ and uses a cyclic group in elliptic curve cryptography [16]. Using ElGamal$_{\mathbb{Z}}$ and ElGamal$_{ECC}$, the protocol is shown to be secure, while for Paillier it is shown to be secure only under some circumstances.

There are several reasons to consider multiple encryption schemes. As presented in section Section 4. ElGamal$_{\mathbb{Z}}$ yields the most efficient implementation of *InnerCircle* for small key sizes, and ElGamal$_{ECC}$ for larger. Paillier has the important feature of being able to decrypt an arbitrary ciphertext.

4.1 Paillier

The Paillier cryptosystem [31] is public key encryption scheme providing semantic security. For the scope of this paper is only important to know that Paillier is additively homomorphic under ciphertext multiplication and that its plaintext space is $\mathbb{Z}_n$ for $n = p \cdot q$ with p and q prime. It turns out that creating a random function $\mathcal{R}$ is not possible using Paillier, due to the fact that there are subgroups within $\mathbb{Z}_n$, namely the groups of all multiples of p, all multiples of q, and $\mathbb{Z}_n^*$. However, granted that the input to $\mathcal{R}$ is an encryption of a number in $\mathbb{Z}_n^*$ randomness is guaranteed, see Lemma 2 and it's proof.

Lemma 2. *Let $a \in \mathbb{Z}_n^*$ and $\rho \xleftarrow{\$} [1..n-1]$ a uniformly random number of $\mathbb{Z}_n \setminus \{0\}$. Then $a \cdot \rho$ is a uniformly distributed random number in $\mathbb{Z}_n \setminus \{0\}$.*

Proof. The claim is $\forall_{i,j \in \mathbb{Z}_n \setminus \{0\}} \Pr[a \cdot \rho = i] = \Pr[a \cdot \rho = j]$. By hypothesis, $a \in \mathbb{Z}_n^*$, so there exists an inverse a^{-1} of a, so the claim is equivalent to $\forall_{i,j \in \mathbb{Z}_n \setminus \{0\}} \Pr[\rho = a^{-1} \cdot i] = \Pr[\rho = a^{-1} \cdot j]$. Both $a^{-1} \cdot i$ and $a^{-1} \cdot j$ are members of $\mathbb{Z}_n \setminus \{0\}$, and since by hypothesis ρ is uniformly distributed in $\mathbb{Z}_n \setminus \{0\}$, the claim follows.

A randomization function is defined as $\mathcal{R}(e) = e^k$, with k random. This function yields truly random values for non-zero e if e is guaranteed to be in $\mathbb{Z}_n^*$, as per 2. This sampling can only be done reliably (without knowledge of p or q, in which case the security is void) by using number which is magnitudes smaller than both p and q. For a maximal coordinate $\mathcal{C}$, the implementation must assert that $\mathcal{C} << p < q$. Using a precision of one meter and considering a square with the side equal to the earth's circumference $4 \cdot 10^7$ meters, $\mathcal{C} = 2 \cdot (4 \cdot 10^7)^2$. Thus, n must be at least $2 \cdot log_2(\mathcal{C}) \approx 104$ bits for Paillier to be used on earth with 1 meters resolution. Using even a 256 bit key the earth can thus be precisely measured in nanometers, which means that for practical key sizes this is never an issue when all parties are honest. However, an erroneous software or malicious adversary may falsify this assumption. This enforcement is left for future work, but for the scope of this paper it may be assumed that both agents are semi-honest, and therefore will use real coordinates.

4.2 ElGamal

Choosing a plaintext space as a group G with prime order, it is possible to describe a general randomization function $\mathcal{R}$ to achieve Equation (4) for ElGamal. Every element in the group is a generator of the group. Therefore, one can always randomize a non-zero element e by computing e^k for some k chosen uniformly at random in G.

ElGamal$_{\mathbb{Z}}$ Using an identity-mapping in the integers, $map(x) = e$, ElGamal is *multiplicatively* homomorphic. However, it can be turned into an additive scheme ElGamal$_{\mathbb{Z}}$ (for a group $G = Z_q$ where q is a prime number) by using an additively homomorphic mapping function $map(m) = g^m$, and a inverse mapping $rmap(e) = log_g(e)$ [32]. This makes it infeasible to compute the decryption of arbitrary plaintexts, but for the purposes of this paper it suffices that encryptions of zero can be decrypted.

ElGamal$_{ECC}$ Using ElGamal it is possible to leverage elliptic curve cryptography, which under many circumstances is more efficient than working in a group of integers [22]. The group G is then a prime field of elliptic curve, with a generator point $\widehat{g}$. Mappings from integers to coordinates on an elliptic are a well-studied topic. For the purposes of this paper, the simple mapping $map(m) = \widehat{g} \cdot m$ again suffices for our purposes. This is a one-to-one totally ordered mapping. After decryption, it is enough to check that the resulting element is the same as $map(0)$.

As per the general homomorphism ElGamal, the scheme is inherently additive and thus have Equation (1), Equation (3) and Equation (4) by default. Negation in elliptic curves is trivial, as negation of a group element is computed by negating the y-component of the coordinate. Thus, Equation (2) is achieved as:

$$\neg E(m) = -E(m) = E(-m)$$

5 Performance

All benchmarks are run without any latency from network or heavy disk activity using a single machine with the entire state stored in memory. The machine used for the experiments used an Intel i7 CPU operating at 3.6GHz. While many location-enabled devices are cell-phones and often are considered weak, modern phones have fairly similar performance. For instance a Nexus 6 operates at 2.7GHz. The benchmarks consider r from 0 to 100. Key sizes, n, are mainly considered for the case where the user wants either 80 or 112 bits of security.

For these experiments the NIST recommendations of using an RSA modulus of 1024 and 2048 bits for 80 and 112 bits respectively are followed [29] (though ECRYPT recommends larger keys [30]). For elliptic curves, both aforementioned sources agree that 160 and 224 bit keys yields 80 and 112 bits of security respectively. The brainpool standard curves [26] were used.

Optimizations The implementation makes use of the fact that distances in the protocol are calculated as $D = (x_A - x_B)^2 + (y_A - y_B)^2$. They are always a sum of two squares. Thus, there are many numbers that do not need to be considered when computing lessThan. Approximately 44% for $r = 10$, 28% for $r = 100$ and 22% at $r = 500$ are constructed as sums of squares.

Each party creates a cache of all negated values for each respective public key, before starting the protocol to increase performance. Since the resulting ciphertext is randomized by subsequent operations, reusing it does not affect IND-CPA. The time needed to construct this cache for $r = 100$ is 0.12, 17.9 and 23.4 seconds for a Paillier and ElGamal$_{\mathbb{Z}}$ using a 2048 bit key and ElGamal$_{ECC}$ using a 224 bit key, respectively.

InnerCircle is extremely amenable to parallelization. The loops for constructing and interpreting the shuffled list can be completely unrolled as each iteration is independent. The presented benchmarks were created using 8 parallel worker threads, over four cores.

Comparing to state-of-the-art SMC All benchmarks presented consider the online time of the protocol spent by each end-point. We define online as from the time of the first non-reusable bit is being computed. A reusable bit should be useable also in communications with other parties. This means that principals are allowed to a priori communicate keys and other constants, but nothing that could not be reused in subsequent runs of the protocol towards an arbitrary party. Note that with this definition of online, base-OTs are included in the online phase. In systems where pairwise communication is needed, and public keys (and also the cache used by InnerCircle) may be distributed via a public bulletin board, this distinction becomes important.

The benchmarks of InnerCircle generate 50% of all requests using coordinates within a radius of r, and 50% which yield negative proximity results. This is important, as $Alice$ will stop decrypting the list when she finds a positive proximity results. Note that this timing difference is in no way noticeable by Bob within one run of the protocol.

5.1 Performance

This section presents further results for the running time of the protocol from the conducted experiments, and shows how the different protocol instances perform in relation to each other.

Additional benchmarks were performed to investigate corner cases. Using a key size of 512 bits (256 bits of security) for ElGamal$_{ECC}$ completes a run for $r = 100$ in about 4 seconds. An additional experiment for $r = 1000$ was conducted for ElGamal$_{\mathbb{Z}}$ using 1024 bit keys, for which the protocol completed in 84.6 seconds.

Table 2. Results using different number of threads

Threads used	2	6	8	10
Time taken	26.29	12.80	7.57	6.11
Speedup	0.00%	51.29%	71.20%	76.76%
Max theoretical	0.00%	66.67%	75.00%	80.00%

To investigate how well *InnerCircle* scales with added cores, a larger machine was rented at Amazon Web Services (C3 tier, 10-core 2.8GHz CPU). The results are shown in Table 2. The benchmarks were run using $r = 100$, for a number of threads from 2 to 10. Speedup is relative to two threads, the fewest measured. These numbers show that *InnerCircle* scales extremely well with the hardware evolution of today, where more cores are added to a processor, rather than it operating at a higher frequency. On the other hand, neither TASTY nor ABY are multithreaded, and it is not obvious how to parallelize the underlying primitives.

Although the generic implementations perform better than *InnerCircle* for large values of r, many applications would benefit from small proximity ranges. Note that how *far* principals can be from each other depends on the *unit* of r used by the application. In other words, the benchmarks show a limit of the *granularity* of the application, not an upper bound for the *distance* where principals are considered close.

5.2 Message Size

The first message of *InnerCircle* is always three ciphertext, the second message contains $O(r^2)$ ciphertexts. Due to the optimization (see Section 5) there is in practice no need to consider the entire range. Table 3 shows the communication cost incurred by *InnerCircle*, where $|r^2|$ is the number of sums of squares to be considered for a specific r.

Table 3. *InnerCircle* communication cost

r	$\lvert r^2 \rvert$	*ElGamal*$_{\mathbb{Z}}$ *& Paillier*		*ElGamal*$_{ECC}$	
		1024	2048	160	224
0	0	768 B	1.5 KB	240 B	336 B
20	146	37.3 KB	74.5 KB	11.6 KB	16.3 KB
40	505	127 KB	254 KB	39.7 KB	55.6 KB
60	1064	266.8 KB	533.5 KB	83.4 KB	116.7 KB
80	1812	453.8 KB	907.5 KB	141.8 KB	198.5 KB
100	2750	688.3 KB	1.3 MB	215.1 KB	301.1 KB

The empirical analysis shows that using ElGamal$_{ECC}$ is better regardless of the security parameter w.r.t communication cost. Nevertheless, the communication cost during the heavier benchmarks are comparable to downloading a medium-resolution image (around 1.3MB), making it still useful in practice.

When considering TASTY and ABY, they have a static bandwidth usage for any fixed level of security, just as for the performance benchmarks. ABY$_{\mathrm{Y}}$ requires 261 KB and 393 KB of traffic within a protocol run for 80 and 112 bits of security, respectively. ABY$_{\mathrm{AY}}$ requires only 32 KB and 72 KB of communication for 80 and 112 bits of security. TASTY shows that it requires a mere 24KB for 80 bit keys and 28KB for 112 bit keys.

6 Asymptotic Analysis

Performance In the following analysis, the asymptotic performance of the different primitives are expressed as functions of the key size, n. The asymptotic running time of the encryption function is denoted as $\mathcal{E}^n$, and for the decryption function as $\mathcal{D}^n$. The asymptotic running time of $\oplus$, $\odot$, $\neg$ and $\mathcal{R}$ for is herein expressed as $\overline{\underline{\oplus^n}}$, $\overline{\underline{\odot^n}}$, $\overline{\underline{\neg^n}}$ and $\overline{\underline{\mathcal{R}^n}}$ respectively.

IC$_A$ To send the first message in the protocol, recall that the initiator computes $E(x_a^2 + y_a^2)$, $E(2x_a)$ and $E(2y_a)$. The upper bound is thus $\mathcal{O}\left(3 \cdot \mathcal{E}^n\right) = \mathcal{O}\left(\mathcal{E}^n\right)$.

IC$_B$ In order to compute the response message, L_2 is used to compute the distance, after which lessThan is called. The process is dominated by lessThan, as it is not constant with respect to r. It is assumed that list operations (creating, appending, shuffling) are negligible in comparison to the homomorphic operations. lessThan computes $\mathcal{R}(E(D) \oplus \neg E(i))$ for $i \in \{0..r^2\}$, which means to encrypt i, subtract the result from the encryption of D, and randomize the encrypted difference (the encryption of D is previously computed). The upper bound of IC$_B$ is in $\mathcal{O}\left(r^2 \cdot \left(\mathcal{E}^n + \overline{\underline{\oplus^n}} + \overline{\underline{\neg^n}} + \overline{\underline{\mathcal{R}^n}}\right)\right)$.

inProx Analyzing the response means decrypting r^2 times, and then comparing the resulting plaintext to zero. The bound is easily found to be $\mathcal{O}\left(r^2 \cdot \mathcal{D}^n\right)$
InnerCircle Given the bound for each phase of the protocol, what is the upper bound of an entire protocol run? This is easily computed from the analysis above:

$$\begin{aligned}&\mathcal{O}\left(\mathcal{E}^n + r^2 \cdot \left(\mathcal{E}^n + \overline{\underline{\oplus^n}} + \overline{\underline{\neg^n}} + \overline{\underline{\mathcal{R}^n}}\right) + r^2 \cdot \mathcal{D}^n\right)\\ &\quad = \mathcal{O}\left(r^2 \cdot \left(\mathcal{E}^n + \overline{\underline{\oplus^n}} + \overline{\underline{\neg^n}} + \overline{\underline{\mathcal{R}^n}} + \mathcal{D}^n\right)\right)\end{aligned}$$

Comparison and conclusion From the given bounds for abstract additively homomorphic schemes so far given, concrete bounds for the instances presented in Appendix 4 is found in Table 4. The asymptotic analysis for the concrete instances of $\oplus$, $\odot$, $\neg$ and $\mathcal{R}$ are therein presented, followed by the analysis of IC$_A$, IC$_B$, inProx and globally for *InnerCircle*.

In the analysis, $\eta(b, e)$ denotes the complexity of raising a base of size b to a power of size e, $\mu(n)$ denotes multiplication of two numbers of size n, and $\psi(n)$ the time taken to find the multiplicative inverse modulo a number of size n. It is assumed that for all cases, addition is dominated by multiplication, and that $\mathcal{O}\left(\mu(n)\right) \subset \mathcal{O}\left(\eta(n, n)\right) \subset \mathcal{O}\left(\psi(n)\right)$.

Table 4. Asymptotic cost of the different homomorphic operations

	Paillier	ElGamal$_{\mathbb{Z}}$	ElGamal$_{ECC}$
$\mathcal{E}^n$	$\mathcal{O}\left(\eta(n,n)\right)$	$\mathcal{O}\left(\eta(n,n)\right)$	$\mathcal{O}\left(\log(n)\cdot\psi(n)\right)$
$\mathcal{D}^n$	$\mathcal{O}\left(\eta(n^2,n)\right)$	$\mathcal{O}\left(\psi(n)\right)$	$\mathcal{O}\left(\log(n)\cdot\psi(n)\right)$
$\underline{\overline{\oplus^n}}$	$\mathcal{O}\left(\mu(n^2)\right)$	$\mathcal{O}\left(\mu(n)\right)$	$\mathcal{O}\left(\psi(n)\right)$
$\underline{\overline{\odot^n}}, \underline{\overline{\mathcal{R}^n}}$	$\mathcal{O}\left(\eta(n^2,n)\right)$	$\mathcal{O}\left(\eta(n,n)\right)$	$\mathcal{O}\left(\log(n)\cdot\psi(n)\right)$
$\underline{\overline{\neg^n}}$	$\mathcal{O}\left(\psi(n^2)\right)$	$\mathcal{O}\left(\psi(n)\right)$	$\mathcal{O}\left(1\right)$
$pplp_A$	$\mathcal{O}\left(\eta(n,n)\right)$	$\mathcal{O}\left(\eta(n,n)\right)$	$\mathcal{O}\left(\log(n)\cdot\psi(n)\right)$
$pplp_B$	$\mathcal{O}\left(r^2\cdot\psi(n^2)\right)$	$\mathcal{O}\left(r^2\cdot\psi(n)\right)$	$\mathcal{O}\left(r^2\cdot\log(n)\cdot\psi(n)\right)$
inProx	$\mathcal{O}\left(r^2\cdot\eta(n^2,n)\right)$	$\mathcal{O}\left(r^2\cdot\psi(n)\right)$	$\mathcal{O}\left(r^2\cdot\log(n)\cdot\psi(n)\right)$
Total	$\mathcal{O}\left(r^2\cdot\psi(n^2)\right)$	$\mathcal{O}\left(r^2\cdot\psi(n)\right)$	$\mathcal{O}\left(r^2\cdot\log(n)\cdot\psi(n)\right)$

The Paillier primitives quickly become more time-consuming than the ElGamal-based instances as the ciphertext size is n^2 for Paillier, but linear in n for ElGamal. The runtime of the elliptic curve instantiation is highly dependent on point-addition, which is dominated by the time taken to compute the slope of a line in modulo space. Asymptotically in the key size, elliptic curves are slower than integer implementations of ElGamal, though they are asymptotically faster in the bits of security, as is evident in the practical experiments in Section 4.

6.1 Asymptotic behavior of InnerCircle

A summary of the time complexity of the protocol with respect to the proximity range r and the security parameter n for the instances presented above is found in Table 5. Here $\psi(n)$ the time taken to find the multiplicative inverse modulo a number of size n. To interpret the bounds, let $\eta(b,e)$ denote the complexity of raising a base of size b to a power of size e, $\mu(n)$ denotes multiplication of two numbers of size n, and note that for all cases, addition is dominated by multiplication, and that $\mathcal{O}\left(\mu(n)\right)\subset\mathcal{O}\left(\eta(n,n)\right)\subset\mathcal{O}\left(\psi(n)\right)$.

Table 5. Asymptotic cost of concrete implementations

Paillier	ElGamal$_{\mathbb{Z}}$	ElGamal$_{ECC}$
$\mathcal{O}\left(r^2\cdot\psi(n^2)\right)$	$\mathcal{O}\left(r^2\cdot\psi(n)\right)$	$\mathcal{O}\left(r^2\cdot\log(n)\cdot\psi(n)\right)$

The Paillier primitives quickly become more time consuming than the ElGamal-based instances as the ciphertext size is n^2 for Paillier, but linear in n for ElGamal.

The runtime of the elliptic curve instantiation is highly dependent on point-addition, which is dominated by the time taken to compute the slope of a line in modulo space. Asymptotically in the key size, elliptic curves are slower than integer implementations of ElGamal, though they are asymptotically faster in the bits of security, as is evident in the practical experiments in Section 4.

6.2 Communication cost of InnerCircle

This section details the network traffic generated by *InnerCircle*, through analyzing the upper bound of the number of bits transmitted in each message of the protocol. First note that the asymptotic number of bits of a ciphertext is the same for all crypto schemes considered; within Paillier a ciphertext is $\mathcal{O}\left(\log\left(n^2\right)\right) = \mathcal{O}\left(\log\left(n\right)\right)$, for ElGamal$_{\mathbb{Z}}$ $\mathcal{O}\left(2 \cdot \log\left(n\right)\right) = \mathcal{O}\left(\log\left(n\right)\right)$, and for ElGamal$_{ECC}$ it is

$$\mathcal{O}\left(4 \cdot \log\left(n\right)\right) = \mathcal{O}\left(\log\left(n\right)\right).$$

IC_A returns three ciphertexts to be sent in the initial message, and the size of a request is thus $\mathcal{O}\left(3 \cdot \log\left(n\right)\right) = \mathcal{O}\left(\log\left(n\right)\right)$. During IC_B, r^2 ciphertexts are sent, and the size of is thus $\mathcal{O}\left(r^2 \cdot \log\left(n\right)\right)$. The total bandwidth usage of *InnerCircle* is thus:

$$\mathcal{O}\left(\log\left(n\right) + r^2 \cdot \log\left(n\right)\right) = \mathcal{O}\left(r^2 \cdot \log\left(n\right)\right)$$