Thesis for the Degree of Doctor of Philosophy

Robust location privacy

Per Hallgren

CHALMERS | GÖTEBORG UNIVERSITY

Department of Computer Science and Engineering
Chalmers University of Technology and Göteborg University
Göteborg, Sweden 2017

Robust location privacy
Per Hallgren
ISBN 978-91-7597-605-1

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 4286
ISSN 0346-718X

Technical Report 144D
Department of Computer Science and Engineering
Research group: Information Security

Department of Computer Science and Engineering
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31-772 1000

Printed at Chalmers
Göteborg, Sweden 2017

Abstract

The Internet is in many ways both fascinating and yet also scary. For most people, a single commercial entity owns the power to disclose all their personal emails. Most commonly your emails are only disclosed to you and your correspondent, but the power to choose who sees these emails is in fact not yours. The power to control the release of data about ones person is what most people refer to as *privacy*.

In spite of this, almost nothing that the Internet is used for gives the originator of a message control over it. When you use a social media platform, you are given the intuition that you choose which friends who can see any posts and photos that you publish, and of course the connection is encrypted to thwart eavesdropping. However, the service provider may still share this data to anyone they like. From a technological standpoint, a user almost never has the power of their data; in other words, there's normally no privacy on the Internet.

This thesis is describes different ways of giving end-users more control over some parts of their own personal data using cryptography for the specific case of location data, enhancing their privacy. The majority of the thesis explores how to make use of location proximity, to check whether to users are close to each other, as a basic primitive while maintaining privacy through additively homomorphic encryption. The thesis further explores the application scenario of ridesharing, or car pooling, using both additively homomorphic encryption and private set intersection. All of the solutions presented sport proven privacy-preserving capabilities, and have been implemented to measure their performance. This thesis shows in what contexts there's still a ways to go, but also highlights some settings in which it might already be time to move theory into practice.

Acknowledgements

My greatest thanks goes to my two supervisors and good friends, Andrei and Martìn. I am positive that I would not have chosen research as my path in life if not for the two of you.

To Andrei, for continuously turning my attention back towards research. I started working part-time with Andrei a few years back, and I (we? ☺) always had a good time. Andrei's helped me to not only cope with, but enjoy, research on the side of an often busy life situation. Andrei is without doubt the biggest reason why I decided to come back to Chalmers. Not because other employers aren't good, but because having Andrei as your supervisor is downright awesome.

To Martìn. Right from the start it was very fun working with you, and I'm repeatedly amazed that I can learn so much from your approach to the problems we solve. You are a phenomenal support during work hours, and a rare good friend after. I can't help but feel lucky to have stumbled into collaboration with you.

Thanks goes also to the entire department at Chalmers, entirely too many to list here.

Further thanks goes to my academic foster family at the TUM; to Alexander, Dominik, Enrico, Florian, Kobold Superstar, Martìn (again), Matthias, Prachi, Sebastian and Tobias. Thanks for the no-limits-or-boundaries discussions and for the great Thursdays!

My parents deserve so many more thanks than I can give, for always supporting and encouraging me, and not even knowing that they're doing it. And thanks to Henrik for thorough proofreading!

Finally, to my dearest Kristin, the number one fan of my academic career. You are an endless source of support and encouragement, there's nobody who believes more in me than you do. For all the laughs, all the love and all the popcorn you give to me, I cannot thank you enough.

Contents

Introduction 1

Paper **One**
InnerCircle: A Decentralized Privacy-Preserving Location Proximity Protocol 19

Paper **Two**
BetterTimes: Privacy-assured Outsourced Multiplications for Additively Homomorphic Encryption on Finite Fields 53

Paper **Three**
MaxPace: Speed-Constrained Location Queries 77

Paper **Four**
Location-enhanced Authentication using the IoT: Because You Cannot Be in Two Places at Once 113

Paper **Five**
Privacy-Preserving Location-Proximity for Mobile Apps 149

Paper **Six**
PrivatePool: Privacy-Preserving Ridesharing 171

Introduction

The Internet is vast, and in many ways it is an amazing piece of technology. It connects people from all over the world. It lets us do banking, watch movies, listen to music, read newspapers, receive education from home, search for information, and enjoy social interaction with all of our friends. The Internet plays a very important role in our society today. Most of what we are, what defines us as individuals, has at some point passed over the Internet. How many steps we walked today, our weight, travel schedule, our grades and achievements, our emails, our taste in music, our relationships, all of our digital conversations, our attitude towards friends, our mood day-by-day, and the entirety of our economic data. All the data needed to tell who and what we are is on the Internet.

Our information has never before been so exposed. We rely on the Internet in every-day life, many tasks carried out today would not be possible without it. And while we need to transmit sensitive data over the Internet, we also need to to maintain our personal integrity – we can not let the internet degrade us below what is decent with respect to our privacy. Already in 1890 Warren and Brandeis thought that technology was spreading information too fast for privacy to be maintained. In their work "The right to privacy" they respond to the technological advancement of the camera and the increasing proliferation of newspapers making use of photography. In this work, they also gave us the first definition of privacy, as *"right to be let alone"* [24].

> *Instantaneous photographs and newspaper enterprise have invaded the sacred precincts of private and domestic life; and numerous mechanical devices threaten to make good the prediction that "what is whispered in the closet shall be proclaimed from the house-tops."*
>
> – Warren and Brandeis

Though devices are no longer mechanical but instead digital, we still struggle with privacy in much the same way. Today we have much more data than images that we need to protect, and information spreads in a completely incomparable velocity as compared to 1890. What is the impact of cameras and newspapers next to that of smartphones and social media? Given that we have the same concerns, but the technology impact is tremendously much greater I can not help but to raise the question of whether Warren and Brandeis exaggerated the impact then, or if we do not fully appreciate the threat of the modern age.

I think we rely more heavily on personal integrity than we realize in our day-to-day life. The United Nations lists privacy as article 12 of the Universal Declaration of Human Rights by the United Nations [22]. In fact, it is easy to get the feeling that the free world is not likely to continue to function in the event of serious degradation of privacy. Our behavior is known to be more constrained if we know that we are observed [25, 8], making us less likely to speak out of turn, to deviate from the norm, to innovate, and to call out miss-behavior. I would even go so far as to say that a democratic society, as far as we see them today, would be very hard-pressed to exist in a world without privacy. If ballots are not private, can minorities really vote according to their hearts' desire?

I personally do not think there is any doubt that we need to work on maintaining privacy – not only in the short term, but for a sustainable society in the long run – but we also can not cease exposing our information to the Internet. It is therefore important that we are able to use current services on the Internet privately. We should be able to exchange messages without fear of having our privacy violated, but at the same time we need to enjoy not only the services we see today, but also the ever more complex applications of the future.

This thesis focuses on exploring the privacy of a specific piece of information – our *location*. In this work, we use a technique called *secure multi-party computation* (SMC) [26]. Using SMC, we give guarantees that when location information is sent to services on the Internet, it remains impossible for the service provider to read the data without the consent of the end user.

1 Privacy and Confidentiality

Most of this thesis talks about privacy. Privacy is a subjective matter, and what is considered private information is often different in different cultures. Computer scientists often talk about confidentiality instead, which has a more objective and precise meaning. Whether or not a system provides privacy for the user's data is not

something we can prove mathematically – but whether the data is confidential is an objective fact. As an example, tax statements in Sweden are public documents. Being a swede, my privacy is therefore not violated by the fact that my income is not confidential, as this is inherently accepted in the Swedish society. We can construct a system which keeps tax statements confidential, but to construct a system which respects the users' privacy we may have to take into consideration whether they were brought up in Sweden or some other country. In this thesis, as is common in the literature within computer security, the words privacy and confidentiality are sometimes used interchangeably – when talking about the privacy of a users, we mean the confidentiality of their data.

In public media we sometimes see arguments raised against privacy, as it for instance hampers the effectiveness of law enforcement. Law enforcement may need to have surveillance on known criminals to maintain safe living conditions. However, for the scope of this thesis more privacy is always considered better than less privacy.

A large portion of the techniques proposed in academic literature to preserve privacy make use of pragmatic approaches such as simple transformations of the data to hide the most important characteristics, commonly referred to as *obfuscation*. In my opinion, these largely fall short to provide rigorous privacy guarantees. In contrast, this thesis focuses on approaches using SMC in order to protect a user's information, ensuring that the service is *unable* to intrude on privacy, rather than *unlikely* as may be the effect of obfuscation techniques.

Location Privacy Location privacy was defined as *"the ability to prevent other parties from learning one's current or past location"* by Beresford and Stajano in 2003 [2]. Later, a more precise definition was presented by Duckham and Kulik in 2006, who called it *"a special type of information privacy which concerns the claim of individuals to determine for themselves when, how, and to what extent location information about them is communicated to others"* [9]. This definition by Duckham and Kulik captures subtle nuances of location information that can be disclosed: *when*, *how* and *what*. It is reasonable that a user is more concerned about disclosure of their recent location updates than older information. How information is released is also of importance, for instance a service may allow the owner of location data to decline certain location requests rather than always automatically dispatching information about their location. Finally, what information is disclosed, and in what detail, is naturally of interest – a user may be fine with disclosing which city they are currently in, but not willing to disclose whether they are in the hospital.

Some researchers have argued that many users are willing to give away their location information [1, 6]. In my opinion this fact does not in any way lessen the importance of location privacy, it simply shows that users are keen to use the service. Since it has hitherto not been possible to easily deploy a privacy-preserving Location-Based Service (LBS), I conjecture that such approaches will be increasingly popular as they apply to more and more services.

I would argue that if privacy is inherently guaranteed by the technology, many audiences would likely be willing to provide more of their private data. Thus, these techniques open new venues for operating on private data, and enables *more private data to be collected*. They would need to trust only the technology, instead of the service provider. Using SMC, and thus allowing more private data to be used while making sure that less data is known, could give very powerful platforms in the future.

2 A Gentle Introduction to Secure Multi-party Computation

Existing cryptographic techniques have for a long time made information *unreadable* except towards intended parties [23, 27]. SMC is a separate strain of research which recently picked up much momentum and makes information *usable* while still being unreadable [18, 17, 11]. Traditional cryptography handles static information. It allows us to send and receive information privately by encrypting communication when surfing the web, reading emails and using instant messaging. It also allows us to store data privately by encrypting our hard drives.

With traditional cryptography, the data that can be decrypted is exactly the data that was encrypted. With SMC, we can encrypt some piece of data, and perform computations on the ciphertext, to decrypt something else. As an example, let us imagine three users Alice, Bob and Claire. Assume that Alice and Bob can encrypt data, but only Claire can decrypt. Alice and Bob each has a secret number a and b, respectively. Alice and Bob wants to let Claire now the sum of their numbers. To achieve this, Alice encrypts her number, say $a = 2$, sends it to Bob, who can add his number to the ciphertext, say $b = 3$. Now Claire can decrypt the number 5, while Alice and Bob retains the privacy of their secret inputs.

The above example is not a very useful application compared to most services we see on the internet, but as we will see later in the thesis, SMC can be used for real-world applications as well. This means that maintaining privacy no longer implies the necessity to remove functionality – Alice and Bob can keep a and b secret while still allowing Claire to learn the sum. We can focus on achieving the functionality

of an application, rather than on telling the service provider all information needed in the necessary computation. Companies behind social networks wants to *target adds* towards you, they do not necessarily need to *know* your private data. A good example of how these cryptographic techniques can be applied is shown by Bogetoft et al. [5], where they present results from an experiment for sugar beet farmers in Denmark. The farmers sold their crops in a privacy-preserving manner with the help a cryptographic auctioning system. The sales were processed without disclosing any bids, except the final one, to any party.

The three distinct areas that currently dominate the SMC scene are *homomorphic encryption* (HE) [18, 11], *garbled circuits* (GC) [26] and *secret sharing* (SS) [21]. Homomorphic encryption schemes commonly support only computation of either multiplication or addition [17], and are then called partially homomorphic. However, *fully homomorphic encryption* (FHE) schemes also exist [11], which can compute both additions and multiplications given only ciphertexts. Unfortunately, FHE is much less efficient than HE schemes which are only additively or multiplicatively homomorphic.

Homomorphic encryption can compute arithmetic integer operations in constant time with respect to the size of the operands. Garbled circuits is often faster for more complex functions where variable size is small and fixed. There is active and accelerating research in both fields, and which solution performs best is typically application-dependent. Secret sharing normally requires an honest majority (for instance three parties with at most one is misbehaving), but is for many applications the most efficient approach to SMC. Secret sharing has been shown to be suitable for several real-life scenarios as exemplified through usages of the ShareMind Project [4, 13] where this level of trust is acceptable and efficiency is paramount.

SMC can be integrated in a system in different ways depending on which of FHE, HE, SS and GC is used. FHE is a great solution for cloud computing scenarios, as the party holding the private key does not need to be active during computations. For partially HE and GC, it is usually not the case that the party receiving the output has to do less work than if the protocol is run in the plaintexts, though it has been demonstrated that GC can be used to speed up computations for some applications [7]. For partially HE and GC, it is common that the function is computed by a party that also is providing inputs. SS is mostly suitable when the involved stakeholders in a system are fixed, and where they inherently are reluctant to collude. This could be the setting of a set of governmental institutions, for instance. When applying SMC for location privacy HE is a good choice, as geometric computations

are carried out using many arithmetic operations, which is why this is the choice made for most works included this thesis.

Homomorphic Encryption Several papers in this thesis focuses on how to use HE. HE schemes are a subset of public-key encryption schemes. Public-key cryptographic systems are asymmetric, where only the holder of the private key can decrypt and anyone who holds the public key can encrypt. The private key must remain secret, while the public key can be published and considered globally known. Some other cryptographic systems are symmetric, which means that the same key is used both for encryption and decryption, but these are not important for this thesis.

HE allow for a user who does not hold the private key (and thus cannot decrypt the data) to compute functions on the ciphertexts, which have *predictable meaning* in the plaintexts. The most canonical example is school-book RSA. Given a private key k and a public key K, encryption works by exponentiation, a message m is encrypted by computing m^K, resulting in a ciphertext c. Given two ciphertexts c_1 (encrypting m_1) and c_2 (encrypting m_2), any party can compute the ciphertext $c_3 = (m_1 * m_2)^K$ by simply multiplying c_1 and c_2.

The second and third paper of this thesis uses additively homomorphic encryption. Given two ciphertexts encrypted using an additively homomorphic cryptographic system, such that c_1 is the encryption of m_1, and c_2 is the encryption of m_2, one can compute another ciphertext c_3 encrypting $m_1 + m_2$. Using additively homomorphic encryption, it is also possible to compute the multiplication if one plaintext is known to the evaluator. To compute $c_1 \cdot c_2$ while knowing that c_2 encrypts m_2, one adds c_1 to itself m_2 times, computing $\sum_0^{m_2} c_1$.

3 Contributions

This section outlines the contributions presented later in the thesis. The thesis contains six separate papers, which all follow Alice and Bob as they try to communicate different functions of location data with different privacy guarantees.

The first paper proposes a concise privacy-preserving protocol for proximity testing, called *InnerCircle*. InnerCircle is a building block used in several of the following papers. InnerCircle only gives privacy guarantees if the attacker is honest, in the sense that they follow the protocol. This attacker model, called *semi-honest*, is a normal setting when the adversary cannot easily change the source code of the running program. The second paper provides a new primitive, called *BetterTimes*, which can be used in InnerCircle and many other protocols to allow them to tackle

stronger attackers, called *malicious*, who are also able to deviate from the intended protocol flow.

Another drawback of InnerCircle is that it only considers what privacy guarantees are achieved during a single invocation of the protocol. But when included in an application, the protocol is likely to run many times. This is addressed with the *MaxPace* policy, which tackles malicious adversaries while giving privacy guarantees even as the protocol is rerun.

The fourth paper utilizes the Internet of Things (IoT) to enable stronger authentication. This is done by aggregating the location data of all devices believed to be carried by the user. While such a system may be run by any trusted third party, we also provide a privacy-preserving version utilizing similar techniques as those used in InnerCircle.

The fifth paper studies how InnerCircle as implemented to be used in an Android app compares to existing popular Android applications found on Google Play.

The last paper of the thesis considers a larger functionality than proximity testing, that of ridesharing. The paper studies what patterns users which may enjoy ridesharing may follow, and shows two separate tracks to detect ridesharing opportunities, one via an extension of InnerCircle, and the other via novel primitive we call threshold private set intersection.

3.1 Decentralized Privacy-Preserving Location Proximity

The first paper of this thesis focuses on the problem of location proximity, where principals are willing to reveal only whether they are within a certain distance from each other. The principals are privacy-sensitive, not willing to reveal any further information about their locations, nor the distance.

Privacy-sensitive location information of end users is commonly sent to the LBS in plaintext, trusting a third party is to handle principals' locations. However, due to privacy concerns it is better to avoid trusting third parties. We therefore take the road of making the data computationally unobtainable, encrypting it with a private key known only to the user whom the data concerns.

Many simple approaches to location privacy are based on obfuscating a principal's position. Such techniques often decrease the usability of the service due to the introduction of inaccurate results. These approaches lead to false positives and false negatives. In some cases over 66% of reported positives can be false [16] A major challenge to be addressed is to provide precise results without unnecessary information disclosure.

Homomorphic encryption is an apt tool to give control of location data to the principal whose location is being measured. It allows for them to encrypt the data before dispatching it to the LBS, while still enjoying the service normally. By using SMC, a user may control *what* information is disclosed even after the information has been sent away from devices controlled by the user.

InnerCircle is a privacy-preserving protocol for location proximity requiring merely one round-trip. InnerCircle allows for only the general proximity of a principal, with a radius r, to be disclosed while maintaining privacy of each principal's input. In contrast to most of the related work, we fully dispense with any third parties while maintaining a precise result, yielding no false positives or negatives at all. Further, InnerCircle benefits highly from parallelization in contrast to much previous work which gives better efficiency than existing approaches for realistic parameters.

Statement of Contribution This paper was co-authored with Martìn Ochoa and Andrei Sabelfeld. All authors contributed equally to the technical development and writing of the material.

This paper was published in the proceedings of the 13th IEEE conference on Privacy, Security and Trust (PST 2015).

3.2 Privacy-assured Outsourced Multiplications

This paper is a more theoretical, high-level contribution, presenting a system to compute any arithmetic formula in a privacy-preserving manner using an additively homomorphic encryption scheme. An arithmetic formula can be seen a directed graph where each node is an operation, each source is an input and where there is a single sink. This is illustrated in Figure 1.

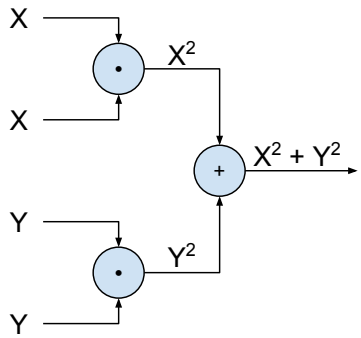

Fig. 1. An arithmetic formula computing $X^2 + Y^2$

A good example of an arithmetic formula that is frequently seen in the literature is to compute the (squared) euclidean distance between two coordinates. The technique is leveraged by a range of protocols within privacy-preserving biometric authentication and privacy-preserving LBS [10, 20, 14, 19, 28]. This can be used to compare feature vectors to find users with similar interests, to compare eigenvectors when comparing images, or for LBS.

The squared euclidean distance is shown in Equation 1 (where x_A and y_A are inputs from principal A and x_B and y_B are inputs from principal B). Recalling that Bob can add and multiply by scalar numbers, one can separate inputs from both parties, such that e.g. A can send three ciphertexts $\alpha = 2x_A, \beta = 2y_A, \gamma = x_A^2 + y_A^2$ to B to let B calculate the distance homomorphically.

$$D = x_A^2 + y_A^2 + x_B^2 + y_B^2 - (2x_A x_B + 2y_A y_B) \tag{1}$$

However, a problem arises if A decides to send three ciphertexts such that $\gamma \neq \left(\frac{\alpha}{2}\right)^2 + \left(\frac{\beta}{2}\right)^2$. In this case, A can trick B into computing another function than the euclidean distance between (x_A, y_A) and (x_B, y_B) which causes unwanted leakage of information.

The novelty in the paper is a privacy-assured multiplication protocol, called BetterTimes. Using BetterTimes, a system for arbitrary arithmetic formulas is proposed, which allows us to let Alice send (x_A, y_A) directly, instead of the three ciphertexts computed from her coordinates. This system can be applied to upgrade much existing work from being secure only against honest-but-curious adversaries to being secure in the malicious adversary model. The approach approach is evaluated using a prototypical implementation. The results show that the added overhead of our approach is small compared to insecure outsourced multiplication.

Statement of Contribution This paper was co-authored with Martìn Ochoa and Andrei Sabelfeld. All authors contributed equally to the technical development and writing of the material.

This paper was published in the proceedings of the 9th LNCS conference on Provable Security (ProvSec 2015).

3.3 Speed-Constrained Location Queries

Combining the two previous papers, we can construct a protocol to test the proximity of Alice and Bob while preserving privacy even in the case when Alice is misbehaving in any arbitrary manner (technically, she is a malicious adversary). However, when considering real-world applications of location proximity, we see that

even with protections against malicious adversaries *in a single run* of the protocol, we are still vulnerable to adversaries that re-run the protocol in order to learn Bob's location. This is usually referred to as a multi-run attacker or continuous querying.

To mitigate these concerns we develop MaxPace, a general policy framework to restrict proximity queries based on the speed of the requester. We demonstrate the boost of privacy by comparative bounds on how the knowledge about the users' location changes over time. The effectiveness of the policy is illustrated in Figure 2, where and unconstrained attacker locates Bob (who's position is marked by a star) in three attempts, while an attacker under the MaxPace policy needs nine attempts.

MaxPace applies to both a centralized setting, where the server can enforce the policy on the actual locations, and a decentralized setting, dispensing with the need to reveal user locations to the service provider. The former has already found a way into practical location-based services. For the latter, we develop a protocol using techniques from both InnerCircle and BetterTimes, which also incorporates the speed constraints in its design. We formally establish the protocol's privacy guarantees and benchmark our prototype implementation to demonstrate the protocol's practical feasibility.

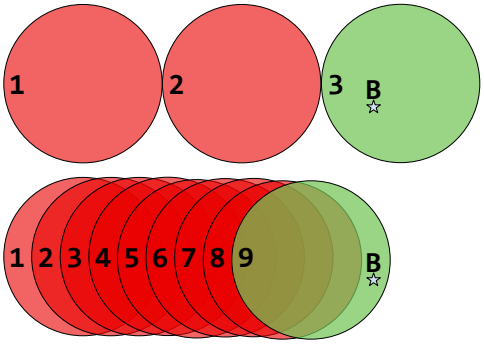

Fig. 2. Different protocols

Statement of Contribution This paper was co-authored with Martìn Ochoa and Andrei Sabelfeld. All authors contributed equally to the technical development and writing of the material.

This paper was published in the proceedings of the IEEE conference on Communications and Network Security (CNS 2016).

3.4 Location-enhanced Authentication using the IoT

User location can act as an additional factor of authentication in scenarios where physical presence is required, such as when making in-person purchases or unlocking a vehicle. This paper proposes a novel approach for estimating user location and modeling user movement using the Internet of Things (IoT). The goal is to utilize the scale and diversity of devices in the IoT to estimate the user's location robustly.

We leverage the increasing number of IoT devices carried and used by them and the smart environments that observe these devices. We also exploit the ability of many IoT devices to "sense" the user. Correct estimation of a user's location can be used to stop adversaries from using compromised user credentials (e.g., stolen keys, credit cards, passwords, etc.) in arbitrary physical locations. An example is given in Figure 3, where a user's devices are observed in their home at 8:00 AM, in a coffee shop at 8:15 AM, and where the user tries to enter their office building at 8:35 AM. The user has left their tablet device at home, but since there are more devices with the user near the office than at home, the system will detect the user's true position in the office. If instead, for instance, the credit card would be in the coffee shop, and all the user's devices in the office, a purchase should not be allowed.

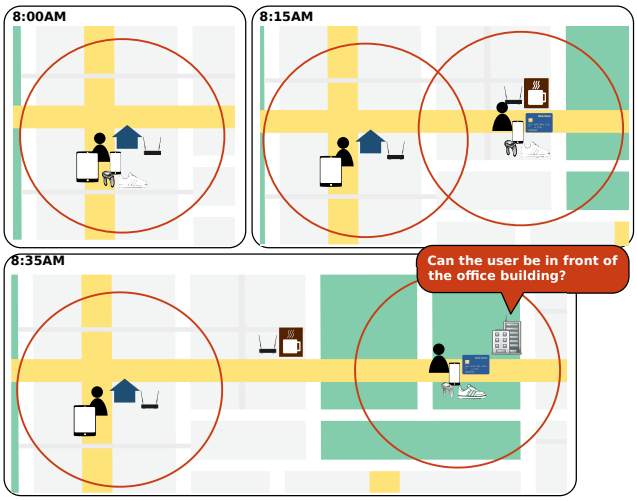

Fig. 3. An example of location-enhanced authentication

To demonstrate how effective and how efficient the approach is, a concrete system was developed, called Icelus. Experiments with Icelus shows that it exhibits a smaller false-rejection rate than for instance smartphone-based location-based authentication and it rejects attackers with few errors (i.e., false acceptances). Icelus collects location and activity data from IoT devices to model user movement and location. Icelus can run as a service on a device of the user, such as a smarthome hub, or it can be hosted in the cloud. To collect data, it organizes the various devices in a hierarchy, so that the ones with Internet connectivity can relay the data of the ones without to the system. Third-party systems can also provide data by directly connecting to Icelus or indirectly by forwarding notifications of certain events (e.g. the use of a credit card at a location, an entry in the user's calendar, etc.). To alleviate

privacy concerns, we also develop a privacy-preserving extension of the protocol used in Icelus that allows us to operate purely on distances, without revealing the actual locations of individual devices. At the core of the extension is a secure multi-party computation protocol that leverages additively homomorphic encryption and blinding.

Statement of Contribution This paper was co-authored with Ioannis Agadakos, Dimitrios Damopoulos, Georgios Portokalidis and Andrei Sabelfeld. I contributed with the privacy-preserving architecture and implementation. All authors contributed equally to the writing of the material.

This paper was published in the proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC 2016).

3.5 Privacy-Preserving Location-Proximity for Mobile Apps

While, as seen above, there has been much recent progress by the research community on developing privacy-enhancing mechanisms for LBS, their evaluation has often focused on the privacy guarantees, while the question of whether these mechanisms can be adopted by practical LBS applications has received limited attention. This paper studies the applicability of privacy-preserving location proximity protocols in the setting of mobile apps. We categorize popular social location-based apps and analyze the trade-offs of privacy and functionality with respect to privacy-enhancing enhancements. To investigate the practical performance trade-offs, we present an in-depth case study of an Android application that implements InnerCircle, a state-of-the-art protocol for privacy-preserving location proximity. This study indicates that the performance of the privacy-preserving application for coarse-grained precision is comparable to real applications with the same feature set.

Statement of Contribution This paper was co-authored with Simonas Stirbys, Omar Abu Nabah and Andrei Sabelfeld. Omar, Simonas and I contributed to the technical development during the project. All authors contributed equally to the writing of the material.

This paper was published in the proceedings of the 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP 2017).

3.6 Privacy-Preserving Ridesharing

Location-based services have revolutionized transportation business, as witnessed by the success of Uber, Lyft, BlaBlaCar, and the like. From a privacy point of view,

these services leave much to be desired. The location of the user is shared with the service, opening up for privacy abuse, as in some recently publicized cases [3].

To mitigate such privacy concerns in ridesharing applications, the last paper of the thesis presents PrivatePool, a model for privacy-preserving ridesharing. While primitives like proximity-testing are rather easy to define, ridesharing is a "large" concept. We focus on scenarios more aligned with car-pooling approach taken by BlaBlaCar, rather than the taxi-like structure like that of Uber. We formalize the case when two users specify an origin and a destination of a trip, and they want to find out whether they can share a ride in a privacy-preserving manner. Our resulting model is rather complex, and creating even a privacy-insensitive system to match rides in this manner proves rather impractical. Instead, we focus on two corner cases. The first is when both the origins and the destinations for the two users are close, which we call endpoint-matching. The second is when a large portion of the routes overlap, which we call intersection-based matching. Intersection-based matching can be useful in many cases even if the endpoints are very far apart. The two situations are depicted in Figure 4, where the left image shows end-point matching and on the right-hand side we can see intersection-based matching.

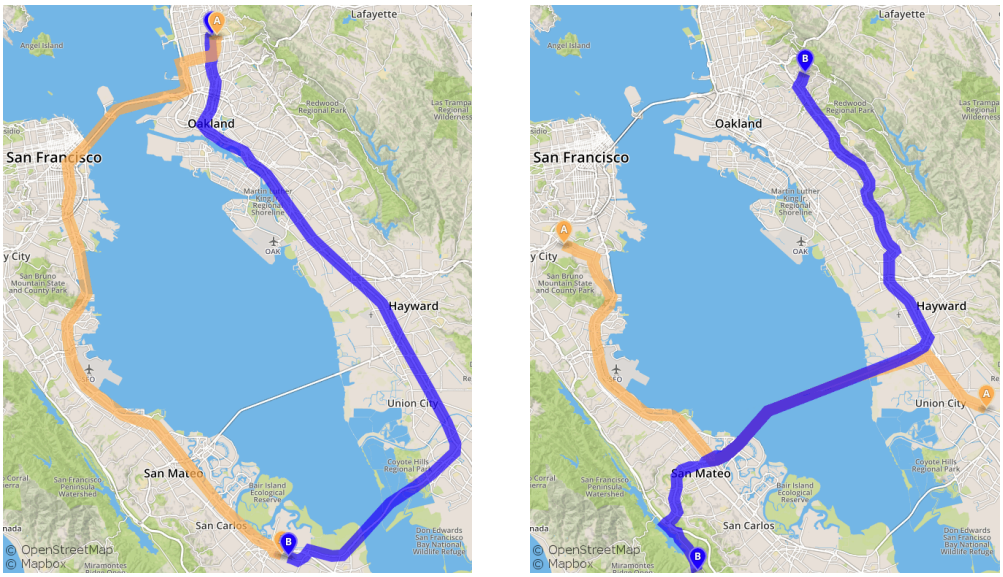

Fig. 4. End-point matching and route intersection

The paper presents secure multi-party computation techniques for endpoint and intersection-based matching that allow the protocols to be run without trusting third parties. At the same time, the users learn of a ride segment they can share and noth-

ing else about other users' location. For endpoint matching, we build on InnerCircle to create an SMC protocol to detect if both of the endpoints are sufficiently close. For intersection-based matching, we created a novel cryptographic technique, called threshold key encapsulation (T-KEM). We plug T-KEM into existing solutions for privately computing the intersection of two sets, one held by each user. This fits nicely in our case, if the two sets are defined by the points traversed during their trips.

Statement of Contribution This paper was co-authored with Claudio Orlandi and Andrei Sabelfeld. All authors contributed equally to the technical development and writing of the material.

This paper will be published in the proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF 2017).

4 Future work

There are still much to do in the interest of achieving better location privacy. While many of the cryptographic techniques used in this thesis can be called practical out-of-context, a fully-fledged system built on top of SMC needs more work before it is useful in practice.

On the one hand, practically-oriented research results are needed to show how to use SMC in general without information leakage from a running system. As SMC works on the application layer, it is oblivious to information on other levels such as IP addresses etc. Each user currently needs to trust the environment which is running the application both on their machine and on the machine which they are communicating with, and they need to trust the service that is distributing the binary for the application they are running.

On the other hand, further foundational work is needed in terms of scalability to large numbers of users. The state of the art is making great leaps in this direction, such that is now possible to efficiently compute a fixed function of many users [15]. However, state-of-the art SMC protocols are only efficient for a limited number of users for cases like ours where each party wants to evaluate a different function, as the question "who can *I* share a ride with?" is context-sensitive.

On another note, I also feel there is a need for more work from the software-engineering community. It is very different to debug an application with and without access to the values stored in each variable. We need to adapt processes and tools, to discuss things such as logging, backup, and many other issues which are more-or-less solved in the traditional setting also in the encrypted domain.

5 Conclusions

The thesis presents usages of SMC within location-based services and an augmentation of additively homomorphic encryption to add a privacy-guaranteed multiplication functionality. All of these serve to some extent in giving the far end of the information exchange more control over data disclosure, moving away from centralized architectures relying on trust and achieving a higher level of privacy. Trust in third parties is through these techniques reduced, and the owner of data can be more confident that it is handled and used as they intend.

References

1. Ahern, S., Eckles, D., Good, N., King, S., Naaman, M., and Nair, R. Over-exposed?: privacy patterns and considerations in online and mobile photo sharing. In *CHI* (2007), M. B. Rosson and D. J. Gilmore, Eds., ACM, pp. 357–366.
2. Beresford, A. R., and Stajano, F. Location privacy in pervasive computing. *IEEE Pervasive Computing 2*, 1 (2003), 46–55.
3. Bessette, C. Does Uber Even Deserve Our Trust? http://www.forbes.com/sites/chanellebessette/2014/11/25/does-uber-even-deserve-our-trust/, Nov. 2014.
4. Bogdanov, D., Laur, S., and Willemson, J. Sharemind: A framework for fast privacy-preserving computations. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings* (2008), S. Jajodia and J. López, Eds., vol. 5283 of *Lecture Notes in Computer Science*, Springer, pp. 192–206.
5. Bogetoft, P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen, T. P., Krøigaard, M., Nielsen, J. D., Nielsen, J. B., Nielsen, K., Pagter, J., Schwartzbach, M. I., and Toft, T. Secure multiparty computation goes live. In *Financial Cryptography* (2009), R. Dingledine and P. Golle, Eds., vol. 5628 of *Lecture Notes in Computer Science*, Springer, pp. 325–343.
6. Brush, A. J. B., Krumm, J., and Scott, J. Exploring end user preferences for location obfuscation, location-based services, and the value of location. In *UbiComp* (2010), J. E. Bardram, M. Langheinrich, K. N. Truong, and P. Nixon, Eds., ACM International Conference Proceeding Series, ACM, pp. 95–104.
7. Carter, H., Mood, B., Traynor, P., and Butler, K. R. B. Secure outsourced garbled circuit evaluation for mobile devices. In *USENIX Security* (2013), S. T. King, Ed., USENIX Association, pp. 289–304.
8. Diener, E., Fraser, S. C., Beaman, A. L., and Kelem, R. T. Effects of deindividuation variables on stealing among halloween trick-or-treaters. *Journal of personality and social psychology 33*, 2 (1976), 178.

9. DUCKHAM, M., AND KULIK, L. Location privacy and location-aware computing. *Dynamic & mobile GIS: investigating change in space and time 3* (2006), 35–61.
10. ERKIN, Z., FRANZ, M., GUAJARDO, J., KATZENBEISSER, S., LAGENDIJK, I., AND TOFT, T. Privacy-preserving face recognition. In *Privacy Enhancing Technologies* (2009), I. Goldberg and M. J. Atallah, Eds., vol. 5672 of *Lecture Notes in Computer Science*, Springer, pp. 235–253.
11. GENTRY, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009* (2009), M. Mitzenmacher, Ed., ACM, pp. 169–178.
12. GHORBANI, A. A., TORRA, V., HISIL, H., MIRI, A., KOLTUKSUZ, A., ZHANG, J., SENSOY, M., GARCÍA-ALFARO, J., AND ZINCIR, I., Eds. *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015* (2015), IEEE.
13. GUROV, D., LAUD, P., AND GUANCIALE, R. Privacy preserving business process matching. In Ghorbani et al. [12], pp. 36–43.
14. HALLGREN, P. A., OCHOA, M., AND SABELFELD, A. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In Ghorbani et al. [12], pp. 1–6.
15. KERSCHBAUM, F. Adapting privacy-preserving computation to the service provider model. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, Vancouver, BC, Canada, August 29-31, 2009* (2009), IEEE Computer Society, pp. 34–41.
16. MASCETTI, S., FRENI, D., BETTINI, C., WANG, X. S., AND JAJODIA, S. Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *VLDB J. 20*, 4 (2011), 541–566.
17. PAILLIER, P. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 223–238.
18. RIVEST, R. L., ADLEMAN, L., AND DERTOUZOS, M. L. On data banks and privacy homomorphisms. *Foundations of secure computation 32*, 4 (1978), 169–178.
19. SADEGHI, A.-R., SCHNEIDER, T., AND WEHRENBERG, I. Efficient privacy-preserving face recognition. In *ICISC* (2009), D. Lee and S. Hong, Eds., vol. 5984 of *Lecture Notes in Computer Science*, Springer, pp. 229–244.
20. SEDENKA, J., AND GASTI, P. Privacy-preserving distance computation and proximity testing on earth, done right. In *ASIACCS* (2014), S. Moriai, T. Jaeger, and K. Sakurai, Eds., ACM, pp. 99–110.
21. SHAMIR, A. How to share a secret. *Commun. ACM 22*, 11 (1979), 612–613.
22. Universal declaration of human rights, Dec. 1948. Available: http://www.ohchr.org/EN/UDHR/Documents/UDHR_Translations/eng.pdf (accessed 2017-07-06).
23. TURNER, S., AND POLK, T. Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176 (Proposed Standard), Mar. 2011.

24. Warren, S. D., and Brandeis, L. D. The right to privacy. *Harward Law Review 4*, 5 (December 1890), 193–220.
25. Webb, E. J., Campbell, D. T., Schwartz, R. D., and Sechrest, L. *Unobtrusive measures: Nonreactive research in the social sciences*, vol. 111. Rand McNally Chicago, 1966.
26. Yao, A. C. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982* (1982), IEEE Computer Society, pp. 160–164.
27. Ylonen, T., and Lonvick, C. The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard), Jan. 2006.
28. Zhong, G., Goldberg, I., and Hengartner, U. Louis, lester and pierre: Three protocols for location privacy. In *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers* (2007), N. Borisov and P. Golle, Eds., vol. 4776 of *Lecture Notes in Computer Science*, Springer, pp. 62–76.

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.

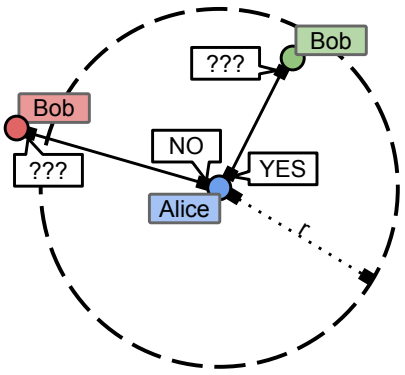

Fig. 1. Privacy-preserving location proximity

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].

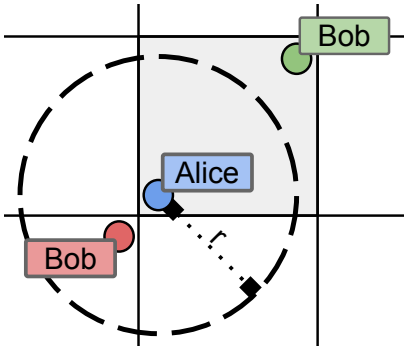

Fig. 2. Grid-based testing

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

3.

Procedure 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:*

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

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 $\stackrel{c}{\equiv}$ 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 us-

ing 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 func-

tionality. 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.

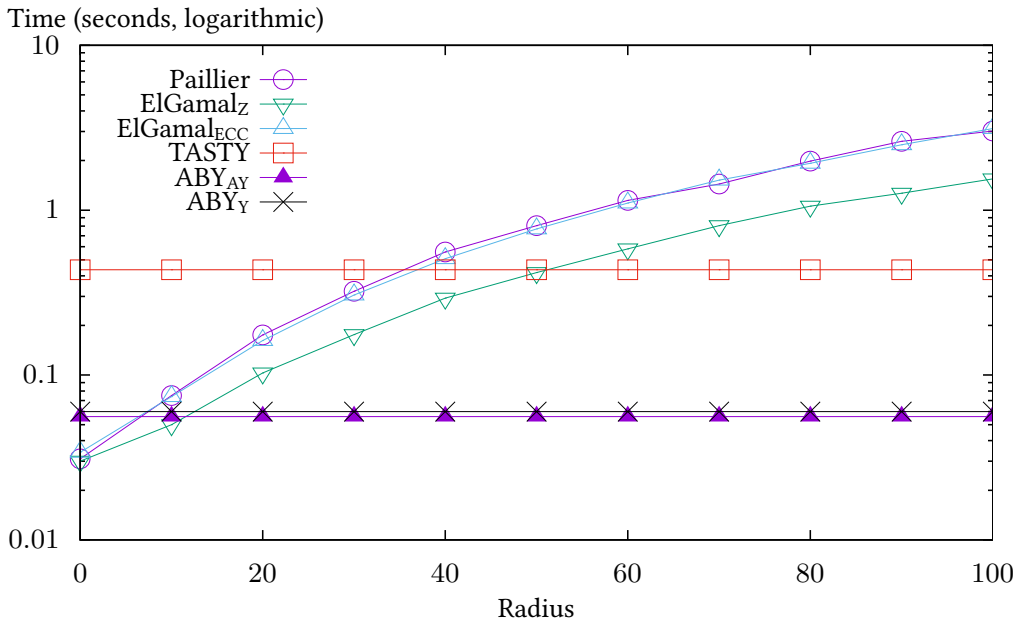

Fig. 4. Time consumption for different algorithms and values of r using 80 bits of security

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.

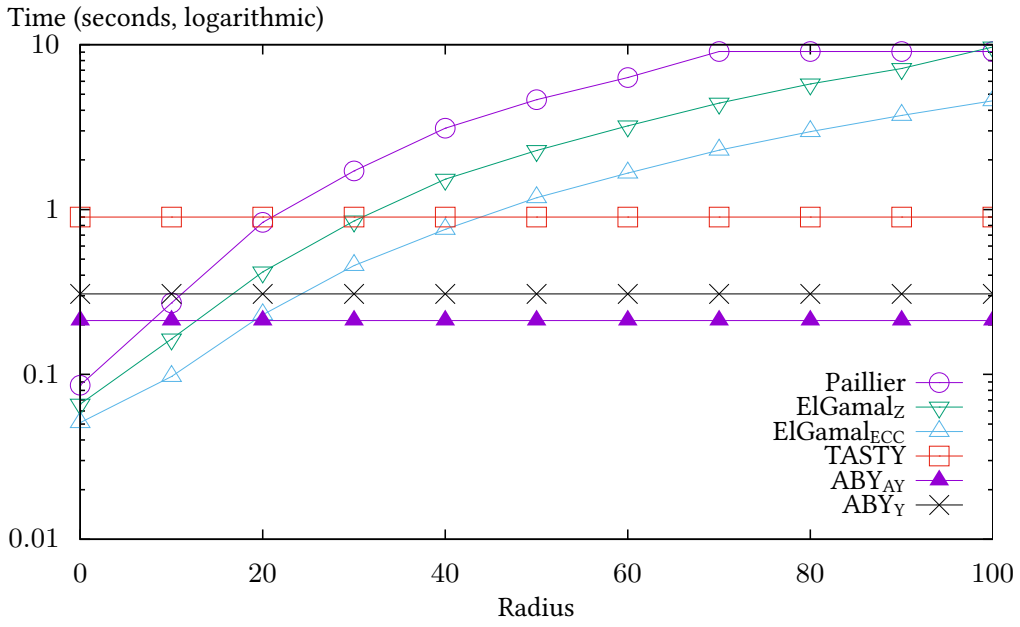

Fig. 5. Time consumption for different algorithms and values of r using 112 bits of security

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 $X \stackrel{c}{\equiv} Y$ 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 :	**Game** MM-IND-CPA :
$(m_0, m_1) \leftarrow \mathcal{A}_1$;	$((m_0^0, ..., m_k^0), (m_0^1, ..., m_k^1)) \leftarrow \mathcal{A}_1$;
$b \xleftarrow{\$} \{0, 1\}$;	$b \xleftarrow{\$} \{0, 1\}$;
$b' \leftarrow \mathcal{A}_2(E_K(m_b))$;	$b' \leftarrow \mathcal{A}_2(E_K(m_0^b), ..., E_K(m_k^b))$;
return $b = b'$	return $b = b'$

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

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

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

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:

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

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-

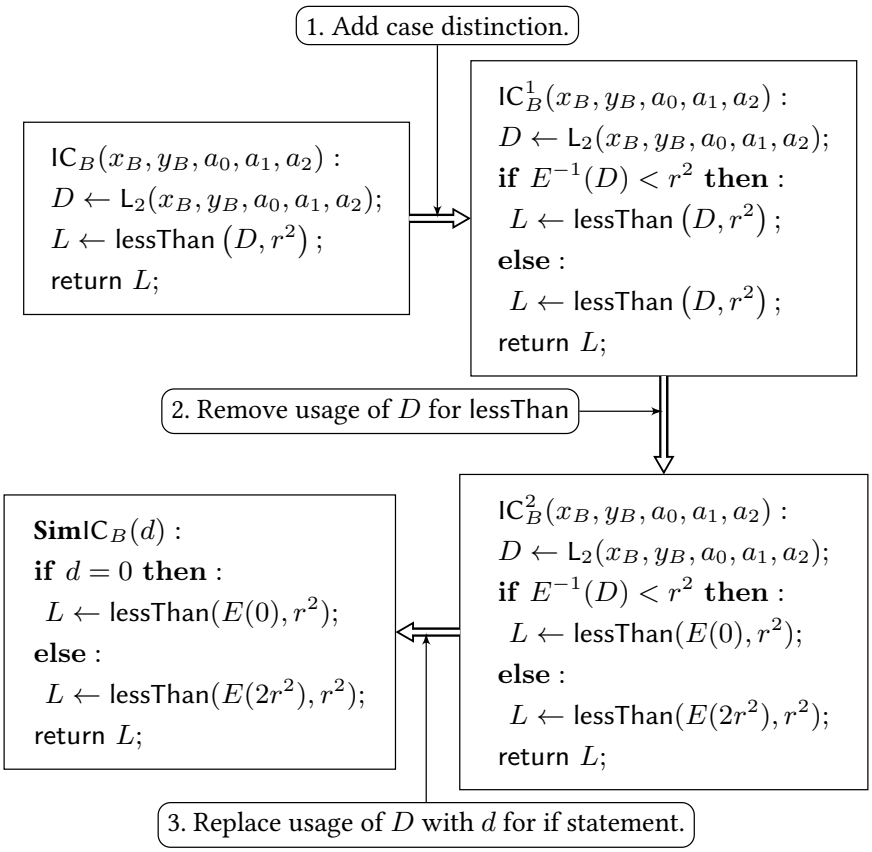

Fig. 8. Simulating IC_B

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:

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

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;

$\mathbf{S}_C()$:
$a_0 \leftarrow E(0)$;
$a_1 \leftarrow E(0)$;
$a_2 \leftarrow E(0)$;
$L \leftarrow \mathsf{lessThan}\,(a_0, 0)$;
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$_{\text{Y}}$ requires 261 KB and 393 KB of traffic within a protocol run for 80 and 112 bits of security, respectively. ABY$_{\text{AY}}$ 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 $\overline{\oplus^n}$, $\overline{\odot^n}$, $\overline{\neg^n}$ and $\overline{\mathcal{R}^n}$ 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 $\mathcal{O}\left(r^2 \cdot \left(\mathcal{E}^n + \overline{\oplus^n} + \overline{\neg^n} + \overline{\mathcal{R}^n}\right)\right)$.

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:

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

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

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 $\mathcal{O}(\mu(n)) \subset \mathcal{O}(\eta(n,n)) \subset \mathcal{O}(\psi(n))$.

Table 5. Asymptotic cost of concrete implementations

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

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:

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

Paper **two**

BetterTimes

Privacy-assured Outsourced Multiplications for Additively Homomorphic Encryption on Finite Fields

Per A. Hallgren, Martìn Ochoa and Andrei Sabelfeld

We present a privacy-assured multiplication protocol using which an arbitrary arithmetic formula with inputs from two parties over a finite field $\mathbb{F}_p$ can be jointly computed on encrypted data using an additively homomorphic encryption scheme. Our protocol is secure against malicious adversaries. To motivate and illustrate applications of this technique, we demonstrate an attack on a class of known protocols showing how to compromise location privacy of honest users by manipulating messages in protocols with additively homomorphic encryption. We evaluate our approach using a prototypical implementation. The results show that the added overhead of our approach is small compared to insecure outsourced multiplication.

Published in the 9th International Conference on Provable Security
ProvSec 2015, Kanazawa, Japan, November 24-26, 2015

1 Introduction

There has been an increase of the public awareness about the importance of privacy. This has become obvious with cases such as Snowden [37] and the Tor project [11]. Unfortunately, the current practice is not yet to address privacy concerns by design [7, 36, 27, 32]. It is by far more common that the end consumer has to send privacy-sensitive information to service providers in order to achieve a certain functionality, rather than the service using privacy-preserving technologies. A major challenge of today's research community is to enable services to address privacy without hampering sought functionality and efficiency.

Recent years have brought much attention to secure computations distributed among several participants, a subfield of cryptography generally known as *Secure Multi-party Computation* (SMC). SMC has in recent years been brought to the brink of being widely applicable to real world scenarios [4, 3], although general purpose solutions with strong security guarantees are still too slow to be widely applied in practice.

This paper proposes a novel and efficient approach to jointly compute an arbitrary arithmetic formula using certain additively homomorphic encryption schemes, while maintaining security against malicious adversaries. The solution is shown to be valuable as a vital complement to boost the security of a class of privacy-preserving protocols [12, 34, 19, 33, 38], where *Alice* queries *Bob* for a function over their combined inputs (see Figure 2). In such scenarios, it is common that *Bob* is intended to learn nothing at all, while still providing *Alice* with useful information such as whether a picture of a face matches a database [33, 12] or whether two principals are close to each other [19, 34, 38]. This work allows such solutions to harden the attacker model from *honest-but-curious* to *malicious* attackers that do not necessarily follow the protocol (both attacker models are standard in SMC and are presented for instance in [17, 28]).

Although some connections have been identified [30, 34, 19], the two communities of Privacy-preserving Services and Secure Multi-Party Computations are still largely separated. One of the goals of this paper is to contribute to bridging the gap, in particular when it comes to rigorously improving the security of efficient protocols using additively homomorphic encryption in the presence of honest-but-curious adversaries, enabling them to also protect against malicious adversaries in an efficient manner.

Problem statement In general in secure two-party computation [28] one considers the case where two parties, *Alice* with inputs $\overrightarrow{x}$ and *Bob* with inputs $\overrightarrow{y}$, want to

compute a functionality $f(\overrightarrow{x}, \overrightarrow{y}) = (g(\overrightarrow{x}, \overrightarrow{b}), h(\overrightarrow{x}, \overrightarrow{y}))$, where the procedure f yields a tuple in which *Alice*'s output is the first item and *Bob*'s output is the second item. For the scope of this work, h is always the empty string, and the inputs of both parties are in $\mathbb{F}_p$, such that $\forall x_i \in \overrightarrow{x} : x_i \in \mathbb{F}_p$ and $\forall y_i \in \overrightarrow{y} : y_i \in \mathbb{F}_p$. That is, *Alice* obtains the result of g whereas *Bob* observes nothing (as usual when using partial or full homomorphic encryption). For this reason, in the following we will refer only to $g(\overrightarrow{x}, \overrightarrow{y})$ as the functionality.

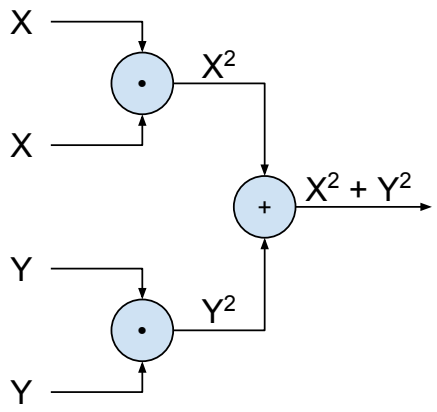

Fig. 1. Arithmetic formula computing $x^2 + y^2$.

Moreover, we set $g(\overrightarrow{x}, \overrightarrow{y})$ to be an arbitrary arithmetic formula over $\overrightarrow{x}$ and $\overrightarrow{y}$ in the operations $(\cdot, +)$ of $\mathbb{F}_p$, that is an arithmetic circuit [35] that is also a directed tree, as the one depicted in Figure 1.

We assume as usual that both *Alice* and *Bob* want *privacy* of their inputs, as much as it is allowed by g. *Bob* is willing to reveal the final output of g, but not any intermediate results, or a different function g' that would compromise the privacy of his inputs. More precisely, we want a secure two-party computation in the malicious adversary model for a malicious *Alice* [28], as depicted in Figure 2.

Note that additions in the formula can be done correctly by *Bob* without the help of *Alice* when using an additively homomorphic encryption scheme. This holds also for all multiplications involving *Bob*'s input only, and multiplications with a ciphertext and a value known to *Bob*. The only operations outside of the scope of the additively homomorphic capabilities are multiplications involving inputs from *Alice* only. For instance in Figure 1, *Bob* can not compute x^2 (assuming x is a private input to *Alice*). In this work therefore we focus on a protocol such that *Bob* can outsource such multiplications to *Alice* without disclosing the value of the operands,

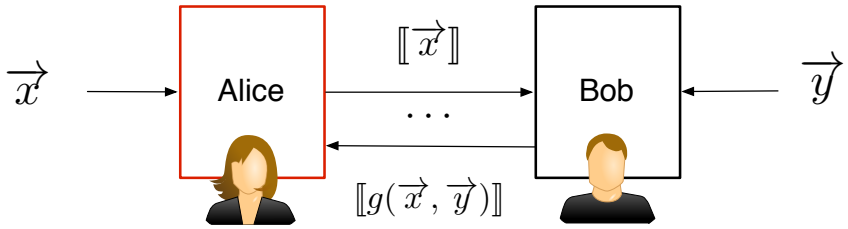

Fig. 2. High-level view of a 2-party computation based on homomorphic encryption, where $[\![\cdot]\!]$ denotes encryption under the public key of *Alice*.

and such that if *Alice* does not cooperate, the final value of the arithmetic formula is corrupted and useless to her. This will allow us to show that our protocol is fully privacy-preserving in the malicious adversary model of SMC.

Fairness of the computation (that is, all parties receive their intended output) is out of scope for two reasons: it is impossible to guarantee this property for two-party computations in the malicious model [28], but more interestingly, note that since by construction *Bob* is allowed to observe nothing, an early abortion of the protocol by *Alice* will only hamper fairness for herself.

Contributions The paper outlines a novel protocol *BetterTimes* which lets *Bob* outsource multiplications using an additively homomorphic encryption scheme (where he does not hold the private key) while asserting privacy of his inputs. *BetterTimes* provides *Bob* not only with the encrypted product but also the encryption of an assurance value (a field element $a \in \mathbb{F}_p$) which is a random value in $\mathbb{F}_p^*$ if *Alice* does not follow the protocol and an encryption of 0 otherwise. The assurance is added to the final output of g thus making the result useless to *Alice* if she tries to cheat. Our contribution thus brings the state-of-the art forward by efficiently giving *Bob* guarantees in the case that *Alice* is malicious.

We illustrate the usefulness of our approach for a class of protocols from the literature [12, 34, 19, 33, 38], which compute whether the distance between two vectors in the plane is less than a threshold. In the presence of malicious adversaries, leakage of private information is possible. A solution using our technique is presented for these protocols. Moreover, we make our implementation fully available to the community[1].

[1] `https://bitbucket.org/hallgrep/bettertimes`

Relation to Zero Knowledge Proofs An alternative solution to the presented problem would be to use a Zero Knowledge schema such that *Bob* can verify that a ciphertext corresponds to a certain multiplication. Such a schema is guaranteed to exist given the general theorem of Goldreich et al. [18]. However, to the best of our knowledge it is not straightforward to constructively devise such a scheme for a given additively homomorphic cryptosystem. Our solution in contrast does not require *Bob* to be able to verify whether a multiplication is correct, but by construction will render the final computation result useless to malicious adversaries.

In a nutshell, the novelty as compared to zero-knowledge proofs is based on the simple realization that *Bob* does not need to know whether *Alice* is cheating or not in order to assure the correctness of the final computation and the privacy of his inputs, which decreases the number of round-trips that such a verification step implies. This is a special case of the *conditional disclosure of secrets* introduced by Gertner et al. [16], where a secret is disclosed using SMC only if some condition is met. In our case, the condition is that $z_i = x_i \cdot y_i$ for each multiplication in the formula, and the secret is the output of g.

To the best of our knowledge, there is no previous solution to accomplish secure outsourced multiplications for additively homomorphic encryption in the malicious model without the use of zero-knowledge proofs.

Outline The paper first introduces necessary background and notation in Section 2. Following, in Section 3 the BetterTimes protocol is described, and its application to computing arbitrary arithmetic formulas is discussed. Section 4 presents the security guarantees in the malicious adversary setting. Section 5 presents benchmarks that allow one to estimate which impact the approach would have in comparison to only protecting against semi-honest adversaries. Section 6 positions this work in perspective to already published work. Finally, Section 7 summarizes the material presented in this paper. Before delving into details, a concrete application of the proposed solution is outlined in Section 1.1.

1.1 Exploits for Proximity Protocols

We illustrate the usefulness of our approach by an attack on a class of protocols from the literature [12, 34, 19, 33, 38], which compute whether the distance between two vectors in the plane is less than a threshold in a privacy-preserving manner. Popular applications of this algorithm are geometric identification and location proximity. For concreteness, this section focuses on the distance computation used in the *InnerCircle* protocol by Hallgren et al. [19]. The same attack also applies to the other

representatives of the same class of protocols [12, 34, 33, 38], but in many cases a successful exploit does not have as visible effects.

Hallgren et al. present a protocol for privacy-preserving location proximity. It is based on the fact that *Bob* can compute the euclidean distances from a point represented as three ciphertexts $[\![2x]\!]$, $[\![2y]\!]$ and $[\![x^2+y^2]\!]$ to any other point known by *Bob* using additively homomorphic encryption (here $[\![\cdot]\!]$ stands for encryption under the public key of *Alice*). A problem with the approach is that *Bob* has no knowledge of how the ciphertexts are actually related, he sees three ciphertexts $[\![\alpha]\!]$, $[\![\beta]\!]$ and $[\![\gamma]\!]$. In the case that $\gamma \neq (\alpha/2)^2 + (\beta/2)^2$, subsequent computations may leak unwanted information. The distance is expressed as the (squared) distance as shown in Equation (1), computed homomorphically as shown in Equation (2) where only some of *Bob*'s inputs are needed in plaintext.

$$D = x_A^2 + y_A^2 + x_B^2 + y_B^2 - (2x_A x_B + 2y_A y_B) \tag{1}$$

$$[\![D]\!] = [\![x_A^2 + y_A^2]\!] \oplus [\![x_B^2 + y_B^2]\!] \ominus ([\![2x_A]\!] \odot x_B \oplus [\![2y_A]\!] \odot y_B) \tag{2}$$

Here, $\oplus$, $\ominus$ and $\odot$ are the homomorphic operations which in the plaintext space map to $+$, $-$ and $\cdot$ respectively (see Section 2). Now, by replacing the information sent by *Alice* by α, β and γ and observing that *Alice* can choose α and β arbitrarily, the expression becomes as in Equation (3):

$$D = x_B^2 + y_B^2 + \gamma + \alpha x_B + \beta y_B \tag{3}$$

The effects of the attack are very illustrative in [34, 19, 38]. In these works, *Bob* wants to return a boolean $b = (r^2 > D)$ indicating whether two principals are within r from each other. Thus the result given to *Alice* is the evaluation of the function $r^2 > x_B^2 + y_B^2 + \alpha x_B + \beta y_B + \gamma$. This is equivalent to the result of $r^2 - \gamma > x_B^2 + y_B^2 + \alpha x_B + \beta y_B$. Given that *Alice* knows r, she can encode it into the manipulated variables thus forcing the evaluation of $\delta > x_B^2 + y_B^2 + \alpha x_B + \beta y_B + \eta$, with $\gamma = r^2 - \delta - \eta$. By changing α, β and η, *Alice* can move the center of the queried area, and by tweaking δ she can dictate the size of the area, causing unwanted and potentially very serious information leakage (for instance by querying in arbitrarily located and precise areas such as buildings).

Securing affected protocols Based on the novel asserted multiplication presented in Section 3, a new structure for the protocols of Hallgren et al. can be constructed. Similar amendments can easily be constructed in similar form for other afflicted solutions [12, 34, 33, 38]. Using the system proposed in this paper, it is possible to send

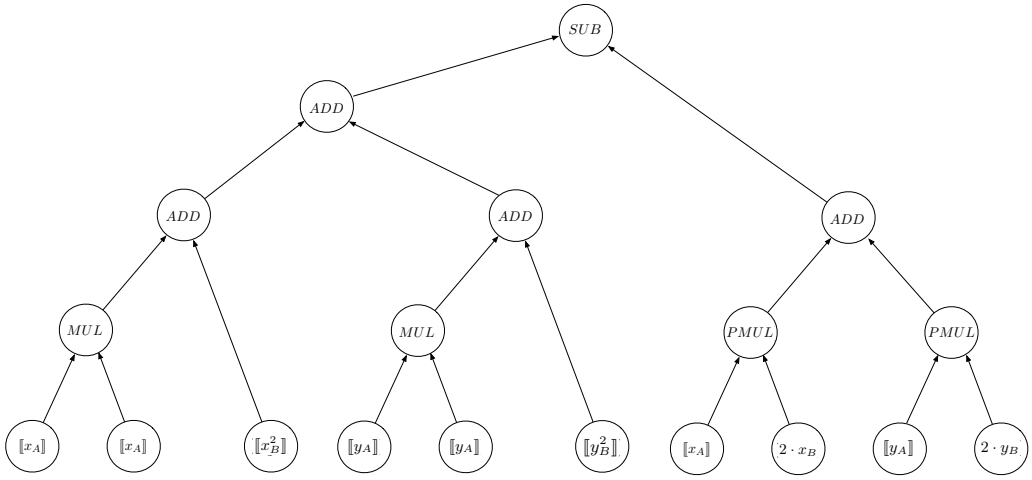

Fig. 3. Tree depicting computation of a secured version of the protocol.

only the encryption of x_A and y_A in the initial message, and securing the necessary squaring by means of *BetterTimes*.

An arithmetic formula which computes the distance directly using x_A, y_A, x_B and y_B is already defined in Equation (1). Now remains only to model this such that it can be computed by the system presented later in this paper, after which the protocols can proceed to compute the proximity result as they would normally.

The result is an algorithm modeled using the recursive data structure Ins, which simply is passed to the procedure $evaluate$ by *Bob*, see Section 3 and Figure 6. The formula of can be depicted as a tree as in Figure 3. The concrete instructions (instances of Ins) are spelled out in Appendix 1.

2 Background

The solution proposed in this paper makes use of any additively homomorphic encryption scheme which provides semantic security and where the plaintext space is a field (for instance such as the DGK Scheme [9]). For a definition of semantic security see [2].

Additively Homomorphic Encryption Schemes Here and henceforth, k is the private key belonging to *Alice* and K and is the corresponding public key. Let the plaintext space $\mathcal{M}$ be isomorphic to the field $(\mathbb{Z}_p, \cdot, +)$ for some prime number p and the ciphertext space $\mathcal{C}$ such that encryption using public key K is a function $E : \mathcal{M} \to \mathcal{C}$ and decryption using a private key k is $D : \mathcal{C} \to \mathcal{M}$.

The vital homomorphic features which is used later in the paper is an addition function $\oplus : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, a unary negation function $\neg : \mathcal{C} \to \mathcal{C}$, and a multiplication function $\odot : \mathcal{C} \times \mathcal{M} \to \mathcal{C}$.

$$E(m_1) \oplus E(m_2) = E(m_1 + m_2) \tag{4}$$

$$\neg E(m_1) = E(-m_1) \tag{5}$$

$$E(m_1) \odot m_2 = E(m_1 \cdot m_2) \tag{6}$$

Note that in a finite field any non-zero element multiplied with a non-zero random element yields a non-zero uniformly distributed element. Formally:

$$E(m_1) \odot \rho = \begin{cases} E(0) \text{ if } m_1 = 0 \\ E(l) \text{ with } l \in \mathcal{M}^{\mathcal{U}} \text{ otherwise} \end{cases}, \text{with } \rho \in \mathcal{M}^{\mathcal{U}} \tag{7}$$

where $m_1 \in \mathcal{M}$, $m_2 \in \mathcal{M}$ and $\mathcal{M}^{\mathcal{U}}$ is a uniformly random distribution of all elements in $\mathcal{M} \setminus \{0\}$.

Syntax and conventions For readability, the operations $\oplus, \odot, \neg, E$ and D do not have a key associated to them, we assume they all use the usual k, K pair where *Alice* holds k. 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 $c_1 \ominus c_2$. For further brevity, a ciphertext c encrypting a plaintext p under the public key of *Alice* is denoted as $[\![p]\!]$.

The protocol descriptions in Figure 5 and Figure 6 are given in the language pWHILE [1]. For the convenience of the reader a few constructs used in the paper are outlined here, but for details the reader is directed to [1]. $a \leftarrow b$ 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.

Security Concepts In the following we briefly recall some fundamental concepts from SMC that will be useful for the security guarantees discussion of Sect. 4.

Definition 1 (Negligible functions). *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.

Definition 2 (Indistinguishability).

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:*

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

3 Arithmetic formulas through assured Multiplication

As previously discussed, our goal is a system which can compute any arithmetic formula in the presence of a malicious *Alice* (who holds the private key), without leaking any information derived from *Bob*'s inputs except the result of g. To show how to reach this, we first outline the primary building block, *BetterTimes*.

3.1 Privacy-assured Outsourced Multiplication

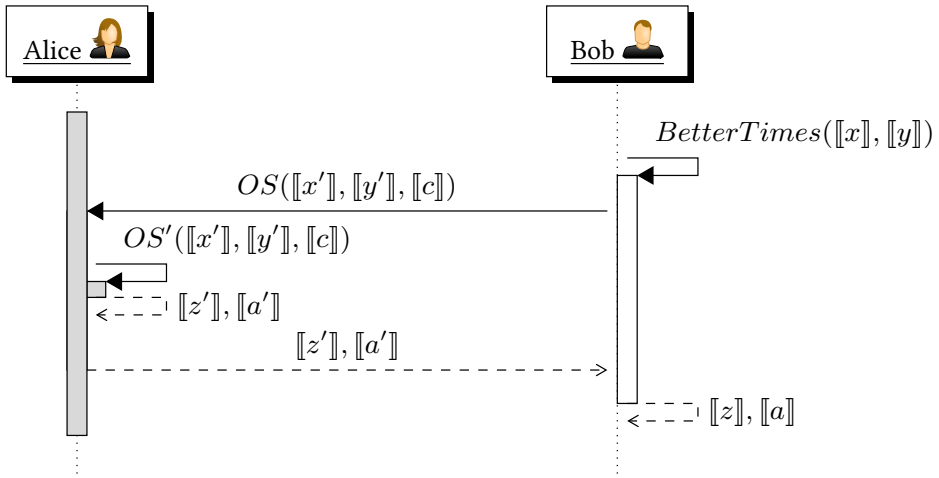

Fig. 4. Visualization of the attested multiplication protocol

The core of the solution is a novel outsourced multiplication protocol with privacy guarantees, *BetterTimes*. The protocol is visualized in Figure 4 and detailed in Figure 5. *BetterTimes* allows *Bob* to calculate a multiplication by outsourcing to *Alice*, while retaining an attestation value with which it is possible to make sure that *Alice* can learn no unintended information.

The two principals interact only once during BetterTimes, where *Bob* contacts *Alice* through the procedure OS (for outsource), defined in Figure 5. As a result of this interaction, *Bob* can compute a value $[\![z]\!]$ which corresponds to the encryption

Proc. $BetterTimes([\![x]\!], [\![y]\!])$:
$c_a \xleftarrow{\$} \{0..p\}; c_m \xleftarrow{\$} \{1..p\};$
$b_x \xleftarrow{\$} \{0..p\}; b_y \xleftarrow{\$} \{0..p\};$
$\rho \xleftarrow{\$} \{1..p\};$
// Blind operands
$[\![x']\!] \leftarrow [\![x]\!] \oplus [\![b_x]\!]; [\![y']\!] \leftarrow [\![y]\!] \oplus [\![b_y]\!];$
// Create challenge
$[\![c]\!] \leftarrow ([\![x']\!] \oplus [\![c_a]\!]) \odot c_m;$
// Outsource multiplication
$([\![z']\!], [\![a']\!]) \leftarrow OS([\![x']\!], [\![y']\!], [\![c]\!]);$
// Compute assurance value
$[\![a]\!] \leftarrow ([\![a']\!] \ominus [\![z']\!] \odot c_m \ominus [\![y']\!] \odot (c_a \cdot c_m)) \odot \rho;$
// Un-blind multiplication
$[\![z]\!] \leftarrow [\![z']\!] \ominus ([\![x']\!] \odot b_y \oplus [\![y']\!] \odot b_x \oplus [\![b_x \cdot b_y]\!])$;
return $([\![a]\!], [\![z]\!])$;

Proc. $OS([\![x]\!], [\![y]\!], [\![c]\!])$:
return $((E(D([\![x]\!]) \cdot D([\![y]\!]),$
$E(D([\![c]\!]) \cdot D([\![y]\!])))$;

Fig. 5. The attested multiplication protocol

of the multiplication $x \cdot y$ if *Alice* is honest and an attestation value a which will be uniformly random if *Alice* does not comply with the protocol. *Alice* can only deviate from the protocol by using $OS' \neq OS$.

BetterTimes contains several random variables, here follows a brief explanation of their names to make the procedures easier to follow. The first two, c_a and c_m, serve to construct the challenge c used in the attestation. c_a and c_m are an additive and multiplicative component, respectively. The second pair, b_x and b_y, are used to blind the operands x and y, respectively, when outsourcing the multiplication. Finally ρ is used to make sure that an attestation which doesn't match the supplied product causes a random offset of the final result.

Note that the attestation is only needed when outsourcing a multiplication. The blinding used in BetterTimes has also been presented and used by, among others, Kolesnikov et al. [22]. The construction using the challenges c_a and c_m yield the following computations in the plaintext, starting with the attestation value a in Equation (8). Through the procedure OS, *Alice* replies with (in the plaintexts) as in Equation (9). Thus, assuming *Alice* is honest, we see that Equation (10) must hold.

$$a = (a' - z' \cdot c_m - y' \cdot c_a \cdot c_m) \cdot \rho = (a' - z' - y' \cdot c_a) \cdot c_m \cdot \rho \tag{8}$$

$$a' = ((x' + c_a) \cdot c_m) \cdot y' = (x' \cdot y' + y' \cdot c_a) \cdot c_m \tag{9}$$

$$a = (x' \cdot y' - z') \cdot c_m \cdot \rho \tag{10}$$

Since by assumption *Alice* is honest, $z' = x' \cdot y' \implies a = 0$. To see that this is the case if and only if *Alice* honest, see Section 4.

3.2 Privacy-assured Arithmetic Formulas

Using *BetterTimes* as described above, the following discusses how to construct arbitrary arithmetic formulas. The general idea is to accumulate any errors caused by misbehavior by *Alice* using attestations a_j, one for each outsourced multiplication. The other operations require no attestations as they can be calculated locally by *Bob*. If *Alice* is dishonest during an outsourced multiplication, the corresponding attestation a_j is a uniformly random variable. Once an arithmetic formula has been fully evaluated, and the result obtained as $[\![result]\!]$, *Bob* instead returns the value $[\![result]\!] \oplus \sum a_i$. The returned value is $[\![result]\!]$ if and only if *Alice* is honest, and the encryption of a uniformly random field element if she is dishonest.

Given an arbitrary arithmetic formula g, the system is designed using a recursive data structure **Ins**, modeling an *instruction* representing g. An instruction either contains an operation and two operands or a scalar. Formally, **Ins** $\in \{[o, l, r]|x\}$, where o is the operator, l and r are the left- and right-hand side operands, and x is a scalar. The operands are nested instances of **Ins**. The operator is an enum-like variable, with four possible values $\{ADD, SUB, MUL, PMUL\}$. The scalar member holds a ciphertext or a plaintext. An instance ins of **Ins** is created using either $Ins(scalar)$, or $Ins(op, ins1, ins2)$. An instruction to compute the addition of two encrypted values $[\![x]\!]$ and $[\![y]\!]$ thus looks like as e.g.: $Ins(ADD, Ins([\![x]\!]), Ins([\![y]\!]))$. At the start of the protocol, *Bob* must collect *Alice*'s encrypted inputs, and hard-wire them into the algorithm. For an example, see Appendix 1.

The core of the setup is the recursive procedure $binOp$, defined in Figure 6, which recursively computes an instruction including any nested instructions. The $binOp$ return value has the same structure as that of *BetterTimes*, but the attestation in the first part of the return value is now an accumulated value over all nested instructions.

The main function, wrapping all functionality, is the $evaluate$ procedure, see Figure 6. It takes as parameter an algorithm, which is modeled using an instruction

Proc. $binOp(ins)$:
if $isScalar(ins)$ **then** :
    return $([\![0]\!], ins)$;
else :
    $(a_1, x) \leftarrow binOp(ins[1])$;
    $(a_2, y) \leftarrow binOp(ins[2])$;
    switch$(ins[0])$:
        case ADD :
            return $(a_1 \oplus a_2, x \oplus y)$;
        case SUB :
            return $(a_1 \oplus a_2, x \ominus y)$;
        case $PMUL$:
            return $(a_1 \oplus a_2, x \odot y)$;
        case MUL :
            $(a_3, z) \leftarrow BetterTimes(x, y)$;
            return $(a_1 \oplus a_2 \oplus a_3, z)$;

Proc. $evaluate(alg)$:
$(a, result) \leftarrow binOp(alg)$;
return $result \oplus a$;

Fig. 6. The procedures to evaluate recursive instructions.

with nested instructions. Evaluate adds the attestation values and the result of the instructions, creating the final result – which is the output of g if and only if *Alice* is honest. For a visualization of messages exchanged and actions taken by each principal, see Appendix 2.

4 Security guarantees

The goal of this section is to show that the result of $evaluate$ as defined above is secure in the malicious adversary model for *Alice* (as depicted in Figure 2), following standard SMC security definitions. We have already introduced the fundamental notion of computational indistinguishability in Sect. 2.

Malicious adversary Recall that a malicious *Alice* in possession of the private key can attack the privacy of the inputs of *Bob* by deviating from the original protocol (as discussed in Section 1.1 for a proximity calculation protocol). Intuitively, a malicious *Alice* will deviate from the protocol every time it fails to answer to the outsourced multiplication with the expected values z' and a' as defined in Figure 5. A deviation would be for example failing to multiply x' with y', in order to change the intended jointly computed arithmetic formula. Formally, we set out to prove the following

theorem, which is an instance of the general definition of [28] where the concrete SMC protocol π will depend on the arithmetic formula g to be jointly computed. In the following indistinguishability will be established with respect to the size p of the field $\mathbb{F}_p$ (p is thus the security parameter).

Theorem 1. *For a fixed but arbitrary arithmetic formula $g(\overrightarrow{x}, \overrightarrow{y})$ represented by a recursive instruction $\iota \in$ **Ins**, for every adversary $\mathcal{A}$ against the protocol π resulting from $evaluate(\iota)$, there exist a simulator $\mathcal{S}$ such that:*

$$\{\mathit{IDEAL}_{g,\mathcal{S}(s)}(\overrightarrow{x}, \overrightarrow{y})\} \overset{c}{\equiv} \{\mathit{REAL}_{\pi,\mathcal{A}(s)}(\overrightarrow{x}, \overrightarrow{y})\}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability of distributions.

Here the IDEAL function gives the distribution of the output of a simulator $\mathcal{S}$ that interacts with an idealized implementation of the functionality g on behalf of *Alice*, where both parties give their inputs to a trusted third party that computes g and gives it back to $\mathcal{S}$. Recall that in our setting *Bob* receives no output from the ideal functionality. Therefore, it does not make sense for the adversary to abort the protocol. Also, this means that fairness guarantees for *Bob* are out of scope, so we do not account for abortions of the protocol by the simulator.

On the other hand REAL stands for the distribution of the output of a real adversary $\mathcal{A}$ against concrete executions of the protocol π. The parameter s stands for extra information known to the attacker, in this case we assume that the adversary knows the abstract arithmetic formula g and therefore knows how many multiplications it contains.

Before proceeding with the proof, we introduce the following Lemma.

Lemma 1. *In the outsourced multiplication protocol BetterTimes the attestation value* $\mathbf{a}$ *is equal to 0 if the protocol is followed, and is indistinguishable from a randomly distributed non-zero element otherwise.*

Proof. First recall the calculations from Figure 5:

$$[\![a]\!] \leftarrow ([\![a']\!] \ominus [\![z']\!] \odot c_m \ominus [\![y']\!] \odot (c_a \cdot c_m)) \odot \rho; \tag{11}$$

$$[\![z]\!] \leftarrow [\![z']\!] \ominus ([\![x']\!] \odot b_y \oplus [\![y']\!] \odot b_x \oplus [\![b_x \cdot b_y]\!]) \tag{12}$$

Which in the plaintexts corresponds to:

$$a = (a' - (z' + y' \cdot c_a) \cdot c_m) \cdot \rho \tag{13}$$

$$z = z' - (x' \cdot b_y + y' \cdot b_x + b_x \cdot b_y) \tag{14}$$

where a' and z' are produced by *Alice*. It is easy to see that if a' and z' are computed following the protocol, then $a = 0$ by construction.

To see that if *Alice* does not comply with the protocol then a is a randomly distributed non-zero element with very high probability, first note that there are three cases for non-compliance, either $z' \neq x' \cdot y'$, $a' \neq y' \cdot c$ or both. In any case of non-compliance, the goal of *Alice* is to construct a' and z' such that:

$$a' - (z' + y' \cdot c_a) \cdot c_m = 0$$

since otherwise by construction a will be random. Then it must hold:

$$a' = (z' + y' \cdot c_a) \cdot c_m$$

Note that given $(x' + c_a) \cdot c_m$ (which is known by *Alice*), the probability of guessing c_m is at most $\epsilon = \frac{1}{2^p}$ where p is the size of the field, since multiplication is a random permutation and c_a is unknown and uniformly distributed.

Now by contradiction, lets assume that the probability of *Alice* of computing $a' = (z' + y' \cdot c_a) \cdot c_m$ with $z' \neq x' \cdot y'$ is bigger than ϵ. If this holds, then she can also compute:

$$\alpha = a' - (x' + c_a) \cdot c_m \cdot y' = (z' - x' \cdot y') \cdot c_m$$

But then she could also compute $c_m = \alpha(z' - x' \cdot y')^{-1}$ with probability bigger than ϵ, since by hypothesis $z' \neq x' \cdot y'$ and thus $(z' - x' \cdot y') \in \mathbb{F}_p^*$ is invertible, which contradicts the fact that the probability of guessing c_m is smaller than ϵ. □

Now, for the proof of Theorem 1:

Proof (Theorem 1). Without loss of generality, we assume that $\iota \in$ **Ins** has m instructions of type MUL. We will distinguish two cases.

$\mathcal{A}$ follows the protocol It is easy to see that all m intermediate messages sent from *Bob* appear uniformly random to *Alice* (and independent) due to the fact that they are all of the type $r_i = (x', y', c)$ where each value is blinded. In the case when $\mathcal{A}$ complies with the protocol, the last message contains the correct output g, since *Bob* is an honest party. This implies that the output of $\mathcal{A}$ depends exclusively on $r_0, \ldots, r_m$ uniformly distributed triples and $g(\overrightarrow{x}, \overrightarrow{y})$, so we can simulate an adversary as:

$$\mathcal{S} := \mathcal{A}(r_0, \ldots, r_m, g(\overrightarrow{x}, \overrightarrow{y}))$$

$\mathcal{A}$ does not follow the protocol Note that independently of the cheating strategy of $\mathcal{A}$, all m intermediate messages sent from *Bob* appear uniformly random to $\mathcal{A}$ since the blinding is done by *Bob* locally with randomization independent from $\mathcal{A}$'s inputs. Now, as a consequence of Lemma 1, if $\mathcal{A}$ does not follow the protocol for at least one of the outsourced multiplications, the final message will be blinded by the accumulated attestation value, which is indistinguishable from random. Therefore, the last message will contained the encryption of a random value, denoted r_{m+1}. Therefore we can simulate this in the ideal model as:

$$\mathcal{S} := \mathcal{A}(r_0, \ldots, r_m, r_{m+1})$$

for pairwise independent and random variables r_i. □

The requirements on fields Additively homomorphic schemes are commonly defined over groups where when multiplying a non zero element γ with a uniformly chosen ρ, the result is not necessarily uniformly distributed, thus potentially affecting the blinding of $g(x, y)$. For instance, in groups such as $\mathbb{Z}_n$ for composite $n = p \cdot q$ (as used by the Paillier [31] encryption scheme) when multiplying a non invertible element with random ρ, the result stays in the subgroup of non-invertible elements. In that setting is thus possible to show a counterexample to the theorem above, which motivates our restriction to constructions over fields.

5 Evaluation

The approach has been implemented in Python using the GMP [13] arithmetic library. The implementation as been benchmarked to show the impact of using our approach compared to the more common approach of naive outsourced multiplications. In the naive approach, *Alice* is honest-but-curious, and the operands are therefore only blinded. For this implementation, the DGK [9] cryptosystem was used.

Table 1 shows time in milliseconds for different sizes of plaintexts and keys for the two cases when outsourced multiplication is performed using BetterTimes, or naively. The difference between the two approaches is a small factor of about 1.5 for both key sizes, though slightly smaller for the larger keys. The factor is only marginally increasing as the plaintext space grows from 2^2 to 2^{24}.

The benchmarked time shows only the processing time for each multiplication, the communication overhead is exactly twice for our approach as compared to the naive solution.

Table 1. Benchmarks for outsourced multiplication

Plaintext space	Time (in milliseconds)					
	1024 bits			2048 bits		
	This approach	Naive approach	Extra work	This approach	Naive approach	Extra work
2^2	6.286	4.016	56.52%	29.686	19.458	52.56%
2^8	6.400	4.017	59.32%	30.052	19.484	54.24%
2^{16}	6.432	4.148	55.06%	30.188	19.574	54.22%
2^{24}	6.538	4.100	59.46%	30.578	19.801	54.43%

6 Related Work

There are three current approaches to compute an arbitrary formula in the two-party setting in the presence of malicious adversaries, Fully Homomorphic Encryption, Enhanced Garbled Circuits and Zero-knowledge proofs.

FHE is by far the most inefficient approach, and its use is often considered not feasible due to the heavy resource consumption. We do not consider FHE a viable alternative to additively homomorphic encryption for practical applications. Garbled Circuits is an excellent tool for boolean circuits, but has been found to not perform as well for arithmetic circuits as approaches built on homomorphic encryption. Zero-knowledge proofs could be used instead of the proposed approach, but at the cost of more computations and/or round trips.

Zero-knowledge Proofs The technique which most resembles *BetterTimes* is that of *Zero-Knowledge* (ZK) proofs. Any statement in NP can be proven using generic, though inefficient, ZK (Goldreich et al. [18]). However, to the best of our knowledge, there is no ad-hoc proof for correct multiplications that directly applies to the setting of additively homomorphic encryption without significantly more overhead than the proposed approach, by e.g. introducing more round trips.

Some protocols in the literature can be used efficiently for proving correct multiplications, with only one additional round trip. One such is the Chaum-Pedersen protocol [6], which however is not trivially applicable to an arbitrary encryption scheme. Another interesting solution was introduced by Damgård and Jurik [10], but which is constructed specifically for the Damgård-Jurik cryptosystem.

Secure Multi-party Computations There are two promising solutions for private remote computations for the two-party case: Homomorphic Encryption and Garbled Circuits. Through recent research they are both near practical applicability (see [24,

20, 25] and [5, 20, 15]). However, which of the two approaches to choose is typically application-dependent [26, 23]. Our approach brings state-of-the-art SMC solutions based on additively homomorphic cryptographic systems forward by protecting against malicious adversaries when outsourcing multiplications, while remaining strongly competitive to the efficient though less secure approaches which currently are popular examples.

There are several works that combine the use of an additively homomorphic scheme with secret sharing, to compute multiplications securely using threshold encryption. This line of work stems from the SMC schemes developed by Cramer et al. [8]. Note that such approaches are secure only against malicious *minorities*, and are not directly applicable in scenarios with only two parties.

To compare against GC-solutions which can compute arbitrary formulas, some experiments using FastGC, a Garbled Circuit framework by Huang et al. [20] were conducted. Any arithmetic circuit can be expressed as a binary circuit, and vice versa[14]. In this framework for arbitrary computations, integer multiplication of 24-bit numbers needed 332 ms to finish, approximately 5078% slower than Better-Times. Note however that FastGC is only secure in the honest-but-curious model, and thus not as secure as the approach presented in this paper. Further work exists in the direction of efficiently providing security against malicious adversaries by the authors of FastGC [21], however where one bit of the input is leaked. Moreover, work on optimizing garbled-circuits in the honest-but-curious model also exists, e.g. recently [29], but so far without enough speedup that it can compare to additively homomorphic encryption for privately computing arithmetic formulas.

7 Conclusions

We have presented a protocol for outsourcing multiplications and have shown how to use it construct a system for computation of arbitrary arithmetic formulas with strong privacy guarantees. We have shown that the construction is secure in the malicious adversary model and that the overhead of using the approach is a small constant factor.

The need for such a protocol is justified by the format attacks we have unveiled in known protocols, and presented a concrete exploit targeting [38] where we can alter the format of a message and gain more than the intended amount of location information. We have made a case for using a more realistic attacker model and identified examples from the literature which are vulnerable to this stronger at-

tacker, while also showing how to amend such vulnerabilities. Moreover, we make our implementation fully available to the community.

As future work we plan to investigate the non-trivial task of applying closely related primitives (such as Zero-Knowledge constructions [6] and Threshold Encryption [8]) to achieve the same security guarantees, and benchmark those solutions to compare them to *BetterTimes*.

Acknowledgments Thanks are due to Allen Au for the useful comments. This work was funded by the European Community under the ProSecuToR project and the Swedish research agencies SSF and VR.

References

1. 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.
2. 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.
3. D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In S. Jajodia and J. López, editors, *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2008.
4. P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In R. Dingledine and P. Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.
5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011.
6. D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.
7. D. Coldewey. "Girls Around Me" Creeper App Just Might Get People To Pay Attention To Privacy Settings. TechCrunch, March 2012.
8. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques,*

Innsbruck, Austria, May 6-10, 2001, Proceeding, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, 2001.

9. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer, 2007.
10. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In K. Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.
11. R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.
12. 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.
13. Free Software Foundation. The gnu multiple precision arithmetic library. `http://gmplib.org/`, 1991-2013.
14. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.
15. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.
16. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.
17. O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
18. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, July 1991.
19. P. A. Hallgren, M. Ochoa, and A. Sabelfeld. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In A. A. Ghorbani, V. Torra, H. Hisil, A. Miri, A. Koltuksuz, J. Zhang, M. Sensoy, J. García-Alfaro, and I. Zincir, editors, *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, pages 1–6. IEEE, 2015.
20. Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*. USENIX Association, 2011.
21. Y. Huang, J. Katz, and D. Evans. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 272–284. IEEE Computer Society, 2012.

22. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. From dust to dawn: Practically efficient two-party secure function evaluation protocols and their modular design. *IACR Cryptology ePrint Archive*, 2010:79, 2010.
23. 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.
24. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.
25. B. Kreuter, A. Shelat, and C. Shen. Billion-gate secure computation with malicious adversaries. In T. Kohno, editor, *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 285–300. USENIX Association, 2012.
26. 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.
27. 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 *MobiHoc*, pages 43–52, 2014.
28. Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008:197, 2008.
29. C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. Oblivm: A programming framework for secure computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 359–376. IEEE Computer Society, 2015.
30. 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.
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. I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis. Where's wally?: Precise user discovery attacks in location proximity services. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 817–828. ACM, 2015.
33. 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.

34. 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.
35. A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.
36. M. Veytsman. How I was able to track the location of any Tinder user, February 2014. Web resource: http://blog.includesecurity.com/.
37. M. Wachs, M. Schanzenbach, and C. Grothoff. On the feasibility of a censorship resistant decentralized name system. In J. L. Danger, M. Debbabi, J. Marion, J. García-Alfaro, and A. N. Zincir-Heywood, editors, *Foundations and Practice of Security - 6th International Symposium, FPS 2013, La Rochelle, France, October 21-22, 2013, Revised Selected Papers*, volume 8352 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2013.
38. 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 A concrete instantiation to secure Hallgren et al.

To make the protocol from Hallgren et al. (and other afflicted protocols) secure against format attacks from *Alice*, the distance can be computed directly on the coordinates instead of using several correlated values. The secured algorithm could be modeled as follows:

$$
\begin{aligned}
&Ins(SUB,\\
&\quad Ins(ADD,\\
&\qquad Ins(ADD, Ins(MUL, Ins([\![x_A]\!]), Ins([\![x_A]\!])), Ins([\![x_B^2]\!])),\\
&\qquad Ins(ADD, Ins(MUL, Ins([\![y_A]\!]), Ins([\![y_A]\!])), Ins([\![y_B^2]\!]))\\
&\quad),\\
&\quad Ins(ADD,\\
&\qquad Ins(PMUL, Ins([\![x_A]\!]), Ins(2 \cdot x_B)),\\
&\qquad Ins(PMUL, Ins([\![y_A]\!]), Ins(2 \cdot y_B))\\
&\quad),\\
&)
\end{aligned}
$$

2 Visualization of privacy-preserving arithmetic formula

Figure 7 depicts the system for privacy-preserving arithmetic formulas presented in this paper, during an execution where *Alice* is honest. *Alice* is the initiating party, and starts by sending her inputs to *Bob*. *Bob* then hardwires both is and *Alice*'s inputs into a instruction of nested operations, forming a tree like in Figure 3. Depending on g, *Bob* computes any local operations and executes BetterTimes as necessary, with as many iterations as necessary. Finally, he computes the ciphertext $[\![result]\!]$. Since *Alice* by assumption is honest, $[\![result]\!]$ will hold the output of g (and would hold the encryption of a random element in $\mathbb{F}_p$ if *Alice* was dishonest). BetterTimes is simplified here, for a more complete visualization see Figure 4.

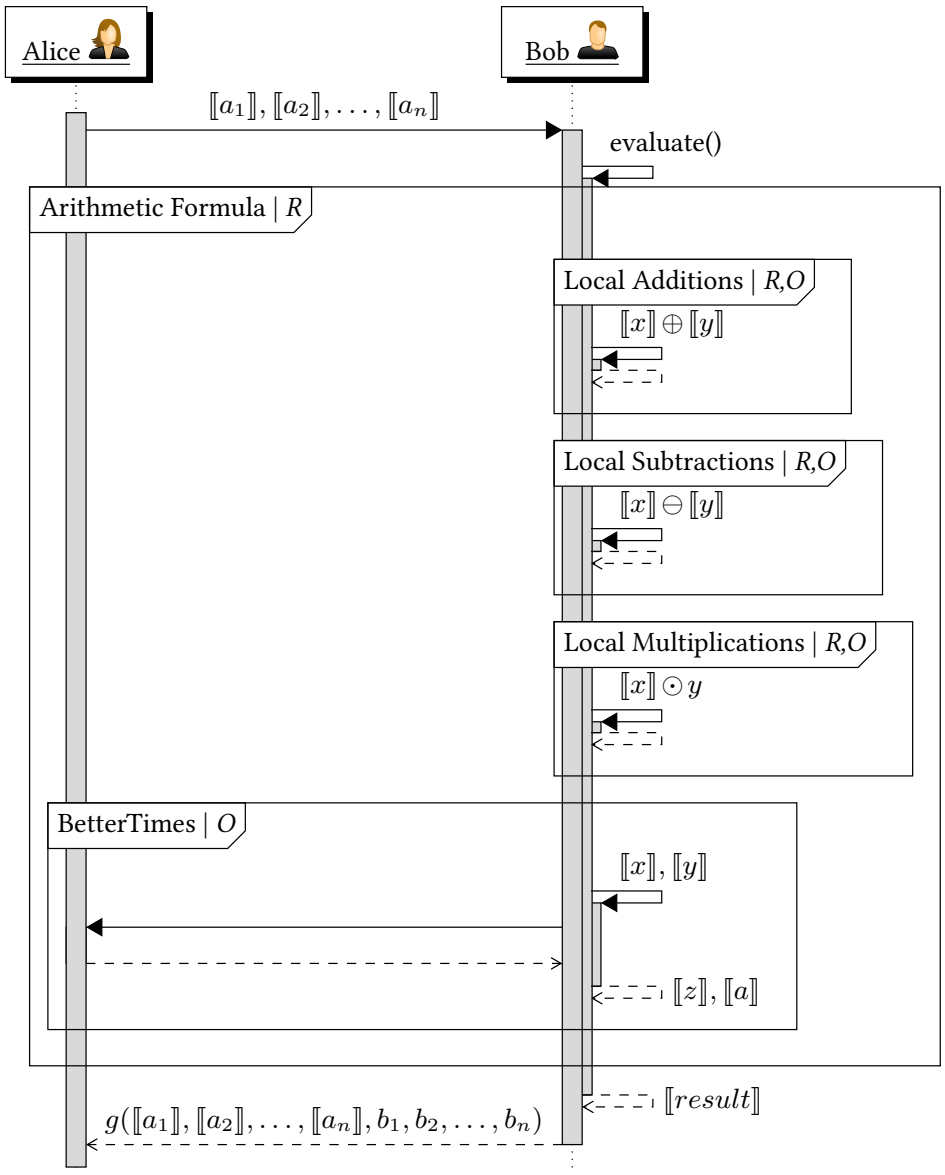

Fig. 7. Visualization of actions by each principal, where *R* and *O* means repeatable and optional, respectively.

Paper **three**

MaxPace

Speed-Constrained Location Queries

Per A. Hallgren, Martìn Ochoa and Andrei Sabelfeld

With the increasing proliferation of mobile devices, location-based services enjoy increasing popularity. At the same time, this raises concerns regarding location privacy, as seen in many publicized cases when user location is illegitimately tracked both by malicious users and by invasive service providers. This paper is focused on privacy for the location proximity problem, with the goal of revealing the proximity of a user without disclosing any other data about the user's location. A key challenge is attacks by multiple requests, when a malicious user requests proximity to a victim from multiple locations in order to position the user by trilateration. To mitigate these concerns we develop MaxPace, a general policy framework to restrict proximity queries based on the speed of the requester. MaxPace boosts the privacy guarantees, which is demonstrated by comparative bounds on how the knowledge about the users' location changes over time. MaxPace applies to both a centralized setting, where the server can enforce the policy on the actual locations, and a decentralized setting, dispensing with the need to reveal user locations to the service provider. The former has already found a way into practical location-based services. For the latter, we develop a secure multi-party computation protocol that incorporates the speed constraints in its design. We formally establish the protocol's privacy guarantees and benchmark our prototype implementation to demonstrate the protocol's practical feasibility.

Published in the 2016 IEEE Conference on Communications and Network Security
CNS 2016, Philadelphia, PA, USA, October 17-19, 2016

1 Introduction

The increasing proliferation of mobile devices drives tremendous developments in the area of mobile computing. Mobile Internet usage already dominates over desktop both by the number of users [13] and time spent [5]. As part of these developments, location-based services enjoy increasing popularity, enabling location-based features such as finding nearby points of interest or discovering friends in proximity.

At the same time, services that involve location information raise increasing privacy concerns. These concerns apply to both protecting the privacy with respect to other users and with respect to service providers. There are publicized cases of both scenarios in practice. For the former scenario, the smartphone app *"Girls around me"* allowed users to find other users (profiled as female) who recently had checked in on Foursquare [2]. Deemed as a serious privacy violation, the app has since been banned from the Foursquare API and removed from the app store. For the latter scenario, the smartphone app Uber, connecting passengers with private drivers, has been the subject of much privacy debate. Uber and its employees have been allegedly involved in privacy-violating activities from stalking journalists and VIPs to tracking one-night stands [1].

These privacy concerns call for developing privacy-aware location based services [12, 27]. Accordingly, our goal is striving for rigorous guarantees for the protocols that underlie practical location-based services.

The following motivates our approach. We start with unconstrained services that freely share user location and gradually illustrate the protection measures that need to be in place to protect against location privacy attacks. In the following paragraphs, let us focus on a scenario where a malicious user attempts to leverage a location-based service to attack location privacy of another user by looking at four techniques that have been applied in practice. Subsequently, we discuss the service-attacks-user scenario in a decentralized setting.

From positions to distances Directly revealing locations may violate privacy. For example, as of May 2015, Facebook Messenger defaulted to sending user location tags with all messages, which was exploited by a "stalking" Chrome extension [11]. Since then, Facebook has deactivated location sharing from the desktop web page.

From distances to approximate distances To aid privacy, the next step is to reveal distances instead of locations. For example, the dating app Tinder revealed distances to other users. It is straightforward to bypass this protection and calculate the location. Indeed, an illustrative attack on Tinder has been detailed by Include Security [28],

revealing exact position of any user. At the core of this and other practical attacks is the technique of *trilateration*.

Trilateration makes use of multiple requests, where each results in learning that the user is located on a circle that is centered in the requester's position. Trilateration derives the user location as the intersection point of three circles, precisely pinpointing the user, as illustrated in Figure 1.

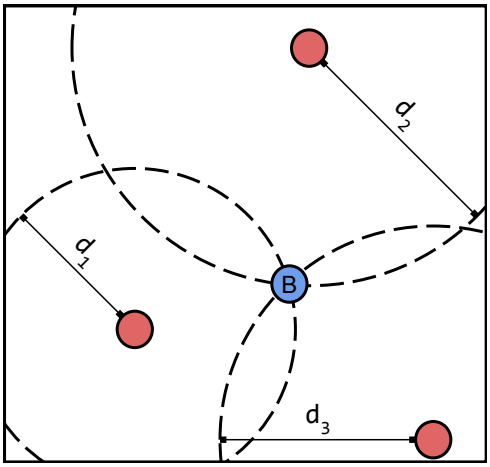

Fig. 1: Trilateration attack

From approximate distances to proximity To mitigate trilateration attacks, some services have started using approximate distance. As an example, Facebook's Nearby Friends feature rounds the distance information. However, this mitigation can be easily bypassed. Recent research systematizes these attacks and identifies a number of location-based services where it is possible to reveal the user location even if distances are approximated/obfuscated [21, 14, 20].

From proximity to speed-constrained proximity Next notch up for privacy is not to reveal the distance but to reveal proximity to the other user. This drastically reduces information about the location: it is only one bit per request. Still, proximity can be viewed as coarse-grained approximation, and attacks to pinpoint user location are still possible.

Instead of trilateration as in the case of distance-based attacks, the attacker in the proximity-based setting essentially (i) solves the *DiskCoverage* problem [20] by covering the plane with non-overlapping circles until getting a positive proximity response, and then (ii) solves the *DiskSearch* problem [20] to get the exact location by aiming to divide the constraining discs by half with each request.

In this paper, we explore in depth the effect of constraining the speed of the requester on the effect of the user discovery attacks. The key idea is that it is unrealistic for honest users to drastically change their location between subsequent requests. Hence, we aim at a policy that impedes the attacker without hampering practical usage of proximity protocols.

The *DiskCoverage* problem is in the focus of this work, as speed-constraining techniques as applied in practice provide little protection against an attacker on the *DiskSearch* problem.

MaxPace: speed-constrained proximity We develop MaxPace, a general policy framework to restrict proximity queries based on the speed of the requester. The effect of speed constraints is illustrated in Figure 2. Each query corresponds to a disk. Large disk overlaps with speed-constrained queries means that the attacker learns little information from each query compared to the unrestricted attacker, and thus needs to issue more requests to learn the victims location.

MaxPace applies to both a centralized setting, where the server can enforce the policy on the actual locations, and a decentralized setting, dispensing with the need to trust the service provider. In the centralized setting, we are encouraged by the recent changes in the policies of the popular centralized location-based services Facebook, Swarm, and Tinder [20] to incorporate forms of speed-based constraints. Our study is intended to provide rigorous analysis and understanding of guarantees provided by this type of constraints.

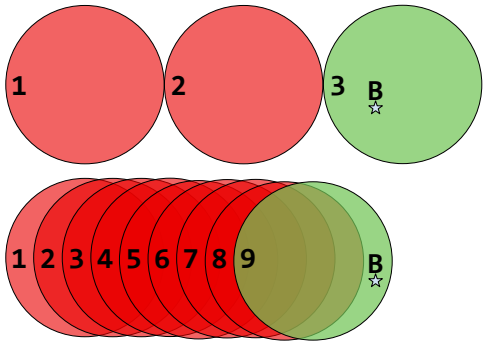

Fig. 2: Different protocols

In the decentralized setting, we develop a secure multi-party computation protocol. This offers a contribution beyond the state of the art. The state-of-the-art protocols often lack formal privacy guarantees [29, 25, 24, 7]. Further, when there are formal guarantees a dominant assumption in the most recent literature on securing location proximity [29, 25, 24, 7, 17, 19, 23, 10] is the assumption of a single run. In contrast, our approach does not impose such an assumption, allowing us to reason about multi-run attacks.

Though this paper tackles malicious attackers, colluding attackers have been out of reach for the state-of-the-art work both in the single-run [29, 25, 24, 7, 17, 19, 23, 10] and also in the multi-run [20] setting. This justifies restricting the scope to non-colluding attackers in this paper. Also note that an attacker may be prevented from using multiple devices and/or multiple accounts by using authentication on top of the proposed solution.

The paper offers the following contributions. Section 2 presents MaxPace, the speed-constrained proximity disclosure policy. Section 3 shows the bounds provided by MaxPace and compares them to the classical unrestricted attacker. Section 4 shows how MaxPace can be enforced without a trusted third party, granting privacy of the locations of both involved parties. Our enforcement is based on a novel se-

cure multi-party computation protocol. We formalize the privacy guarantees of the protocol in Section 5. Section 6 presents benchmarks of our implementation of the protocol, demonstrating its practical feasibility.

2 The MaxPace Proximity Disclosure Policy

We introduce MaxPace, a policy for location proximity disclosure which limits the speed of a querying principal. The attacker is moving freely at arbitrary speed, but the victim is at a fixed position. The case when the victim is moving is an interesting field of study, but which poses challenges [3], left for future work.

The intuition is to force the attacker to behave as a benign user. In a normal setting, users assume any protocol participant behaves according to real-world physical constraints; other participants are e.g. walking, riding a bike, driving a car. The constraints imposed by MaxPace give less freedom to attackers during each query, causing them to learn less about the victim's location, and thus impose an additional effort to locate the victim. The exact benefits gained through MaxPace are discussed in detail in Section 3, as compared to an unconstrained attacker on a proximity protocol.

When a principal wishes to know the proximity of another principal they issue a *location query*, as defined in Definition 1. The queried principal is henceforth referred to as Bob and the querying principal as Alice. That is, Alice asks Bob whether or not they are close. Alice may be malicious, and try to locate Bob by repeated querying.

Definition 1 (Location query). *Let P be the set of possible coordinates (x, y) representing the position of a principal in the plane. A location query q is a tuple $(p, t) \in P \times \mathbb{N}$ where p is the position of the querying principal, and t the timestamp of when the query was issued.*

When *Bob* receives a query from *Alice*, he considers her speed. If *Alice* is respecting the maximum speed h set by the policy, she receives the correct result. However, if she is moving too fast by quickly spoofing coordinates that are far apart she instead receives an unusable $\perp$*-value* as defined in Definition 2. Though MaxPace does not prevent usage artificial locations, it limits the effectiveness of such attacks.

Definition 2 ($\perp$-values). *A value containing no useful information (such as an error message, a* `null`*-value or a freshly sampled uniformly random value) is called $\perp$.*

If a query respects the speed threshold, *Bob* computes the proximity, indicating *only* whether the principals are within a distance r. Distance between the points of two queries is calculated as described in Definition 3.

Definition 3 (Query distance). *The Euclidean distance between two positions p_1 and p_2 is given by $dist(p_1, p_2)$.*

This paper uses a common definition of proximity [10, 20], defining proximity between two positions p_1 and p_2 as:

$$inProx(p_1, p_2, r) = \begin{cases} True & \text{if } dist(p_1, p_2) < r \\ False & \text{otherwise} \end{cases}$$

If *Alice* moves at a speed allowed by *Bob*, she may query for location proximity at any frequency. Once she surpasses this speed, any future requests she initiates yield $\perp$. This is further formalized in Definition 4.

Definition 4 (MaxPace). *The responses $L = \{l_1, l_2, ..., l_m\}$ to a series of location queries $Q = \{q_1, q_2, ..., q_m\}$ from* Alice *to* Bob *respect MaxPace if and only if:*

$$l_i = \begin{cases} \perp & \textit{if } \exists_{j<i} : \frac{dist(p_j, p_{j+1})}{time(q_j) - time(q_{j+1})} > h \\ inProx(p_i, p_B, r) & \textit{otherwise} \end{cases}$$

where $q_i = (p_i, t_i)$ and the position of Bob *is p_B.*

Definition 5 (Query Time). *Given a location query $q = (p, t)$, let $time(q) = t$.*

The two parameters r and h can be considered public and mutually agreed upon by *Alice* and *Bob* prior to running the protocol, e.g. as a part of the key exchange.

If a principal is detected as using a too high speed, they are seen as malicious and indefinitely prevented from learning further location information. However, as an effect of imprecise GPS positioning (e.g. by being under ground), or after having used a means of transportation not considered by the application (e.g. an airplane or high-speed train), the positions reported by a benign user may indicate that the user is traveling at higher speeds than allowed. A benign user can potentially be seen as malicious without having tried to attack the user. For applications in practice, some speed violations might need to be allowed in some cases. To reduce the period of time a benign principal is classified as malicious, a principal who previously has acted as if malicious can be forgiven and be allowed to query for location information again.

To forgive a principal, there are many viable strategies to reset the protocol. The simplest is arguably to reset after a fixed amount of time. An interesting approach could be to reset if the blocked principal returns to the point where it first broke the speed limit. This would capture the case when a user takes a flight abroad, and allow them to resume querying other principals after returning from the trip. If a user who has reported speeds in excess of those allowed is forgiven, the effort of a

malicious attacker is lowered. In the worst case, the attacker knows exactly when a reset occurs and may then move at arbitrary speed between two queries. To what extent the attacker effort is affected is discussed in Section 3.4.

3 Bounds on Attacker Effort

This section defines bounds for the attacker effort to locate the victim using a proximity disclosure protocol, both for the normal case and when applying the MaxPace policy. Disclosing only proximity forces the attacker to search a large portion of the plane to locate them. The bounds calculated here gives a measure of quickly the attacker can search the plane. The analysis considers the space as finite but of arbitrary size in discrete Euclidean coordinates, and the victim's position is a uniformly distributed variable. In this setting, for a fixed time period, the attacker's chance of finding the victim is negligible. Further, any holes left unexplored by the search strategy are of negligible size relative to the remaining area. Polakis et al. [20] present definitions for scenarios similar to the one considered in this paper, where an attacker is trying to locate a user in a finite section of the discrete Euclidean plane. Their terminology is reused in the following for clarity.

As mentioned previously, this work tackles an attacker who is trying to find which disk the user resides in, which is called the *DiskCoverage* problem. More precisely, *DiskCoverage* is defined in Definition 6. Note that the definition here is slightly different from the definition used by Polakis et al., even though the goal of the attacker is equivalent. In the original definition the attacker wants to minimize the time to completely cover a fixed space, while here the attacker attempts, but does not succeed, in solving the *DiskCoverage* problem in a fixed time and thus focuses on maximizing progress.

Definition 6 (*DiskCoverage*). *The* DiskCoverage *problem is to find a set S containing the possible coordinates of the victim, such that $|S| \leq r^2\pi$.*

When calculating the effort for the attacker solve the *DiskCoverage* problem, the progress made with each individual query is needed. Herein, we say that the attacker *learns* an amount of knowledge from a location query, see Definition 7.

Definition 7 (Attacker knowledge per query). *The neighborhood of radius r of a location query $q = (p, t)$ is a set $cov_r(q)$, s.t. $\forall p_i \in cov_r(q) : inProx(p_i, p, r)$. From the response of a single location query, the attacker learns if the victim's position $p_b \in cov_r(q)$. We thus call $cov_r(q)$ the attacker knowledge for query q.*

The *knowledge* of the attacker thus corresponds to the area (or set of points) which is within the proximity of any issued proximity request, and for a set Q of queries, the accumulated knowledge is the union $\bigcup_{q \in Q} cov_r(q)$. For the attackers on both DecentMP and a plain protocol, an optimal attacker is assumed. For the plain protocol, the precise knowledge gained by the attacker can be calculated, while for DecentMP an upper bound on the knowledge is presented. Let the accumulated knowledge of an attacker that is not limited by MaxPace be called a_{plain} and the knowledge of an attacker limited by MaxPace be called $a_{MaxPace}$.

3.1 Knowledge of Unconstrained Attacker

Clearly any query by the unconstrained attacker which overlaps with previously covered areas (such that $cov_r(q_i) \cap cov_r(q_j) \neq \emptyset$) is a bad strategy, as it contains less knowledge (fewer points) than non-overlapping queries. The optimal attacker thus covers an area of $r^2\pi$ with each query. During T time units, the plain attacker does T/t_p queries at distinct locations, where t_p is the minimum time required to receive a response from a location query. This yields $a_{plain} = \pi r^2 \frac{T}{t_p}$.

3.2 Bounds for MaxPace

Now for an attacker constrained by MaxPace, for which an upper bound is given. Comparing the upper bound of an attacker on MaxPace to the attacker on the plain policy gives the *minimum* advantage MaxPace has over a plain policy.

Unlike the unrestricted querying policy, the optimal attacker on MaxPace is forced to query with overlapping coverage – otherwise they are traveling faster than the limit h and learn nothing at all. Note that the attacker may choose to query with a distance of $2 \cdot r$ with a large time interval to not have an overlap but the attacker gains more knowledge when querying as often as possible, as shown through Theorem 1.

Theorem 1 (Optimal attackers on MaxPace query as often as possible). *Given two queries $q_s = (p_s, t_s)$ and $q_e = (p_e, t_e)$ which do not violate the MaxPace policy, and where $p_s \neq p_e$. If it is possible to define a third query $q_i = (p_i, t_i)$ such that for t_i and p_i $(t_s < t_i < t_e) \vee (p_s \neq p_i \neq p_e)$ holds, and q_s, q_i, q_e comply with MaxPace, then issuing the three queries q_s, q_i, q_e yields more information than issuing q_s, q_e.*

Proof. By contradiction, assume that $cov_r(q_e) \cup cov_r(q_s)$ is equal to $cov_r(q_e) \cup cov_r(q_i) \cup cov_r(q_s)$. This implies that either $cov_r(q_s) = cov_r(q_i)$ or $cov_r(q_e) = cov_r(q_i)$, which in turn implies $p_i = p_e \vee p_i = p_s$ ↯.

From Theorem 1, the attacker sends a location query as soon as the policy allows them after moving one distance unit, thus waiting at most $s = 1/h$ time units between each query. The coverage for each query is calculated as the area of a circle of radius r, subtracting the area of the intersection with the previous query. How to calculate the area of circle intersections is given in [18]. The knowledge gained by an adversary for each query after the first one is given in Equation (1).

$$\pi r^2 - \left(2r^2 cos^{-1}\left(\frac{1}{2r}\right) - \frac{1}{2}\sqrt{4r^2 - 1}\right) \tag{1}$$

For a more concise bound, the formula in simplifications are made to over-approximate Equation (1). This means an under-approximation of Equation (2) and over-approximation of Equation (3).

$$-2r^2 cos^{-1}\left(\frac{1}{2r}\right) \qquad (2) \qquad\qquad \frac{1}{2}\sqrt{4r^2 - 1} \qquad (3)$$

Note that $\lim_{x\to 0} cos^{-1}(x) = \frac{\pi}{2}$ (as $r \geq 1$). Thus, an under-approximation of Equation (2) is $-2r^2\frac{\pi}{2}$. Equation (3) can be simplified by approximating $\sqrt{4r^2 - 1}$ to $\sqrt{4r^2}$. The concise over-approximation of Equation (1) is given in Equation (4).

$$\pi r^2 - \left(2r^2\frac{\pi}{2} - \frac{1}{2}2r\right) = \pi r^2 - 2r^2\frac{\pi}{2} + \frac{1}{2}2r = r \tag{4}$$

The attacker is able to perform a total of T/s queries. Including the first query, which covers an area of $r^2\pi$, the final area covered during *DiskCoverage* is given by:

$$a_{MaxPace} = r\left(\frac{T}{s} - 1\right) + r^2\pi$$

3.3 Comparisons

Table 1: Speeds in m/s and km/h for the used scenarios

Activity	Walking	Running	Cycling	Bus	Car (highway)
m/s	2	3	5	14	33
km/h	7.2	10.8	18	50.4	118.8

To evaluate the policy, the example activities of walking, running, cycling, riding a bus, and driving a car are considered, as listed in Table 1. To compare a_{plain} and $a_{MaxPace}$, the ratio $\frac{a_{plain}}{a_{MaxPace}}$ is considered. Given the above example speeds

and reasonable values of r, consider Table 2. The table shows up to 753 times less information disclosure, demonstrating the effectiveness of MaxPace in practical scenarios. The value of t_p is chosen as 200 milliseconds for the plain protocol.

Table 2: Bounds for different speed radiuses

Speed	Radius			
	10	25	50	100
Walking	78.2	194.3	384.4	752.7
Running	52.2	130.0	258.1	508.8
Cycling	31.4	78.2	155.7	308.8
Bus	11.2	28.0	55.9	111.5
Car	4.8	11.9	23.8	47.5

3.4 MaxPace with Resetting

As highlighted in Section 2, there are scenarios where MaxPace is too restrictive. In these cases, it is beneficial if the protocol can be reset. When a reset occurs, an attacker will be able to reposition themselves independently of previous queries. If during a time frame T the attacker performs e resets, the bound of the attacker knowledge is $r\left(\frac{T}{s} - e\right) + \pi r^2 e$.

Concretely, consider a person who is walking and querying a radius of 100 meters, where the protocol is reset every 15 minutes, the time unit is seconds, and where $T = 3600$ (one hour). The attacker on the plain policy covers exactly 565486677.6 square meters. MaxPace without resetting restricts coverage to at most $751315.9m^2$, and MaxPace with resetting gives a coverage of at most $876579.6m^2$. Though resetting in this case causes over 16% extra information leakage, even with resetting MaxPace yields with the given parameters 645 times less information than the plain protocol.

4 Enforcement without Trusted Third Party

As foreshadowed earlier, MaxPace can be implemented in a straightforward way using a trusted third party who stores and manages location information for all users who are utilizing the service. Any already existing service can easily deploy MaxPace as an additional privacy measure. Many applications scenarios lack a natural third

party that can be trusted, and a decentralized trust-model has obvious benefits as compared to giving location information to third parties. Services are usually not deployed in a decentralized manner without trusted parties, as for most application scenarios there are no ad-hoc solutions readily available.

This section describes how MaxPace can be enforced using a *Secure Multi-party Computations* (SMC) protocol without a trusted third party. The concrete protocol is referred to as DecentMP (short for *Decentralized MaxPace*).

4.1 Secure Arithmetics for SMC

There are a variety of primitives for implementing SMC, including garbled circuits [**?**], partial [**?**] and fully homomorphic encryption schemes [8] among others. To instantiate the MaxPace policy without third parties, we chose the BetterTimes system by Hallgren et al. [9]. This construction gives privacy guarantees against a malicious *Alice* Further, being based on additively homomorphic cryptography, it supports storing intermediate values from previous computations, a central feature in the implementation of MaxPace. Additionally, there is an open and efficient implementation of this construction that allows us to benchmark our results and discuss the applicability of our protocol in practice. For the scope of this paper, Alice holds the private key for all BetterTimes computations and is the only principal able to decrypt data. However, *Bob* is able to perform arithmetic computations using the BetterTimes system.

A BetterTimes formula is composed of a number of arithmetic instructions. The BetterTimes-instructions are recursive data structures, An instruction is either a binary operation (e.g. addition or subtraction), for which each operand is another instruction, or a scalar value. The formula is evaluated by *Bob*, and all actions taken by *Alice* are implicitly determined by any messages *Bob* sends. If *Alice* deviates from the protocol while computing a BetterTimes formula the output is a uniformly random number, and *Alice* learns only $\perp$. The guarantees of the construction are discussed further in Section 5.

EasyTimes, more readable BetterTimes-syntax This section details a subset of python, called *EasyTimes*, which directly maps into the syntax of BetterTimes. EasyTimes allows arithmetic formulas to be expressed in a more readable and concise manner than the original syntax. The full translation is detailed in Appendix 4. In short, BetterTimes-instructions are simply represented by normal addition, subtraction and multiplication operations. The operations are overloaded, and when used

a formula is constructed in the background, which later can be evaluated. Outputs are in the new syntax marked using calls to the `output()` function.

All coin tosses can be sampled from a cryptographically secure source using the $random(start, end)$ function. By convention, variables storing a ciphertext uses a prefixing "$c\_$". As in normal Python, exponentiation is written using double multiplication signs; x^y is written as `x ** y`.

Extension to BetterTimes for multiple outputs As an additional contribution of this work, an extension to BetterTimes is constructed which allows for a single formula to yield multiple outputs. The construction is presented in detail in Appendix 1. In short, the extension provides means for utilizing more than one `output()` call.

4.2 Homomorphic primitives

Below, the building blocks needed to construct DecentMP are described before proceeding to present the full protocol.

Homomorphic Distance The squared Euclidean distance (here and henceforth simply called the distance) can be computed using additively homomorphic encryption in a privacy-preserving manner [29, 6, 23, 22, 10]. As shown in [9], most approaches are only secure in the semi-honest model but can be made secure in the malicious model using BetterTimes.

The approaches above require that *Bob* holds one of the two coordinates in the clear. Listing 1 shows a short protocol in EasyTimes syntax which computes the distance between (x_1, y_1) and (x_2, y_2) without any plaintext knowledge. Computing the distance while holding a coordinate in the plain is similar, however where the last two parameters `c_x2, c_y2` are plaintexts and thus have a different type. For the scope of this paper such a method is called $ODist_{plain}$.

Listing 1: Procedure for distance computation

```
def ODist(c_x1, c_y1, c_x2, c_y2):
    c_sq1 = c_x1 * c_x1 + c_y1 * c_y1
    c_sq2 = c_x2 * c_x2 + c_y2 * c_y2
    c_cross = c_x1 * c_x2 + c_y1 * c_y2
    return c_sq1 + c_sq2 - 2 * c_cross
```

Homomorphic Comparisons There are several solutions to compute comparisons homomorphically in the literature [10, 6] by making use of bit parity. Here, a comparison method very similar to the one by Hallgren et al. is used [10].

Hallgren et al. use the fact that $(x - y) \cdot \rho$, with ρ uniformly random, yields $\perp$ if and only if x and y are not equal. Thus, to compare $x < y$, it's possible to check if $\exists i \in \{0..y-1\} : (x - i) \cdot \rho = 0$. However, where Hallgren et al. use an array of equality-checks and shuffle it to hide which slot is equal to the compared value, instead the values are multiplied here, as shown in Listing 2.

Listing 2: Procedure for computing "less than"

```
def lessThan(c_x, y):
    c_l = 1
    for i in range(0, y − 1):
        c_l = c_l * (c_x − i)
    return c_l * random(1, k)
```

Homomorphic Proximity Check To enforce the MaxPace policy, it is necessary to compute whether two points are near each other. In short, the formula consists of chaining $ODist_{plain}$ and `lessThan`, as shown in Listing 3.

Listing 3: Procedure to check the proximity of two points

```
def proximity(c_x1, c_y1, x2, y2, r):
    dist = ODist_plain(c_x1, c_y1, x2, y2)
    return lessThan(dist, r ** 2)
```

Homomorphic Speed The following shows, to the best of the authors' knowledge, the first case where speed computations are used together with additively homomorphic encryption. More precisely, *Bob* calculates whether or not the speed of *Alice* is under an allowed threshold, as shown in Listing 4.

Listing 4: Procedure to check for too fast movement

```
def speed(c_x1, c_y1, c_x2, c_y2, h, t):
    dist = ODist(c_x1, c_y1, c_x2, c_y2)
    return lessThan(dist, (h * t) ** 2)
```

The speed when moving d distance over a time t is computed as d/t. In MaxPace, the goal is to check when the speed exceeds a threshold h. Thus, the sought computation is $\frac{d}{t} \leq h$, which can be re-written (for non-negative integers) as $d \leq h \cdot t$.

4.3 DecentMP

The protocol is shown in Listing 5. The procedure is executed by *Bob*, while operations carried out by *Alice* are implicitly determined through the BetterTimes system.

Listing 5: Request handling using DecentMP

```
def mpRequest(ev,c_xA,c_yA,xB,yB,h,r,cache):
  formula = SecureFormula(ev, h,r, c_xA, c_yA,
              xB, yB, cache['a'], cache['t'],
              cache['x'], cache['y'])
  with formula as sf:
    c_xA,c_yA,xB,yB,h,r,ct,c_ca,c_cx,c_cy
          = sf.inputs
    t = now()
    pr = proximity(c_xA, c_yA, xB, yB, r)
    if 'x' in cache:
      v = speed(c_xA,c_yA,c_cx,c_cy,h,t-ct)
      alpha = random(1, k) * (v + c_ca)
      sf.output(pr + alpha)
      sf.output(alpha)
    else:
      sf.output(pr)

  out = formula.evaluate()
  c_result = out[0]

  cache['a'] = out[1] if 'x' in cache else 0
  cache['t'] = t
  cache['x'] = c_xA
  cache['y'] = c_yA
  return c_result
```

For the first run, the protocol simply returns the proximity result and caches the query's position and time. *Bob* also initializes a special cache value a which is used to accumulate all speed threshold checks. For following requests, the speed threshold v is combined with the accumulated speed threshold. By adding the proximity result to the accumulated speed threshold, *Bob* constructs `c_result`. Note that all values depending on *Alice*'s inputs are encrypted and not readable by *Bob*.

5 Privacy Guarantees of DecentMP

This section shows that DecentMP provides very strong privacy guarantees. The central privacy notion of this work is according to Definition 8, following the standard SMC security definitions of [15] against malicious adversaries.

Definition 8 (Privacy definition). *A protocol π is said to privately implement a functionality g against malicious adversaries if for every adversary $\mathcal{A}$ against the protocol π, there exist a simulator $\mathcal{S}$ such that:*

$$\{\mathit{IDEAL}_{g,\mathcal{S}}(\overrightarrow{x}, \overrightarrow{y})\} \stackrel{c}{\equiv} \{\mathit{REAL}_{\pi,\mathcal{A}}(\overrightarrow{x}, \overrightarrow{y})\}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability of distributions.

For space reasons, we recall the IDEAL and REAL constructions and the computational indistinguishability definitions in Appendix 2. The intuition behind this definition is that the implementation of g by π should be as secure as an ideal implementation of g using a third party. The desired functionality for DecentMP is specified by Definition 9.

Definition 9 (Constrained speed querying functionality). *The functionality of a speed-constraining functionality g is a function from queries to responses: $g : Q \to L$.*

$$g(q_1, \ldots, q_m)[i] = \begin{cases} \bot & \text{if } \exists_{j<i} : \frac{dist(p_j, p_{j+1})}{time(q_j) - time(q_{j+1})} > h \\ inProx(p_i, p_b, r) & \text{otherwise} \end{cases}$$

where $q_i = (p_i, t_i)$ and p_b is the position of Bob.

Bearing the functionality Definition 9 in mind, recall the protocol resulting from the formula shown in Listing 5. Combining the two, the privacy-guarantees sought for DecentMP are captured by Theorem 2.

Theorem 2 (Privacy guarantees of DecentMP). *The protocol π resulting from evaluating the program Listing 5 implements the functionality of Definition 9 privately according to Definition 8.*

5.1 Proofs

Now, to prove that DecentMP is secure according to Definition 8, we need to show: **a)** the protocol implements the desired functionality, that is, when used by honest parties, it implements the functionality g of Definition 9; and **b)** if miss-used by a malicious *Alice*, we can simulate *Alice*'s view based exclusively on her inputs and the outputs of the g.

Proof of correct functionality First, the functionality must be privacy-preserving in the presence of only benign parties. This is captured by Theorem 3, for which the following proof shows that DecentMP is indeed privacy-preserving in the absence of malicious adversaries.

Theorem 3. *DecentMP implements the functionality g as defined in Definition 9*

Proof. By construction, if `alpha` (henceforth α) evaluates to the encryption of 0, then the result of the proximity request is correctly computed and disclosed to *Alice*, building on the correctness of the proximity computation protocol. `cache['a']` initially encrypts 0, and remains constant if and only if every invocation of `speed` returns the encryption of 0, which is the case when the speed limitation is respected. The first time that the speed limitation is violated, α is not an encryption of 0, and thus `c_result` encrypts $\perp$. `cache['a']` accumulates the sum of the previous evaluations of α. Since α evaluates to the encryption of either $\perp$ or 0, after the first speed violation `cache['a']` is an encryption of $\perp$.

Proof of secure joint computation Now for the more interesting case, when adversaries deviate from the protocol to try to infer additional data about the victim. This is captured by Theorem 2, for which the proof is presented in the following. The intuition behind the proof of Theorem 2 is as follows. DecentMP defines α as an arithmetic formula. From the security guarantees of [9], it follows that a malicious *Alice* that tampers with the protocol at any point up to the evaluation of α, will cause α to encrypt $\perp$. This in turn will cause the proximity result sent to *Alice* to be random, and cause `cache['a']` to be updated just as in the case where *Alice* does not respect the MaxPace speed policy, making subsequent location responses yield an encryption of $\perp$.

Since we leverage on primitives of BetterTimes to build DecentMP, we recall the privacy guarantees we obtain from using this construction. The privacy guarantees of [9] against malicious adversaries can be summarized as follows. In this setting, the possessor of the private key (*Alice*) is considered potentially malicious, whereas the party performing homomorphic operations on encrypted data (*Bob*) is considered to be honest. Indeed, as it is usual the case with protocols based on partially homomorphic cryptography, in this setting *Alice* gets the result of the joint computation, which by construction *Bob* cannot learn. *Bob* could still sabotage the outcome of the computation, but no such adversary is considered in this setting, since our focus is on *privacy* guarantees. Now, if *Alice* would tamper with the protocol to try to learn more about the private inputs of *Bob* than allowed by the arithmetic formula (the

functionality), BetterTimes guarantees that she instead receives a fresh uniformly random value.

The main theorem of BetterTimes is proved by showing that all partial computations outsourced to *Alice* are independent and uniformly random, and the final value of the formula is $\perp$ if *Alice* does not comply with the protocol, and the correct output of the formula otherwise. This is captured by Lemma 1. The proof of Lemma 1 is presented in Appendix 3.

Lemma 1 (Fundamental lemma of BetterTimes). *For a fixed but arbitrary arithmetic formula $g(\overrightarrow{x}, \overrightarrow{y})$ represented by a recursive instruction $\iota \in$ **Ins** against the protocol π resulting from $evaluate(\iota)$, all intermediate messages to* Alice *are independent and uniformly random, and the last message $result$ of the protocol is an encryption of the output of $g(\overrightarrow{x}, \overrightarrow{y})$ if* Alice *is honest, and an encryption of a uniformly random value otherwise.*

However, Lemma 1 only shows that the final result is secured. Now, to show that also for intermediate values added to the output are secure, Lemma 1 can be extended to Lemma 2. The proof of Lemma 2 is found in Appendix 3.

Lemma 2 (Extension of Fundamental lemma of BetterTimes). *For a fixed but arbitrary arithmetic formula $g(\overrightarrow{x}, \overrightarrow{y})$ represented by a recursive instruction ι constructed using EasyTimes against the protocol π resulting from $evaluate(\iota)$, all intermediate messages to* Alice *are independent and uniformly random, and the final result and intermediate output values of the protocol are an encryption of $\perp$ for a dishonest* Alice.

Proof (Proof of Theorem 2).

First, note that from the extension to the BetterTimes system as detailed in Appendix 1 and from Lemma 2, Corollary 1 follows immediately.

Corollary 1. *For the arithmetic formula represented by the recursive instruction ι resulting from Listing 5, all intermediate values sent to* Alice *are encryptions of $\perp$. The intermediate output value from the evaluation of α and the final result are encryptions of $\perp$ if* Alice *deviates from the protocol.*

After performing m location queries *Alice* has observed the intermediate values in each query q_i and the respective location response l_i by *Bob*. From Corollary 1, and by construction of the `cache['a']`, it follows that if *Alice* cheats for the first time when jointly computing l_i with *Bob*, then $l_i = \perp$ and $\forall_{j>i}\ l_j = \perp$. Further, from Corollary 1, it follows directly that all intermediate values in a joint computation,

denoted by $\overrightarrow{v}_i$, are equal to $\perp$. Without loss of generality, let's assume that a class of malicious adversaries $\mathcal{A}_x$ are dishonest when jointly computing the location response l_x. The output of any such malicious $\mathcal{A}_x$ against DecentMP can be simulated by a simulator $\mathcal{S}_x$ that outputs:

$$\mathcal{S}_x(q_1, \ldots, q_m, l_1, \cdots l_m) = \\ \mathcal{A}_x(q_1, \ldots, q_m, l'_1, \cdots l'_m, \overrightarrow{v}_1, \cdots \overrightarrow{v}_m)$$

The outputs l'_i corresponding to the view of $\mathcal{A}$ in a real execution are easy to simulate, as they can be computed using only the inputs through:

$$l'_i = \begin{cases} \perp & \text{if } i \geq x \\ l_i & \text{otherwise} \end{cases}$$

And the intermediate messages can be simulated as $\overrightarrow{v}_j = \perp, \cdots, \perp$ as per Lemma 2.

6 Implementation and Benchmarks

This section describes the implementation of DecentMP and presents benchmarks from initial experiments. The prototype was done in Python using the library provided by [9]. The results show that it is feasible to use MaxPace in a decentralized setting for practical scenarios. Five typical scenarios were used, to give data on different speed thresholds (h). Four different values of the proximity threshold are measured (r).

The data shows that the protocol scales reasonably well in both h and r. Arguably, most configurations may be applicable to real applications already in its current state. Several configurations using the prototypical implementation finish within 500 milliseconds, and many applications can load data in the background and do not require instant feedback and would thus be able to use the more expensive settings.

6.1 System Overview

The protocol is implemented straight forward from the description in Section 4, with only one optimization effort. Instead of multiplying all values in the `lessThan` function while computing the proximity, the original idea of [10] to encode the values as an array is used. This reduces the number of round-trips and is securely realized by caching subtrees in the formula (which is needed since the distance is reused).

The additively homomorphic cryptographic scheme used for the experiments was the DGK scheme [4]. For the DGK scheme, the plaintext space is chosen separately from the key size. For decryption to be efficient a table of the size of the plaintext space is saved. A larger plaintext space means that more RAM is needed and also has some costs in terms of performance. The implementation keeps this table in memory, which gives a practical limitation due to RAM consumption (for 2048 bit keys, 22 bits of plaintext space requires 8192 MB RAM).

The plaintext space size is relevant in the context of location based services. As an over-approximation, consider a square with sides equal to the earth's circumference $4 \cdot 10^7$ meters. A coordinate is then $log_2(4 \cdot 10^7) \approx 25$ bits. Thus, 25 bits of plaintext space is needed to measure the earth with 1 meter resolution. 22 bits gives a precision of 5 meters for the earth.

The time between requests where not considered as attackers can be assumed to query often. A benign user may query rarely, but performance issues with large time spans for benign users can be resolved with a resetting strategy.

The benchmarks were performed on a single machine with 16MB RAM and an Intel i7-4790 CPU at 3.60GHz. Both the client and the server application were hosted locally, and were noted to be performing work of the same order of magnitude.

6.2 Performance

Benchmarks were carried out with a key of sizes 2048 bits, and plaintext space 22 bits. Figure 3 visualizes how the time of a single protocol execution time is affected by different configurations of h and r. Though configurations modeling higher speed and a larger radius cause longer protocol execution times, in both of the parameters r and h, the time to complete the protocol grows less than quadratic.

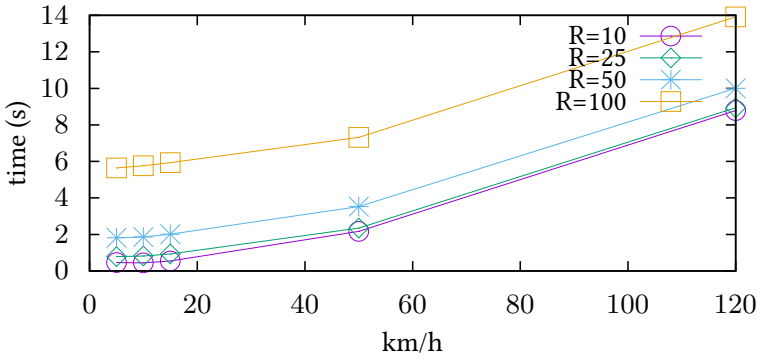

Fig. 3: Different speeds and proximity thresholds

Benchmarks where also performed for a much smaller plaintext space of 10 bits. The results show similar performance, with a difference of at most 200 milliseconds for any scenarios, compared to 22 bits of plaintext space. Given a machine with more RAM, a higher precision can be utilized without noticeably affecting user experience.

6.3 Communication Cost

The protocol may incur a significant communication cost. The number of round trips is dictated by the speed threshold, and the size of the messages depends on the key size. Further, the size of the result array is affected by the proximity threshold. Table 3 shows how different values of r and h affect communication. These numbers exclude the additional overhead of the structure of the messages, which is not significant.

Table 3: Communication cost in kilobytes (messages)

Activity	Proximity Threshold (meters)				Messages
	10	15	50	100	
Walking	166	610	2050	7392	11
Running	196	640	2080	7422	17
Cycling	286	730	2170	7512	35
Bus	1076	1520	2960	8302	193
Car	4706	5150	6590	11932	919

Communication cost ranges from 166 kilobytes to 12 megabytes. 12 MB is rather a lot of data for geometric computations, but seeing as most devices can handle high-quality video streaming, all results are within practical applicability.

7 Related Work

Location privacy is a well recognized issue, as seen from a diverse range of surveys: [12, 27, 16]. Protecting location disclosure during continuous queries through speed limitation has been applied in practice [20], but to the best of the authors' knowledge this is the first formalization of such approaches and the first which quantifies attacker effort in these cases. There are several active research areas which touch upon different components of this work. The following positions this work in relation to the more relevant neighboring approaches.

Within location privacy, different works protect different parts of a user's data. Many approaches provide k-anonymity [26, 16], where the location of the user is indistinguishable among a set of users, where the primary objective is to protect the identity from the attacker. This work protects the location, and does not consider privacy of the identity. Works of this type, though similar, are orthogonal to MaxPace.

Considering moving principals is a key feature of this work. To the best of the authors' knowledge, there is no literature on how to maintain privacy if both the attacker and the victim are moving. Within location proximity specifically, there is a fair amount of work [29, 23, 7, 17, 25, 19]. However, the majority of current research focuses on static principals. A practical application requires security over continuous queries.

One rather intuitive countermeasure for when the attacker is moving is mentioned by Narayanan et al. as a positive side-effect of their construction [19]. By mapping each principal to a grid cell, called a *cloaking region*, and calculating distances between the cloaking regions, nothing more than the region can be leaked. That is, it's impossible to solve the *DiskSearch* problem.

There are several drawbacks with cloaking regions. Foremost, a significant chance of both false positives and negatives, as illustrated in Figure 4. Further, as highlighted by Cuellar et al. [3], if an attacker samples the victims location as the victim changes region, they know that the victim is close to the region's border. In general, simply making use of cloaking regions have no effect on the *DiskCoverage* problem, as is the focus of MaxPace.

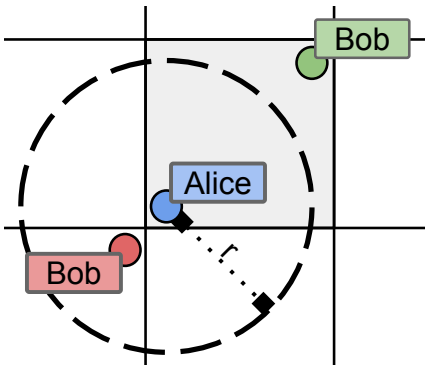

Fig. 4: Precision issues

Polakis et al. [20] investigate different disclosure strategies employed in the wild, such as disclosing distances or rounded distances. For several popular social networks, they perform measurements of how quickly different attack strategies can solve both the *DiskCoverage* and *DiskSearch* problems. Polakis et al. advocate using a cloaking region to improve privacy. As mentioned earlier, this helps to some extent when the victim is static for the *DiskSearch* problem, but has no effect on the *DiskCoverage* problem.

Lastly, one prime feature of MaxPace is that it is possible to deploy without a trusted third party. As highlighted above, trust may be used to enforce any policy

with low computational effort, but such approaches have trouble when the aim is formal privacy guarantees [16].

8 Conclusions

We have developed MaxPace, a framework for speed-constrained location queries. We have demonstrated the advantages over unrestricted location queries by comparing bounds on an attacker's knowledge. The framework is susceptible to both centralized and decentralized deployment. The former has already found a way into practical location-based services. For the latter, we have devised a speed-constrained secure multi-party computation protocol for location proximity and formally established its privacy guarantees. We have reported on experiments with a prototype implementation which shows that the protocol can be used in practice.

Our knowledge bounds focus on the disk coverage problem as the main contributor to an attacker's knowledge. A study of the disk search problem is subject to future work. In addition, we are interested in further developing the resetting strategies outlined in Section 2. This will allow MaxPace to deal with the imprecision of GPS and high-speed transportation. Finally, we plan to investigate scenarios where, in addition to Alice, Bob may also move in-between requests. While Bob's movement makes trilateration more difficult, more work is needed to quantify how Bob's movements affect the privacy guarantees.

Acknowledgments This work was funded by the European Community under the ProSecuToR project and the Swedish research agencies SSF and VR.

References

1. C. Bessette. Does Uber Even Deserve Our Trust? `http://www.forbes.com/sites/chanellebessette/2014/11/25/does-uber-even-deserve-our-trust/`, Nov. 2014.
2. D. Coldewey. "Girls Around Me" Creeper App Just Might Get People To Pay Attention To Privacy Settings. `http://techcrunch.com/2012/03/30/girls-around-me-creeper-app-just-might-get-people-to-pay-attention-to-privacy-settings/`, Mar. 2012.
3. 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.
4. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *Information Security and*

Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer, 2007.

5. K. Dreyer. Mobile Internet Usage Skyrockets in Past 4 Years to Overtake Desktop as Most Used Digital Platform. `http://www.comscore.com/Insights/Blog/Mobile-Internet-Usage-Skyrockets-in-Past-4-Years-to-Overtake-Desktop-as-Most-Used-Digital-Platform`, Apr. 2015.
6. 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, 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings*, volume 5672 of *Lecture Notes in Computer Science*, pages 235–253. Springer, 2009.
7. 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, *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 309–318. ACM, 2010.
8. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.
9. P. A. Hallgren, M. Ochoa, and A. Sabelfeld. Bettertimes - privacy-assured outsourced multiplications for additively homomorphic encryption on finite fields. In M. H. Au and A. Miyaji, editors, *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*, volume 9451 of *Lecture Notes in Computer Science*, pages 291–309. Springer, 2015.
10. P. A. Hallgren, M. Ochoa, and A. Sabelfeld. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In A. A. Ghorbani, V. Torra, H. Hisil, A. Miri, A. Koltuksuz, J. Zhang, M. Sensoy, J. García-Alfaro, and I. Zincir, editors, *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, pages 1–6. IEEE Computer Society, 2015.
11. A. Khanna. Stalking Your Friends with Facebook Messenger. `https://medium.com/faith-and-future/stalking-your-friends-with-facebook-messenger-9da8820bd27d`, May 2015.
12. J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.
13. A. Lella. Number of Mobile-Only Internet Users Now Exceeds Desktop-Only in the U.S. `https://www.comscore.com/Insights/Blog/Number-of-Mobile-Only-Internet-Users-Now-Exceeds-Desktop-Only-in-the-U.S`, Apr. 2015.
14. 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.

15. Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008:197, 2008.
16. E. Magkos. Cryptographic approaches for privacy preservation in location-based services: A survey. *IJITSA*, 4(2):48–69, 2011.
17. 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.
18. W. Mathworld. Circle-circle intersection, 2008.
19. 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.
20. I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis. Where's wally?: Precise user discovery attacks in location proximity services. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 817–828. ACM, 2015.
21. 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.
22. A. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In D. H. 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.
23. 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.
24. L. Siksnys, 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, *Eleventh International Conference on Mobile Data Management, MDM 2010, Kanas City, Missouri, USA, 23-26 May 2010*, pages 75–84. IEEE Computer Society, 2010.
25. L. Siksnys, 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, *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009,*

Aalborg, Denmark, July 8-10, 2009, Proceedings, volume 5644 of *Lecture Notes in Computer Science*, pages 405–410. Springer, 2009.

26. N. Talukder and S. I. Ahamed. Preventing multi-query attack in location-based services. In S. Wetzel, C. Nita-Rotaru, and F. Stajano, editors, *Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010, Hoboken, New Jersey, USA, March 22-24, 2010*, pages 25–36. ACM, 2010.
27. M. Terrovitis. Privacy preservation in the dissemination of location data. *SIGKDD Explorations*, 13(1):6–18, 2011.
28. M. Veytsman. How i was able to track the location of any tinder user. `http://blog.includesecurity.com/2014/02/how-i-was-able-to-track-location-of-any.html`, Feb. 2014.
29. 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 BetterTimes Extension

This section describes a small extension to the BetterTimes system. The extension allows for intermediate values to be securely used outside of the circuit. The *assurance value*, which is an internal value of a BetterTimes formula, is handled separately for these cases. The assurance value carries evidence of whether or not *Alice* has cheated up to the point when the assurance value is computed. The assurance value of *all* outputs are combined using the extension, such that misbehavior at any time while the formula is computed yields only $\perp$ values. Figure 5 shows an example of how the extension is used in DecentMP.

Some background about the features which are used by BetterTimes-instructions are needed in the context of this new construction. BetterTimes-instructions are a recursive data structure, where an instantiated instruction is either a binary operation, for which each operand is another instruction, or a scalar value. These are an addition function $[\![c_1]\!] \oplus [\![c_2]\!] = E(m_1 + m_2)$, a unary negation function $\neg[\![c_1]\!] = E(-m_1)$ and a multiplication function $[\![c_1]\!] \odot m_2 = E(m_1 \cdot m_2)$. Where $[\![x]\!]$ is used to denote that the variable x is encrypted using *Alice*'s public key, $E(m)$ is the encryption function applied to m using *Alice*'s public key, and $[\![c_1]\!]$ and $[\![c_2]\!]$ encrypts m_1 and m_2, respectively BetterTimes also provide means of computing multiplications of two ciphertexts as $[\![c_1]\!] \odot [\![c_2]\!] = E(m_1 \cdot m_2)$ through a secure outsourcing technique in which *Alice* is engaged in interactive protocol.

Definition 10 (Intermediate BetterTimes outputs). *The new BetterTimes operation Ins_O registers the result of an instruction as intermediate output.*

As defined in the original paper $evaluate$ computes the formula, and accumulates all assurance values $[\![a_i]\!]$. The assurance values are summed as the evaluation proceeds. Finally, the assurance value is randomized and added to the result as $([\![result]\!] \oplus [\![a]\!]) \odot \rho$, with ρ random. Note that these operations do not take place in the plaintexts, but are computed homomorphically on the ciphertext. Using the extension, when $evaluate$ encounters an Ins_O instruction, it computes the result and the assurance value as normal, but also saves the intermediate result $[\![z_i]\!]$ and the current assurance value $[\![a_i]\!]$ as a tuple $([\![z_i]\!], [\![a_i]\!])$ to a store S.

After a formula has been fully evaluated, instead of outputting the result directly, a list is constructed where all outputs are assured using a overarching assurance

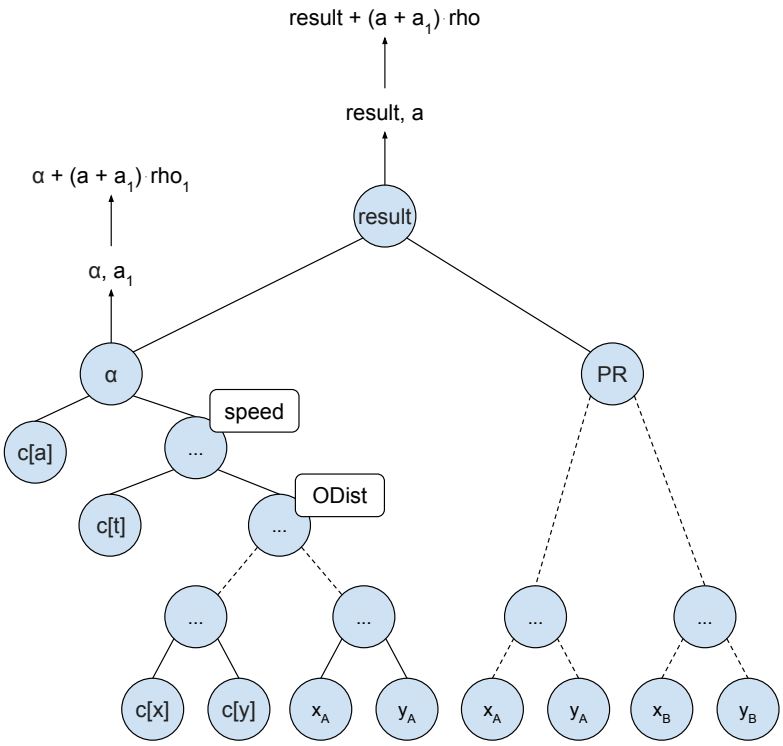

Fig. 5: Example of how the new BetterTimes extension is used

value A, defined as:

$$[\![A]\!] = [\![a]\!] \oplus \left(\bigoplus_{i=0}^{|S|} snd(S[i])\right)$$

Where $[\![a]\!]$ is the final assurance value and $snd()$ retrieves the second value in a tuple ($fst()$ is used below to retrieve the first value).

Finally, the array of the final result and all intermediate values is given as:

$$[([\![result]\!] \oplus [\![A]\!] \odot \rho] :: [(fst(s) \oplus [\![A]\!]) \odot \rho_i \text{ for } s \in S]$$

Where ρ and all ρ_i are independent and uniformly random variables, and :: denotes array concatenation.

2 SMC Notions

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

2.1 Negligible Functions

For the scope of this work, what it means for a function to be negligible is shown in Definition 11

Definition 11. *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.

2.2 Indistinguishability

Indistinguishability is an important notion for the proofs presented in this paper, and is specified formally in Definition 12.

Definition 12. *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:*

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

2.3 Ideal and Real Execution

The IDEAL and REAL executions follow the definitions of Pinkas and Lindell [15]. As highlighted previously, in the IDEAL model the parties interact only with a trusted third party, which ensures that the executed protocol matches exactly an implementation of the functionality, where parties cannot deviate from the protocol. In the REAL model, instead a concrete instance of the protocol is considered.

Let $\overrightarrow{x} \in I_A$ and $\overrightarrow{y} \in I_B$ be the private inputs for two parties, and let $g(\overrightarrow{x}, \overrightarrow{y}) \in O_A \times O_B$ be the output of a functionality g.

Here follows a brief recap of the execution in the IDEAL model, where an adversary $\mathcal{A}_{\text{IDEAL}}$ is controlling a corrupted party (Alice for the context of this paper). The benign party (Bob) sends their input $\overrightarrow{y}$ to the trusted party, and $\mathcal{A}_{\text{IDEAL}}$ tells the corrupted party (Alice) to either send the actual input of Alice (which can be read by $\mathcal{A}_{\text{IDEAL}}$) or another value of the same length as $\overrightarrow{x}$ to the trusted party. The trusted party then computes the output to be received by both parties. For the scope of this paper, Bob has no output, and Alice receives $g(\overrightarrow{x}, \overrightarrow{y})$. Alice forwards the result to $\mathcal{A}_{\text{IDEAL}}$. The original definition also handles the case when $\mathcal{A}_{\text{IDEAL}}$ wishes to abort the protocol. In the context of this paper, since the SMC solution is based on homomorphic encryption, *Bob* receives no output from the ideal functionality. Therefore, it does not make sense for the adversary to abort the protocol. Also, this

means that fairness guarantees for *Bob* are out of scope, so abortions of the protocol by the simulator do not need to be accounted for.

After the *ideal execution* of a functionality on inputs $(\overrightarrow{x}, \overrightarrow{y})$ $\mathcal{A}$ outputs an arbitrary PPT function on the private input of *Alice* and the output of the functionality $(\overrightarrow{x}, g(\overrightarrow{x}, \overrightarrow{y}))$. Formally thus:

$$\mathcal{A}_{\text{IDEAL}} : I_A \times O_A \to O_{\mathcal{A}}$$

for an arbitrary but fixed output space $O_{\mathcal{A}}$ (for instance a string of bits of length n).

The *real execution* of a concrete protocol π is rather intuitive, where $\mathcal{A}_{\text{REAL}}$ takes the place of the corrupted party and acts on their behalf. In this case:

$$\mathcal{A}_{\text{REAL}} : I_A \times view_{\pi}^{A} \times O_A \to O_{\mathcal{A}}$$

where $view_{\pi}^{A}$ are the intermediate values seen by $\mathcal{A}_{\text{REAL}}$ during the execution of π.

Recall that Definition 8 requires that for any adversary $\mathcal{A}_{\text{REAL}}$ against a protocol, there exists a simulator:

$$\mathcal{S} : I_A \times O_A \to O_{\mathcal{A}}$$

such that the distribution of the outputs of $\mathcal{S}$ and $\mathcal{A}$ are computationally indistinguishable:

$$\{\text{IDEAL}_{g,\mathcal{S}}(\overrightarrow{x}, \overrightarrow{y})\} \overset{c}{\equiv} \{\text{REAL}_{\pi,\mathcal{A}}(\overrightarrow{x}, \overrightarrow{y})\}$$

That is, a protocol π is privacy-preserving if it is possible to construct a concrete PPT $\mathcal{S}$ such that for every attacker $\mathcal{A}$ against the real protocol, using only the information available to the attacker by construction (their inputs and the output), its output is indistinguishable from the one of $\mathcal{A}$. If this is possible, an adversary does not learn anything apart from what is disclosed by the functionality by attacking π.

3 Proofs

Recall that by definition, BetterTimes outsources multiplications to *Alice* as depicted in Fig. 6. Since algorithm is probabilistic, we use the pWhile probabilistic imperative programming language of [**?**] for the description, where $x \xleftarrow{\$} M$ stands for uniformly random assignation of a value in the set M to x.

Proc. $BetterTimes([\![x]\!], [\![y]\!])$:
$c_a \xleftarrow{\$} \{0..p\}; c_m \xleftarrow{\$} \{1..p\};$
$b_x \xleftarrow{\$} \{0..p\}; b_y \xleftarrow{\$} \{0..p\};$
$\rho \xleftarrow{\$} \{1..p\};$
// Blind operands
$[\![x']\!] \leftarrow [\![x]\!] \oplus [\![b_x]\!]; [\![y']\!] \leftarrow [\![y]\!] \oplus [\![b_y]\!];$
// Create challenge
$[\![c]\!] \leftarrow ([\![x']\!] \oplus [\![c_a]\!]) \odot c_m;$
// Outsource multiplication
$([\![z']\!], [\![a']\!]) \leftarrow OS([\![x']\!], [\![y']\!], [\![c]\!]);$
// Compute assurance value
$[\![a]\!] \leftarrow ([\![a']\!] \ominus [\![z']\!] \odot c_m \ominus [\![y']\!] \odot (c_a \cdot c_m)) \odot \rho;$
// Un-blind multiplication
$[\![z]\!] \leftarrow [\![z']\!] \ominus ([\![x']\!] \odot b_y \oplus [\![y']\!] \odot b_x \oplus [\![b_x \cdot b_y]\!])$;
return $([\![a]\!], [\![z]\!])$;

Fig. 6: Outsourced multiplication with BetterTimes

Proof (Proof of Lemma 1).

It follows from the definition of Figure 6, that the intermediate values observed by an adversary $\mathcal{A}$ during the protocol execution are encryptions of uniformly random independent triples $(\perp, \perp, \perp)$ corresponding to $[\![x']\!], [\![y']\!]$ and $[\![c]\!]$, which are independently blinded.

From Lemma 1 of [9], the decryption of the assurance value $[\![a]\!]$ is indistinguishable from $\perp$ if an adversary is dishonest. Since the assurance value is added to the result of the computation, the final value is also $\perp$.

Now for the proof of Lemma 2:

Proof (Proof of Lemma 2). By construction, all intermediate messages remain unchanged by the introduction of Ins_O. Showing that intermediate messages and the final output are $\perp$ if *Alice* is dishonest follows from the observations in the proof of Lemma 1, with one small addition. The sum $[\![a]\!] + \bigoplus_{i=0}^{|S|} snd(S[i])$ is the encryption of 0 only with negligible probability since all terms are fresh uniformly random variables.

4 EasyTimes: syntactic improvement of BetterTimes

Herein a python-like syntax is described, for which a direct translation into BetterTimes syntax is provided. The code presented here is python-compatible, and thus

can be executed by the regular python interpreter. See Listing 6 for an example of how this is applied to a concrete example.

Listing 6: Example evaluation

```
evaluator = StringEvaluator()
formula = SecureFormula(evaluator, 4, 3, 2, 1)

def foo(x, z, i2):
  return x + z - i2

with formula as sf:
  i1, i2, i3, i4 = sf.inputs

  c = i1 + i2
  sf.output(c)

  z = i4
  x = i1 * 42 + i2
  x = foo(x, z, i2)
  y = i2 * i3 + i4
  o_1 = x + y
  o_2 = y * z

  sf.output(c * (i3 - i2))

outs = formula.evaluate()
for out in outs:
  print(out)
```

In Listing 6, there are four inputs provided to the formula. There are two outputs, $i_1 + i_2$ and finally $x * y$, which is computed using a separate method. Regardless of how the output is constructed, even when control flow primitives such as if statements and loops are used, the result is a single BetterTimes formula. A `SecureFormula` requires an *evaluator* (which will be discussed later) and an arbitrary number of inputs. The operations done with the inputs, and what the resulting outputs are, is defined by operating inside a **`with`** block. Upon leaving the **`with`** block, variables are no longer mutable, and the formula is fixed. Thus, at this point, the for-

mula can be evaluated. An implementation of such a construct is concise in python, as shown in Listing 7.

Listing 7: Python-class for secure formula

```
class SecureFormula(object):
  def __init__(self, evaluator, *inputs):
    self.__evaluator = evaluator
    formula_inputs = []
    for inp in inputs:
      formula_inputs.append(
          FormulaInstruction(inp))

    self.__context = FormulaContext(
        formula_inputs)
    self.__outputs = None

  def __enter__(self):
    return self.__context

  def __exit__(self, exc_type, exc_val,
               exc_tb):
    self.__outputs = self.__context.outputs

  def evaluate(self):
    return [self.__evaluator.evaluate(out)
            for out in self.__outputs]
```

The `__enter__` method returns a `FormulaContext` object, and is implicitly called when a `SecureFormula` instance is used for a **`with`** statement. The context is basically just a controlled store for inputs and outputs, as seen in Listing 8.

Listing 8: Python-class for formula context

```
class FormulaContext(object):
  def __init__(self, inputs):
    self.__inputs = inputs
    self.__outputs = []

  def output(self, output):
```

```
    self.__outputs.append(output)

  @property
  def inputs(self):
    return self.__inputs

  @property
  def outputs(self):
    return self.__outputs
```

From the context, representations of the input variables can be obtained. When a variable is provided to the `sf.output()` method, the result is marked as a part of the output of the context. representing are stored, by the construction of the `FormulaInstruction` object. A slightly simplified `FormulaInstruction` is seen in Listing 9.

Listing 9: Python-class for formula instruction

```
class FormulaInstruction(object):
  __ADD = object()
  __SUB = object()
  __MUL = object()

  def __init__(self, a=None, b=None, c=None):
    if b:
      self._operation = a
      self._left_operand = b
      self._right_operand = c
    else:
      self._scalar = a

  @property
  def left_operand(self):
    return self._left_operand

  @property
  def right_operand(self):
    return self._right_operand
```

```
  @property
  def scalar(self):
    if self.is_scalar():
      return self._scalar
    else:
      return None

  def is_scalar(self):
    return hasattr(self, '_scalar')

  def is_add(self):
    return self._operation == self.__ADD

  def is_sub(self):
    return self._operation == self.__SUB

  def is_mul(self):
    return self._operation == self.__MUL

  def __op(self, op, o1, o2):
    return FormulaInstruction(op, o1, o2)

  def __add__(self, other):
    return self.__op(self.__ADD, self, other)

  def __sub__(self, other):
    return self.__op(self.__SUB, self, other)

  def __mul__(self, other):
    return self.__op(self.__MUL, self, other)
```

Now to see how accumulating `FormulaInstruction`'s and storing them as outputs is a mapping to BetterTimes instructions, consider the `StringEvaluator` shown in Listing 10. This evaluator consumes an instruction, then recursively explores all branches of the formula, and finally prints them in BetterTimes syntax.

Listing 10: Python-class for formula instruction

```
class StringEvaluator(FormulaEvaluator):
```

```
  def evaluate(self, instruction):
    if instruction.is_scalar():
      return 'Ins(%s)' % instruction.scalar
    return "Ins(%s, %s, %s)" % (
      self.__bt_op(instruction),
      self.evaluate(instruction.left_operand),
      self.evaluate(instruction.right_operand)
    )

  def __bt_op(a, instruction):
    if instruction.is_add():
      return 'ADD'
    elif instruction.is_sub():
      return 'SUB'
    elif instruction.is_mul():
      return 'MUL'
```

By mapping an arbitrary `SecureFormula` to BetterTimes syntax, it's easy to see that any formula constructed using this syntax can be evaluated securely using the BetterTimes system.

Paper **four**

LOCATION-ENHANCED AUTHENTICATION USING THE IOT

Because You Cannot Be in Two Places at Once

IOANNIS AGADAKOS, PER A. HALLGREN, DIMITRIOS DAMOPOULOS,
ANDREI SABELFELD AND GEORGIOS PORTOKALIDIS

User location can act as an additional factor of authentication in scenarios where physical presence is required, such as when making in-person purchases or unlocking a vehicle. This paper proposes a novel approach for estimating user location and modeling user movement using the Internet of Things (IoT). Our goal is to utilize its scale and diversity to estimate location more robustly, than solutions based on smartphones alone, and stop adversaries from using compromised user credentials (e.g., stolen keys, passwords, etc.), when sufficient evidence physically locates them elsewhere. To locate users, we leverage the increasing number of IoT devices carried and used by them and the smart environments that observe these devices. We also exploit the ability of many IoT devices to "sense" the user. To demonstrate our approach, we build a system, called Icelus. Our experiments with it show that it exhibits a smaller false-rejection rate than smartphone-based location-based authentication (LBA) and it rejects attackers with few errors (i.e., false acceptances).

PUBLISHED IN THE 32ND CONFERENCE ON COMPUTER SECURITY APPLICATIONS
ACSAC 2016, LOS ANGELES, CA, USA, DECEMBER 5-9, 2016

1 Introduction

Electronic user authentication is increasingly used in the physical world, where it is frequently employed to protect financial transactions and to control access to physical spaces and vehicles. Typical means to authenticate users entry include passwords and PIN codes, tokens (e.g., smartcards), and biometrics (e.g., fingerprints). Cards are frequently used to unlock doors, mainly in offices, either through swiping the card through a reader or by proximity of an RFID-based card to the reader. Smart locks (e.g., Kevo) enable user's to use their smartphone instead of a key, while an increasing number of vehicles use wireless key fobs to unlock their doors and start the engine. Credit and debit cards, and even smartphones today, also act as tokens that (usually) along with a PIN code enable users to authorize transactions.

While these advances have improved convenience and even security, they are not without problems. Fraudsters engage in various forms of deception for financial gain, like in Japan where $13m were stolen from ATMs [24]. The methods employed involve stealing, cloning, and counterfeiting credit and debit cards to perform transactions at POS [5, 43]. In the US such attacks are bolstered also by the limited deployment of PIN and chip technology [28]. Systems like Android Pay and Apple Pay, that enable users to pay with their smartphones or smart-wearables, can also be compromised if the PIN code [13, 52] or biometric [1, 14] used is bypassed. Door and car locks have also suffered various types of attacks including cloning RFID cards [17], relaying signals [6], and exploiting weakness in the authentication protocols [60, 63].

The state of the art in authentication mandates using multi-factor authentication, that is, combining a secret, a token, and a biometric. Interestingly, card-based financial transactions already use two factors, a PIN code and a card token. Security is compromised by eavesdropping the PIN (e.g., through tampered terminals) and creating a copy of the card. In other cases, multi-factor authentication is not used correctly [8] or not used at all because of usability issues [2]. Biometrics, such as fingerprints, have also been found to be vulnerable to attacks [14, 58] or reduce utility [40, 48].

Certain promising approaches employ user location as an additional factor of authentication for financial transactions [23, 41, 46]. They exploit the fact that users rarely get separated from their smartphones [29] and use them to either confirm the user's location or transparently provide location as an authentication factor. These approaches can fail if the smartphone is not present or not operational [20] (e.g., due to limited battery life), which may have stifled their broad deployment. Smartphones and other portable devices have been also used as proximity-based tokens [32].

In this paper, we propose using the Internet of Things (IoT) to model user location and movement for making user location continuously available as an additional factor of authentication, independently of whether a device is available (online) when the user authenticates. In contrast to prior works in location-based authentication (LBA) works, we argue that using the increasing number of smart things that users carry, wear, drive, or even have in their body, enables more robust methods for estimating user location. In other words, it allows us to estimate the location of users despite individual devices being offline or not with them.

IoT devices can help us locate their user by reporting their location (e.g., through GPS or WiFi) or by proximity to other other devices with known coordinates (e.g., wearables). Smart environments can also "observe" user things. For example, access points, smarthome hubs, etc. can report which devices are connected, financial institutions can report when and where a credit card is used, and smart traffic lights can report the location of vehicles. However, IoT devices can do much more, they can "sense" when they are being used. For instance, wrist wearables know when they are being worn because they sense walking and/or the user's heartbeat, and a smartphone that has just been unlocked with a PIN or fingerprint knows that the user is holding it. This is crucial in estimating whether a user is with a set of devices, as we no longer need to assume that users are de-facto with their smartphone, smartwatch, etc.

We couple location and activity data reported by devices to model users and their movement. Maintaining such a model enables us to estimate how likely it is that a user is at a particular location, without relying on any device being available and able to provide the user's location at the moment of authentication. Moreover, it enables us to use potentially-sparse data, as certain IoT devices may only report occasionally.

Another factor differentiating this work from previous ones is the way we use location data. Querying parties are *not* allowed to ask for the coordinates of any user device. Instead, they can place generic queries such as "Can the user be physically present at this location?" By only allowing such queries, we inhibit "curious" services from attempting to arbitrarily locate the user. More important, to respond to queries, we rely on evidence indicating that user is *not* at a given location. For example, we respond positively, only if we are confident that a user is not at a location. This strategy serves a twofold goal; first, to prevent falsely rejecting users that forget at home or have inoperable devices and, second, to prevent stolen devices from being misused to subvert the system.

To demonstrate our approach, we design and develop Icelus, a system that collects location and activity data from IoT devices to model user movement and location. Icelus can run as a service on a device of the user, such as a smarthome hub [30], or

it can be hosted in the cloud [42]. To collect data, it organizes the various devices in a hierarchy, so that the ones with Internet connectivity can relay the data of the ones without to the system. Third-party systems can also provide data by directly connecting to Icelus or indirectly by forwarding notifications of certain events (e.g., the use of a credit card at a location, an entry in the user's calendar, etc.). To alleviate privacy concerns, we also develop a privacy-preserving extension of the protocol used in Icelus that allows us to operate purely on distances, without revealing the actual locations of individual devices. At the core of the extension is a secure multi-party computation protocol that leverages additively homomorphic encryption and blinding. Finally, we evaluate Icelus by deploying it on set of devices readily available today.

Briefly, our contributions are the following:

- We propose a new approach that utilizes the IoT to estimate the location of a user and use it as an additional factor of authentication that is more robust that smartphone-only approaches .
- We develop a user movement model using the location data provided by IoT devices, which enables us to operate even when devices are not reachable.
- We define a method for determining the probability (referred to as confidence score) that the user actually is with a set of his devices, utilizing both the number of devices present and the user activities captured by device sensors.
- We develop Icelus, a prototype system that implements the proposed approach.
- We define a privacy-preserving protocol and formally establish privacy guarantees under an honest but curious attacker.
- We evaluate our approach by deploying Icelus on a set of devices readily available today and performing two field studies. Our results show that we can achieve a false-rejection rate of 4%-6%, which is lower than that of smartphone-based location-based authentication. At the same time, we are more resilient to attacks. We also evaluate the performance of our approach and find that it imposes negligible overhead on the devices tested of below 1%.

2 Threat Model

The attacks we aim to thwart in this paper include attempts to bypass user authentication with physical objects and terminals to gain unauthorized access to places, property, etc. of the user, as well as third-parties. Such attacks may include compromising passwords, security tokens (e.g., swipe cards and USB keys), or biometrics. The methods employed can involve stealing, cloning, and counterfeiting credit and

debit cards to perform transactions at POS [5, 43]. In the US such attacks are bolstered also by the limited deployment of PIN and chip technology [28]. Systems like Android Pay and Apple Pay, that enable users to pay with their smartphones or smartwearables, can also be compromised if the PIN code [13,52] or biometric [1,14] used is bypassed. Door and car locks have also suffered various types of attacks including cloning RFID cards [17], relaying signals [6], and exploiting weakness in the authentication protocols [60, 63].

We do not assume that the user's devices have not been compromised, instead our model considers that they could be physically stolen, tampered, or remotely compromised. We rely on the scale of the IoT for resistance to subversion. Our goal is to maintain correct operation, as long as the majority of devices have not been compromised.

3 Approach Overview

We propose using the IoT to estimate *where* a user can possibly be and use the *confidence* of our estimations to augment physical security decisions. For example, if someone enters the credentials of a user at a known physical location, such as a door keypad, we want to be able to answer the question "Can the user be in front of the door at this time?" Being able to answer this question will improve security, as the presentation of credentials on physical terminals without the legitimate user being present can indicate that a credential has been stolen or compromised. Our approach can enable policies that reject credentials or request additional identification, when it is determined that the user cannot physically be at the point of authentication (e.g., activate multi-factor authentication).

3.1 Viewing the Real World Through the IoT

An increasing number of objects contain computational and networking capabilities. The Internet of Things consists of objects that are carried by users or reside in their environment. They are able to sense each other (e.g., through Bluetooth) and frequently communicate with each other. They are also able to sense the environment and their user, e.g., wrist wearables can sense the heartbeat of the wearer.

We claim that through the IoT we can glimpse into the physical world to establish the location of their users. For example, according to a study 79% of people aged 18-44 have their smartphones with them 22 hours a day [29]. Smartphones can also establish their location by using information from network base stations, GPS, and WiFi, hence, they provide a strong indicator of their owner's location. Other objects,

like tablets or modern cars, also come with Internet connectivity and GPS, but may be shared among a few people, like the owner's family. They also provide hints, albeit weaker ones than smartphones, on the location of at least one of its usual users. These hints that can be significantly strengthened, when individual users can be authenticated. For example, both iOS and Android support multiple users, and certain BMW vehicles also support driver profiles.

Certain devices, like many wearables, do not feature GPS. However, they are able to connect to other devices through a wireless protocol, like Bluetooth, WiFi, etc. Such devices provide a different type of hint regarding the location of the user; because they are usually personal devices and the connection protocols have a limited range, being able to connect, or even just establishing the presence, of one indicates that the user is nearby. For example, paired Bluetooth devices can establish each other's presence, while the Bluetooth Low Energy (BLE) protocol enables the same without pairing. If we can establish the location of a single device in such a *cluster*, we are able to "ground" it and locate the user. Similarly, it is sufficient that a single device in a cluster is able to connect, directly or indirectly, to the Internet to make this information available to other parties.

Moreover, many IoT devices are not only able to sense each other but also the user. Smartphones and tablets provide PIN-based or even biometric-based authentication, fitness wearables can establish user movement and heart rate, while smart in-body health devices, such as insulin pumps, are always on the user. Thus, they can also help us detect when they are actually used by the user instead of being idle.

Hints on the location of a user are be also provided by third parties that observe one of the objects that the user owns. Observations are not limited to devices; tokens, like credit cards and passwords, are also "things" that can be observed. For instance, the bank observes that the user has used a debit card at an ATM or POS, and an employer notices that the user entered his credentials at a keypad-protected office door.

An example of devices and tokens owned by users and third parties that can observe them is depicted in Fig. 1. Collecting information from user-owned devices and third-parties provides us with the locations of his things. We assume that the data are collected under the purview of the user, for example, by a service hosted on the cloud with the user retaining ownership and control of the data. Of course, estimating the location of a user's things does not mandate that the user is necessarily with them, which raises the question: "How *confident* are we that the user is actually with a set of his things?", which is described in Sec. 5.

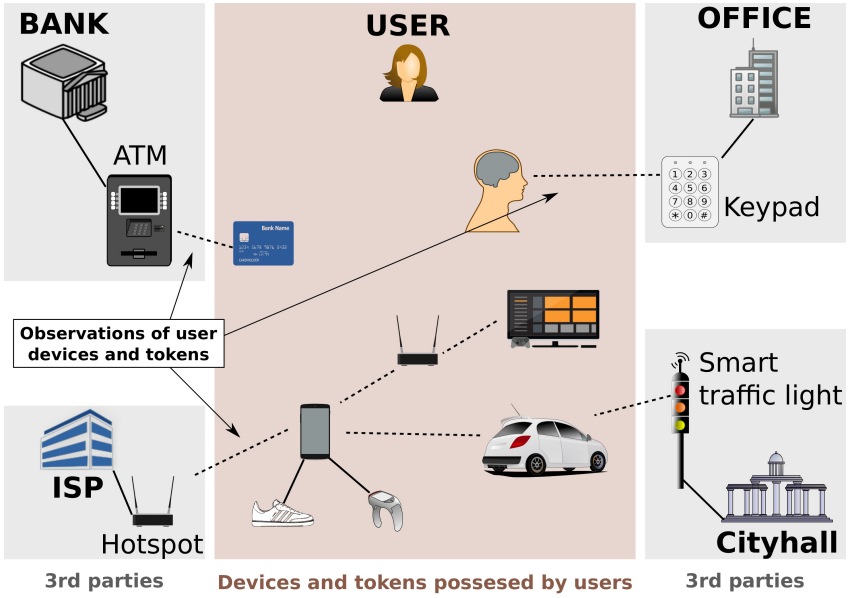

Fig. 1. "Things" owned by users and third-parties that can observe them. By collecting reports containing location information and proximity between things we can estimate user location.

3.2 Modeling User Location and Movement

We model users through *Avatars*, essentially their representations in the digital world, and multiple Avatars may concurrently exist for the same user. The location of an Avatar can be updated whenever information, including location information, is received from an IoT device. Since location reporting is not continuous, due to limited resources and connectivity, user movement must also be taken into account to enable meaningful location-possibility queries at any given time. User movement speed can be estimated using recent location reports and device sensor data [37]. However, we can also model certain vehicles or modes of transportation. Users walking, driving, or cycling can attain speeds within well established parameters. So by estimating an Avatar's speed, we can at any point establish the range of an Avatar, i.e., without making any assumptions on its direction remaining consistent, we can define the region where it is physically possible for the actual user to be. Additional physical world characteristics, like terrain and road networks, can also be incorporated to more accurately establish the range of an Avatar. For example, when on a car, the user is limited to driving on roads. However, we do not explore them in this paper.

The advantage of modeling the user is that we can remain operational even if there are inaccuracies in the reported data, location updates are sparse, and some devices have been compromised.

3.3 Using Location in Authentication

By obtaining a set of locations where the user may be, we enable pre-authorized locations to issue binary, "yes" or "no" type queries about whether the user can be present at the registered location. The reason for such queries is to avoid leaking the location of the user unnecessarily. If the user has just presented a security token at a location (e.g., a credit card), responding "yes" would confirm that the user is at the location, without leaking any additional information. If the response is "no", the service gains no additional information. A single third party with many pre-registered locations could attempt to maliciously narrow down the area where a user may be, by issuing multiple queries. However, such entities can be easily singled out and submitted to throttling or blocked entirely. As a result, location can be made part of a security decision without actually divulging the exact coordinates of the user.

Responses can be generated using a variety of policies. For example, we may want to simply check that there are no strong indicators placing the user at a different location, essentially, looking for paradoxes (the user cannot be in two places at once). Alternatively, we may actually require evidence that the user is actually at a particular location. In this paper, we adopt and evaluate the first strategy. Particularly, we respond negatively, only if there is an Avatar that cannot be at the querying location and its confidence is over a *rejection threshold*. Different services may require a different threshold, as not all actions have the same gravity. For example, the wrong person entering the gym is not as a serious problem as a fraudster withdrawing $10,000 from a bank account.

3.4 An Example

We illustrate how our approach through the example shown in Fig. 2. At 8:00 AM, the user is at home with all of his devices, including a tablet, a smartphone, sensors in his shoes, and his smartwatch, getting ready to walk to work. His WiFi access point also confirms that all of his devices are at home. He leaves home without his tablet and, at 8:15 AM, he stops at a coffee shop to buy a cup of coffee using his credit card. At this point, there are two Avatars. The first, is associated with the user's tablet. Since it is only a single device and it is idle, the confidence for that Avatar is low. However, because the tablet is powered on, it still sends regular reports, allowing us to limit the range of that Avatar around the house. The second Avatar is at the coffee shop, where an activity report from the user's bank has placed his credit card, and where the smartphone reported itself and the rest of the user's devices.

Finally, the user arrives at the office at 8:35 AM. He swipes his badge to enter, at which point the office queries the service. The moment the query is made, there are

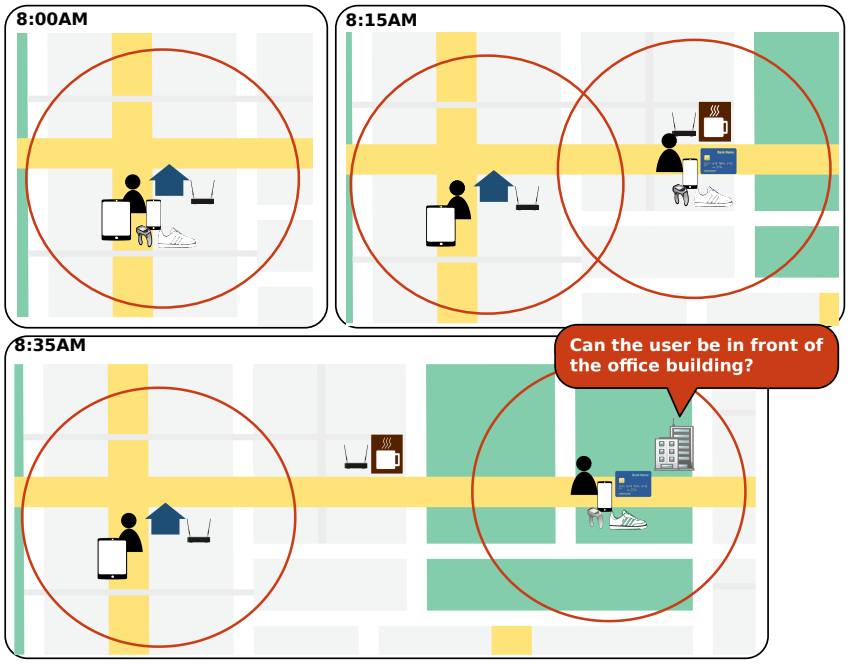

Fig. 2. Example scenario, where a user walks from his home to the office, stopping for a coffee.

still two Avatars, one at home and one on the street very close to the office location. Notice that the credit card previously used at the coffee shop lingers with the Avatar it was associated with when it was used. Our goal is to both identify that the Avatar located at home corresponds to idle devices and, as such, we are less confident it corresponds to the real user, as well as be more confident that the moving Avatar corresponds to the actual user. This way we can respond correctly to the query.

4 Icelus Architecture

We realize the approach presented in Sec. 3 with the Icelus system. Its architecture, which we present here, can incorporate the majority of IoT-devices that could be of use. Icelus organizes IoT devices into a hierarchy, where more powerful interconnected devices collect data from smaller devices, as depicted in Fig. 3. In turn, those devices send the information to a hub, which hosts the Icelus service.

4.1 The Icelus Hub

The brain of the system is a hub collecting information from various sources. We assume that the Hub is under the control of the user, so the data collected are never seen by third parties. We envision that it is hosted either in the cloud [42] or in a

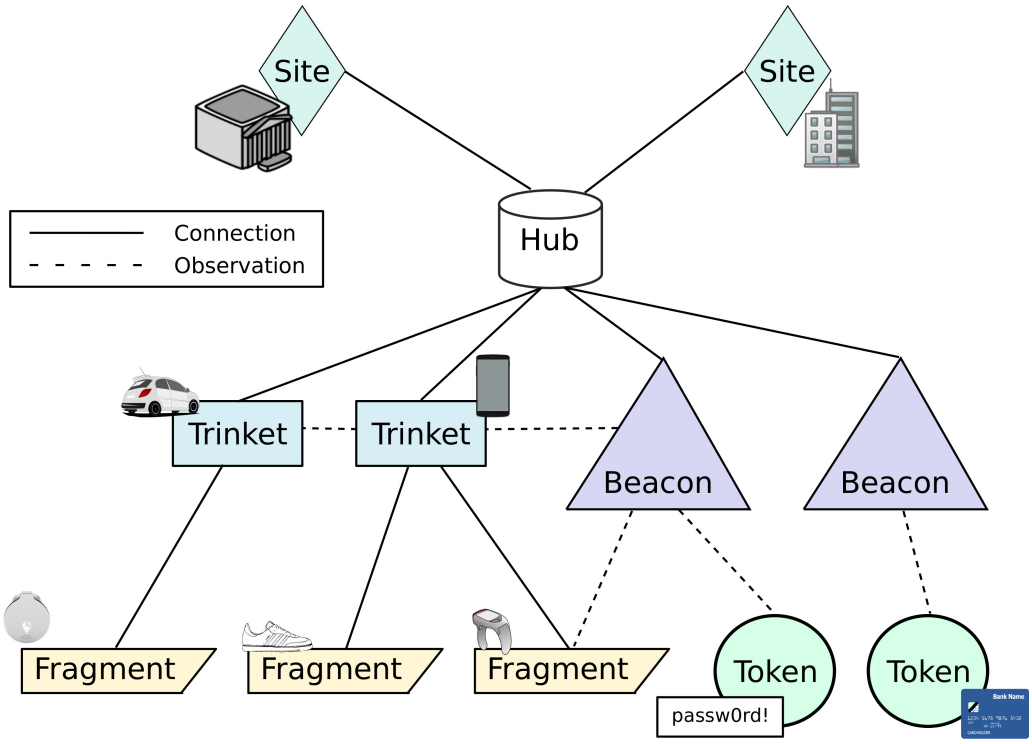

Fig. 3. Architecture overview.

smarthome hub device [30]. Hosting in the cloud provides us with all of its benefits and risks. We assume that the cloud provider is not malicious, however, it may be curious or compromised. In Sec. 6, we present a privacy-preserving extension that can alleviate such risks. Other approaches, such information-flow tracking [45] and SGX [54] are also applicable.

4.2 User-Owned Devices

Based on their intrinsic characteristics and communication capabilities, we classify the devices that can be part of Icelus into the classes described below:

– **Trinkets** are devices that can connect to the Internet directly (e.g., over WiFi, LAN, or 3G) or indirectly (e.g., by tethering through Bluetooth with a device that is connected). Such devices include smartphones, smartwatches, and even Internet-connected cars [16]. To join the system, Trinkets need to first register with the Hub, during which the two exchange their public keys. Thereafter, the Trinket uses its secret key to digitally sign all the information it reports and the Hub's public key to encrypt transmitted information.

– **Fragments** are similar to Trinkets, but cannot directly connect to the Internet. They are, however, able to connect to a Trinket directly (e.g., over Bluetooth). Such devices may include wrist wearables like a Fitbit, other smart wearables like shoes, and could even be in-body devices. A multitude of devices in the IoT are Fragments. Based on whether it is possible to load additional client software on them, they are more like Trinkets, in the sense that they can register with the Hub and sign their data, even though they still rely on a Trinket to relay their data to the Hub. If public-key cryptography cannot be supported, a shared secret key can be established with the Hub to use more lightweight data signing algorithms. On the other hand, other Fragments devices will rely on Trinkets for most operations. In the most restricted case, a Trinket may only be able to report that a Fragment is in the vicinity (e.g., BLE tags [4]).
– **Tokens** are devices that cannot actively connect to anything including things such as smartcards, magnetic identification cards (commonly referred to as swipe cards), RFID tags, etc. Tokens are passive and can only be observed through another device, commonly some short of a reader.

4.3 Beacons

Beacons are third-party devices, or even entire systems, able to report information about the whereabouts of a user or one of his devices. Reports can be generated after the user interacts with a Beacon or when it observes one of the user's devices. For example, the user *interacts* with a Beacon when entering valid authentication credentials at a physical terminal or when using a credit card at a POS. While, a Beacon *observes* the user's smartphone, when it authenticates with a wireless hotspot. In both cases, the location of Beacons is known. Services associated with Beacons must register with the Hub to be able to push observation information to it. However, it is also possible to extract information already available in other channels. For example, many banks transmit credit card usage reports through SMS or email, which can be used to locate Tokens like credit cards.

4.4 Avatars

An *Avatar* is a digital estimate of a user's location in the physical world. Each Trinket reporting geolocation information attempts to generate an Avatar at its location and attaches to it along with all its slave Fragments, so even the ones that are not with the user will create their own. Any devices in the same vicinity will be joined under one Avatar. We define the same vicinity to be as a circle with a center on the previous

Avatar location coordinates with a radius equal to 8 meters which is the *worst case* [61] accuracy of standard commercial GPS systems. When a Token appears, due to an observation from another device or a beacon, it is linked to the Avatar at the location of the report. Because Tokens appear only momentarily when they are used, they linger with the Avatar they are connected to, until a new report about them is received. Tokens appearing away from existing Avatars, create a new Avatar at the location where they were observed.

The **Confidence Score** of an Avatar represents the confidence of our system that it corresponds to the actual user. Our approach can support a variety of algorithms for calculating it. In Sec. 5, we present one such algorithm.

4.5 Servicing Location Queries - Sites

The entities that may query the system about whether it is possible for the user to be physically present at a location are referred to as *Sites*. A Site can be part of Icelus itself, because it is property of the user (e.g., the Trinkets corresponding to the user's home front door or car), or a third-party, such as the user's bank or employer. Sites need to be authorized by the user and registered with the Hub, before being allowed to issue queries, by *supplying the locations* they wish to place queries for.

The Hub listens for queries from Sites, which semantically follow the format: "Can the user be at location L?" When a query is received, it asks Trinkets to report with fresh data. The system can then wait for a bounded amount of time, collecting new reports and updating its model of possible user locations. Note that the model is continuously maintained independently of whether any requests have been made, by having devices report periodically or opportunistically. As a result, Icelus can always issue a response in a bounded amount of time, the only thing that changes is the freshness of the model used to make a decision, which could be milliseconds or few-minutes old.

The Hub then examines if an Avatar with confidence score higher, than the rejection threshold configured for the Site, exists in a location other than L. The threshold can also be a global setting. Our design is flexible; other factors can be introduced when responding to queries and a variety of policies can be implemented. For instance, instead of immediately responding negatively, Icelus can prompt users and ask them to authenticate on their devices that have such capabilities.

5 Avatar Confidence Score

An Avatar is a in reality a set of collocated devices in an area of a given radius, the Confidence Score of each Avatar is a quantity representing the probability that the user is physically "near" that set of devices.

In this section, we present an algorithm for calculating this likelihood, however, our design is not inherently bound with the presented Confidence formula.

Given its definition, an Avatars Confidence intuitively should be directly correlated with the number of devices in a given area where higher number (of colocated devices) should be positively correlated with a higher confidence score and a higher probability thus that a user is also present.

While our model is based on this intuition, we will see that there are additional factors in play, that significantly seem to affect the likelihood a user is present, other than simply the number of devices. The most prominent of which is user actions, that decisively identify idle or forgotten devices with devices that are in his presence. User *Activities* is a feature Icelus exploits and plays a central role to our Confidence calculation mechanism.

5.1 Device Credit

We call "Device Credit", the likelihood a user is colocated with a device at a given time and place.

In statistical terms, we are defining a probability where: *given that a device sends a report to Icelus, a user is also physically in the vicinity of that device*. More formally expressed, we are defining "Device Credit" to be the probability given by the following Bayesian formula:

$$P(U \mid D) = \frac{P(D \mid U) * P(U)}{P(D)} \text{ Where,} \tag{1}$$

- $P(U \mid D)$ Is what we are looking for, given that a device report is received by Icelus, what is the likelihood the user is collocated.
- $P(D \mid U)$ Formally, it is the probability that given a "sighting" of the user at any time, what is the expected probability he also has the device with him. In simpler terms: the probability the user is "carrying" the device.
- $P(U)$ The probability that if someone is operating the device it is the intended user.
- $P(D)$ How frequently the device is used and thus it is active and reporting. We used the survey to assign this value initially based on how often the user (believes)

	Sample Question
P(U)	Would you share your smartwatch with ...? [I do not share it][family][spouse]
	Would you share your smartphone with ..? [I do not share it][family][spouse]
P(D|U)	You carry your smartphone ..[I do not pay attention to its whereabouts][always]
	You wear your smartwatch ..[I do not pay attention to its whereabouts][always]
P(D)	You check or use your smartwatch ..[multiple times per hour] [rarely]
	You check or use your tablet .. [multiple times per hour] [rarely]

Table 1. Some of the survey questions, and the attributes they are associated with.

he checks or uses his device. However Icelus can calculate this value with higher accuracy over a period of time by simply counting the number of reports the device sends daily. For instance if the report window is 5 mins and the device is reporting in 144 out of 288 then $P(D) = 50\%$.

Since for novel devices such as the smart wearables and BLE sensors there are no studies we could find, that provide statistics such as what percentage of the time the user is near or using them (unlike the case for smartphone where it is studied extensively) we deployed a small user questionnaire in order to elicit the required elementary probabilities.

5.2 Questionnaire

We created an anonymous online questionnaire and disseminated it through various channels after obtaining IRB approval from our institution. Our goal was to obtain information about what kind of devices users own and how they use them in their day-to-day routine, we used responses to elicit elementary probabilities for our device weights.

Although our collected data to this point represent a population of 100 individuals, this survey does not have the statistical properties (demographic diversity or sufficient samples) to represent the general population. We used these weights as a starting point to evaluate the performance of our system. Our insight is that while these values are

adequate for our experiment, should our system be deployed they should be calibrated to each user during the device registration step. Since having a user complete a survey might be subject to both inaccuracies and potentially add to user frustration, Icelus could implicitly estimate device weights for each user by employing machine learning methods on the Hub; this however is beyond the scope of this work.

Max Credit We define as Max Credit the maximum credit a device may provide to the avatar it is attached to. Max Credit is equal to the value calculated from Equation 1.

Base and Activity Credit Since idle devices are generating false Avatars, erroneously leading Icelus to believe a user is in a different location, it is thus crucial that devices that actually follow the user to be able to out-weight those that are idle and at the same time idle devices should only be able to generate weak Avatars. In order to achieve both goals we separate its device's credit to two parts *Base* and *Activity* Credit.

Base Credit is a baseline value that the owner is colocated with this device just because it is powered on and reporting. A device is awarded Base Credit whenever it sends a report .

Activity Credit When the user performs activities on the device such as: unlocking the smartphone with a pin or the Fitbit detecting a heart-rate it is awarded extra credit.

Base and Activity credit together compose the Max Credit as defined previously. Activity Credit is assigned to each activity based on the evidence this activity provides that a user is present and it is the intended user, each software client is configured on a per device basis. For instance a smartphone that is able to perform fingerprint based authentication provides more credit than a simple pin authentication activity. Figure 4 attempts to illustrate composition of Credit clearly. `http://mathworld.wolfram.com/MovingAverage.html` *Moving Average [65]:* To smooth the erratic nature of user activities we introduce and evaluate for a variable number of past values an averaging window. For example for window of 1 if t is the current report window the new credit value is: $C_{New} = \frac{C(t-1)+C(t)}{2}$. A window size of 0 means we only consider the current estimated credit value.

5.3 Confidence Score Estimation

The aggregated credit of each device is the **Avatar's** confidence, and thus the likelihood that the owner is denoted by the cluster of devices composing this Avatar. We

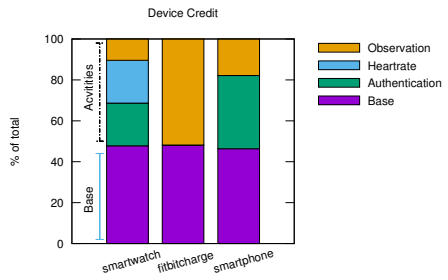

Fig. 4. Activity contribution example to device credit. Device credit is continuously calculated using Base credit plus credit from performed Activities.

also assume that the probability of each device being with the user is independent of others, and we mathematically model as an independent random variable. Based on this, the confidence score of an Avatar A with N attached devices is given by Equation 2. If there are more than one Avatars per user (e.g. the user has devices reporting from different locations) then each Avatar will have a Confidence score calculated from the devices under its area of influence.

$$A_{conf} = 1 - \prod_{i=1}^{N}(1 - C_i) \tag{2}$$

This can also be read as the *complementary probability of the event that the user is not near any of the devices attached to the Avatar*.

Rejection Threshold Since sets of devices in different locations lead to different Avatars, such as when idle devices are left home, we introduce a minimum confidence threshold to select only the Avatar that represents the devices "following" the user. We set this threshold to be equal to the *maximum* confidence an Avatar would produce if *all* registered devices are idle and attached to it. Since this is the maximum confidence possible achievable by any set of idle devices it will also satisfy the case of more than one idle Avatars.

Icelus blocks access to a Site *if* an Avatar is found with confidence strictly above this threshold. By following this decision policy we never falsely reject a transaction performed by the real user due to idle or powered off devices. As we will show in Section 8, our model is able to generate Avatars with sufficient confidence while filtering idle devices due to our activity mechanism.

Safe Zones Early results showed that the user is present at some locations while his devices are idle, such as being home during the night. This is reflected in our

confidence formula by adding extra credit when devices are detected to be in the users house during certain hours. This way there is a high confidence Avatar generated while the user is sleeping, protecting him from unauthorized accesses at different locations. In the current system version we annotate manually, the user specifies when he is at home. Our insight is that this can be learned automatically by Icelus but we do not evaluate it in the current work.

6 Privacy-Preserving 3rdParty Hub Hosting

Hosting a personal hub may be a challenge for many users. Alternatively, a third party can host a hub as a software-as-a-service. As highlighted previously, this may raise privacy concerns as this party may learn the positions of the user's devices at the time of protocol engagement.

We leverage *Secure Multi-party Computations (SMC)* to mitigate these privacy concerns, with the goal of allowing the hub to learn only distances between reported positions, and not the actual positions. For simplicity yet without loss of generality, we focus on the case of a single site (referred to as "the site").

6.1 Additively Homomorphic Encryption and Additive Blinding

SMC is an active area of research in cryptography including tracks on secret sharing [56], garbled circuits [66], or homomorphic encryption [51, 62]. Homomorphic encryption is suitable for arithmetic computations, which makes it our choice for dealing with Euclidean distances [22, 26]. For efficiency, we require an additively homomorphic encryption scheme such as the one provided by Paillier [44].

Additively homomorphic encryption schemes provides features as described by equations 3–5, which shows the three primary operations of addition, negation, and multiplication (with a known plaintext):

$$[\![m_1]\!] \oplus [\![m_2]\!] = [\![m_1 + m_2]\!] \tag{3}$$

$$\neg[\![m_1]\!] = \quad [\![-m_1]\!] \tag{4}$$

$$[\![m_1]\!] \odot m_2 = \; [\![m_1 \cdot m_2]\!] \tag{5}$$

For our purposes, let the plaintext space $\mathcal{M}$ be isomorphic to the group $(\mathbb{Z}_n, +)$ for some number n, and the public and private key be K and k respectively. For the scope of this paper there is only one such key-pair, for which only the site holds k but where K is known by all parties. For readability, the operations $\oplus, \odot, \neg$ described below do not have an explicit key associated to them, we assume they all use the usual

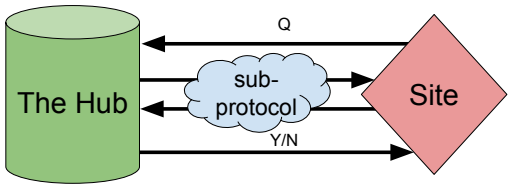

Fig. 5. Depiction of communication when a site queries a third-party hub

k, K pair. 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 $c_1 \ominus c_2$. For brevity, the encryption of a plaintext p using the key K is denoted as $[\![p]\!]$.

As a building block in our protocol, we will use the technique of *blinding*. A party A can blind a variable x by addition with a uniformly random value $b \in \mathcal{M}$ as $x' = x + b$. B cannot distinguish x' from a random sample in $\mathcal{M}$ but can return to A a value $x' + y$, from which A can compute $x' - b = x + y$.

6.2 Protocol outline

A user-owned hub can receive location information in the clear, and continuously update avatars. For a privacy-preserving third-party hosted hub, all location reports will arrive at the hub encrypted using K. When a query is made by the site, the hub will initiate a sub-protocol run together with the site. Through this sub-protocol, detailed in the following section, the hub is able to compute distances between any pair of locations. Holding the pairwise distance between three points, it is possible to calculate their relative positions. Thus, using the sub-protocol three times per location, the hub can calculate a full relative coordinate system for all locations. The setup is visualized in Figure 5.

If the hub needs several historical locations for a trinket, fragment, or token (e.g. to compensate for movement), it can cache them and calculate the relative positions retroactively.

6.3 Privacy-preserving distance calculations

There are several existing works on Euclidean distances using additively homomorphic encryption (e.g., [26, 67]). In most cases however, one of the two parties knows the coordinates of one of the two positions. In our case, neither the site nor the hub should learn any positions.

The hub needs to initiate the sub-protocol multiple times. For each invocation, the hub chooses two encrypted positions $([\![x_1]\!], [\![y_1]\!])$ and $([\![x_2]\!], [\![y_2]\!])$ and then runs the protocol for them.

The goal is to compute the squared distance together with the site as $(x_1 - x_2)^2 + (y_1 - y_2)^2$. This requires two roundtrips. The first one is due to the fact that the hub cannot compute a squaring in the ciphertexts. This will be done by requesting that the squaring is done by the site, in a blinded fashion. After the first roundtrip, the hub holds the encrypted squared distance, and will ask the site to decrypt it, again using blinding. Finally, the hub can compute the square root and arrive at the distance between the two points. The protocol follows as:

1. The hub computes $[\![x]\!] = [\![x_1]\!] \ominus [\![x_2]\!]$ and $[\![y]\!] = [\![y_1]\!] \ominus [\![y_2]\!]$ and creates a blinded version of each as $[\![x']\!] = [\![x]\!] \oplus [\![b_x]\!]$ and $[\![y']\!] = [\![y]\!] \oplus [\![b_y]\!]$. The hub caches b_x and b_y and sends $[\![x']\!]$ and $[\![y']\!]$ to the site.
2. The site decrypts $[\![x']\!]$ and $[\![y']\!]$, computes their squares, and sends $[\![x'^2]\!]$ and $[\![y'^2]\!]$ to the hub.
3. The hub derives $[\![x^2]\!] = [\![x'^2]\!] \ominus [\![2xb_x + b_x^2]\!]$, and $[\![y^2]\!] = [\![y'^2]\!] \ominus [\![2yb_y + b_y^2]\!]$
4. The hub then computes the encrypted squared distance $[\![d^2]\!]$ and sends a blinded ciphertext $[\![d']\!]$ to the site and caches the blinding as b_d.
5. The site decrypts d' and sends it to the hub.
6. The hub computes $d = \sqrt{d' - b_d}$.

We establish privacy guarantees by proving that only the distance between the devices can be learned by the hub and nothing else about the positions of the devices. The formal concepts and proofs are detailed in Appendix 1.

7 Implementation

We developed a prototype of Icelus including the Hub service and client software for Android smartphones and wearables. The prototype implementation is henceforth referred to as Icelus. The Hub service in Icelus is implemented as a web application deployed under JBoss AS 6.3. The Hub uses RESTfull services to receive reports from devices encrypted using public- or shared-key cryptography over HTTP connections. Icelus performs location modeling without using the privacy-preserving protocol presented in Sec. 6. Instead, we developed a separate proof-of-concept implementation that utilizes the privacy-preserving protocol to calculate distances between devices to evaluate its performance, which we plan to integrate in future implementations of the Hub.

Currently Icelus client software was created for smartphone and wearable devices running Android. We implemented two versions of the client, one for Trinkets and one for Fragments. The Trinket and Fragment clients were developed using the Android SDK v17 and v20, respectively, and communicate over Bluetooth using the Android-recommended messaging framework for hand held-to-wearable communication. The Trinket client is also able to monitor Fragments that do not feature client software, such as Fitbit devices, by passively monitoring the devices paired with the Trinket over Bluetooth .

Messages from The Hub to clients are sent over Google Cloud Messaging (GCM), while HTTP is used in the opposite direction except during the initial registration step that takes place over HTTPS. To register a device the user logs in to Icelus UI over HTTPS, and performs a standard two way registration step with the Icelus client, server-client keys are generated and exchanged. After this step all communication happens over HTTP.

The messages exchanged can be broken down to three parts: a header, which is composed by the device ID and a flag indicating whether the message body is encrypted, the body which includes timestamped sensor data, and a tail, where the digital signature of the header and body, produced using the private key of the sender, is placed. Sensor data can include information such as GPS data (coordinates, speed, bearing), step count (indicates activity), the SSID of the currently connected WiFi access point, list of paired Bluetooth devices, etc.. *Optimizations:* Clients can choose to omit certain data from reports when they determine that no significant change to their state has occurred. For example, when the device has not moved and is idle. The client will then send the equivalent of a heartbeat message that simply notifies the Hub that the device is active and at its previous location and state.

8 Evaluation

In this section, we present the evaluation of Icelus in terms of effectiveness in making authentication decisions and efficiency.

8.1 Effectiveness

To evaluate our approach, we performed two field studies having one of the paper's authors use Icelus. We hosted a Hub on Amazon's EC2 cloud and registered the following user devices: an Asus Zenfone 2 smartphone as a *Trinket*, a Samsung Gear Live smartwatch as a *Trinket*, a Fitbit Charge wrist wearable as *Fragment*, and a TrackR Bravo BLE tag attached on the user's keychain as a *Token*. Trinkets were set

to periodically report to the Hub every five minutes. We could not place code to control the Fitbit Fragment, and the TrackR Token is passive and can be observed by the Smartphone and the TrackR's crowdGPS service. We did not include any third-party services as Beacons.

In the first study, $S1$, we deployed Icelus using the smartphone, the Fitbit, and the TrackR and collected data over the course of one month. In the second study, $S2$, we deployed Icelus using the smartphone, the smartwatch, and the TrackR and collected data over the course of one workday.

Accuracy To test the accuracy of the decisions made by Icelus, we emulated query requests coming from the institute, which acts as a Site, whenever the magnetic id swipe card of the user was used to enter any of the access-controlled spaces in the institute. For example, this included the door to the user's office, the gym, etc. We obtained this data through the institute's IT department. In total, this included 49 accesses in 5 different card-protected doors in the first study, and 5 accesses in 3 doors in the second one. In all cases, the user was the one actual using the access card, so these data also correspond to the ground truth. We use the access-card and study $S1$ data to calculate the following authentication metrics:

- **False Reject Rate (FRR).** FRR is the rate of falsely denying access to the real user. It occurs when an Avatar above the rejection threshold is estimated to be *at a different location* from the one the Site is querying about, an access-controlled door in our case
- **Potential False Acceptance Rate (PFAR).** Since during our experiments there were no attempts to gain illegal access, PFAR represents the potential False Acceptance Rate (FAR) of Icelus. For calculating PFAR, we assume that an attacker continuously attempts to enter the institute. This means that the attacker has obtained the user's swipe card and attempts to enter the institute every five minutes. Since we are using a five minute period to update the Confidence Score a higher attempt frequency would not change anything. Hence, PFAR is the rate of falsely accepting such an ideal malicious user, because the Confidence Score of existing Avatars is below the rejection threshold.

Table 2 presents the results, when we employ a different window size in the moving average calculation of the Confidence Score. The FRR and PFAR are equal for a window size of one. Note that because we were not able to receive live queries from the swipe-card system, we relied strictly on the data periodically received by the Hub, that is, we could not request for fresh data from Trinkets.

Window size	**FRR** (# FR)	**PFAR**
0	6.12% (3)	8.16%
1	6.12% (3)	6.12%
2	4.08% (2)	8.16%
3	4.08% (2)	8.16%
4	4.08% (2)	10.20%
5	4.08% (2)	12.24%

Table 2. Accuracy of Icelus in field study $S1$ conducted over the period of a month. We report FRR and PFAR when using different window sizes in the moving average function in the calculation of Avatar Confidence Score. The total number of authentication requests was 49.

We also investigated the three false rejections of the system. Two of them occurred because of invalid data received from the user's smartphone. In detail, the user was driving to the office, a short drive of about 5 minutes. The smartphone reported once during the drive to the Hub, but failed to read its updated GPS location. That triggered a bug in our implementation that transmits the previous coordinates, if new coordinates cannot be read from the GPS, which also led to an invalid estimation of the user's speed. As a result, the Avatar remained at the previous location and its range did not increase. In a full deployment of Icelus, we would be able to contact the user's Trinkets to update location at the point of authentication and prevent such false rejections.

The third rejection occurred when the user forgot his smartphone when going to the gym, which is only a few minutes away from the office. As a result, the now forgotten smartphone, only reported that it is idle and no longer finds the user's Fitbit in the next 5-minute time window. We should note that such cases may not cause a huge inconvenience to the user, who only needs to walk a few minutes to retrieve the smartphone. We could argue that it is similar to forgetting one's keys.

Lessons Learned Besides correcting the buggy behavior in Icelus Trinket software, other actions that we are considering to address such issues is to enable devices to asynchronously report to the Hub, when a significant change in acceleration occurs. Immediately reporting Fragments that disappear is another option. In future work, we also plan to explore using learning to identify user habits for the same purpose. For instance, knowing that the user goes to the gym every afternoon could prevent errors.

Comparison with Smartphone-only LBA A smartphone-only location-based authentication (LBA) system [41] would accept the user correctly as long as the smartphone is with him, it is on, and it has Internet connectivity. Moreover, it may require

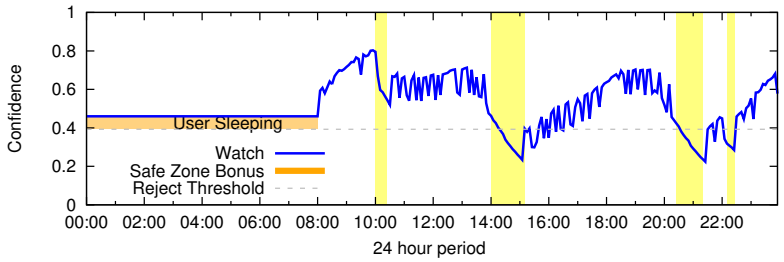

Fig. 6. Confidence plot when user carries a smartwatch and a Bravo tracker. Yellow rectangles denote no WiFi access.

interaction with the user. Using the smartphone data obtained from our field studies, we estimate the FRR for such a system to 8.1% (4 rejections). These occurred when the user forgot/left his phone at the office when going to the gym.

Assuming the user's smartphone has not been compromised, the smartphone-only approach does not have false acceptances. However, compromising the smartphone leads to an 100% FAR. *In contrast*, the true power of Icelus lies in numbers. For example, in study $S2$ the user has two Trinkets; if the smartphone is compromised by an adversary, the smartwatch and the remaining devices enable us to still reject the attacker.

Let us consider the following attack scenario. The adversary steals the user's smartphone at a coffee shop, which is also a wallet containing they access card. Alternatively, the adversary may have cloned the user's contactless access card earlier [17]. The user still carries TrackR's Bravo attached in his key chain and wearing his smartwatch, which is connected to the shop's WiFi. The adversary then attempts to enter the user's office, while the user is still enjoying his coffee.

In the case of smartphone-only LBA, the adversary has obtained all the tokens required to gain access. With Icelus, on the other hand, the adversary can only attempt to reduce confidence score of the user's Avatar to gain access. In Fig. 6, we plot the confidence of the user's Avatar in study $S2$ after removing the smartphone from him, that is, ignoring all data obtained through it. We notice that the Confidence Score of the user's Avatar remains over the rejection threshold for 80% of the day even without the smartphone, which would reject prospective attackers.

Simulation of Larger Scale To evaluate how our solution will fare in the future, when more devices are included, we conducted a simulation with a larger number of Trinkets, Fragments, and Tokens. Our goal is to compare the confidence score of Avatars corresponding to the legitimate user, and a set of devices that have been left

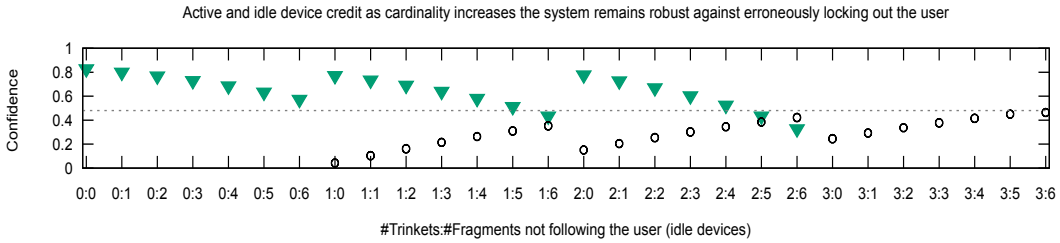

Fig. 7. Comparing the Confidence Score that can be achieved by Avatars corresponding to the legitimate user, and an unattended cluster of devices. Our system's robustness increases with higher device cardinality. Forcing the attacker to compromise multiple devices. The X axis shows the number of devices not carried by the user, formatted as *Trinkets #*:*Fragments #*.

unattended or compromised (e.g., forgotten at home). As before, the adversary may have compromised devices physically or virtually to reduce the Confidence Score of the user's Avatar. *As the number of devices of a certain type increases*, we also normalize the Max Credit of each device splitting it equally amongst all devices of the same type that the user registers.

We present the results of these simulations in Fig. 7 The y-axis in the plots measures Confidence Score, while the x-axis corresponds to the set or number of devices that are *not* carried by the user. For example, *2:5* corresponds to the scenario where the user is missing two of his Trinkets and 5 of his Fragments. As it is evident even when the user has under his control the minority of the devices our proposed mechanism produces an Avatar strong enough to differentiate him and answer inquiries about him with confidence while at the same time we vastly benefit from device number since as it is shown the attacker would have to compromise an ever increasing number of devices to achieve the same result.

8.2 Efficiency

We evaluate Icelus in terms of efficiency by measuring the response times and the impact of running the client on mobile devices in terms of battery consumption. We tested the following Android devices, which were not modified in any way other than adding our client app: (a) **3 smartphones**: an Asus Zenfone 2 with an Intel Atom Z3560 Quad-core CPU at 1.8GHz, a Samsung Galaxy S5 SM 900H with a Cortex-A15 Quad-core CPU at 1.9GHz, and a Samsung Galaxy S4 GT-I9500 with a Cortex-A15 Quad-core CPU at 1.6GHz, and (b) **1 smartwatch**: a Samsung Gear Live E42F with a Snapdragon 400 CPU at 1.2GHz.

Device	**Time and stddev.**	
	WiFi	3G
Devices with direct connection to the Hub		
Zenfone 2	170.8ms ±53.8	645.8ms ±280.8
Galaxy S5	173.7ms ±47.7	683.1ms ±300.1
Galaxy S4	172.9ms ±52.8	710.2ms ±290.9
Bluetooth devices tethered through Zenfone2		
Gear (as a Trinket)	280.8ms ±70.8	680.8ms ±283.8
Gear (as a Fragment)	310.8ms ±83.8	745.8ms ±330.8

Table 3. Device response time.

Device Response Time To evaluate the amount of time it takes for devices to report to the Hub, we conducted an experiment were devices submit 255-byte long reports to the Hub and timed the operations. We issued over 20,000 reports, where each report included data from available device sensors, their list of connected devices, and movement speed. Messages are padded to 255 bytes and encrypted with a 4096-bit RSA key before being sent.

Table 3 shows the average time and standard deviation in milliseconds to complete the operation when connected on our campus WiFi and over 3G. Note that the total time required for the Hub to request and receive a report include the time required to notify the devices, which in Android's case happens through the GCM service. So while previous works [41] have demonstrated low connection times, others have reported that they fluctuate when the notified device is offline [3]. The experiment shows that the process takes less than a second. Hence, even though not required, requesting new information from devices will not impose significant delays on authentication.

Reporting Period How frequently devices report to the Hub, affects battery consumption and the accuracy of the location information. In Table 4, we show the battery consumption imposed by our prototype client over a 10-hour period, when using different report windows. The last row shows the radius of the area where a user may have moved during the report window, assuming he is walking at 4Km/hr. Since energy consumption may change non-linearly depending on the battery's charge, we initiated all tests with fully charged devices. Battery levels were collected using system calls and utilities available on Android. Continuous messaging, appearing on the first column, pertains to sending requests to the server as fast as receiving the previous response. The results show that the effect on battery is small even when reporting

Period	Zenfone 2	Gear Live (as Trinket)	Gear Live (as Fragment)	Accuracy*
0s	7.1%	22.4%	10%	$\leqslant$ 5m
15s	1.5%	4.5%	3%	17m
30s	$\leqslant$ 0.8%	2.3%	1.3%	35m
60s	$\leqslant$ 0.4%	1.5%	$\leqslant$ 1%	70m
300s	$\leqslant$ 0.1%	1%	$\leqslant$ 1%	330m

*How accurately can we estimate location, assuming the user is walking

Table 4. Effects of reporting period in battery consumption and location accuracy.

frequently. As such, we expect that the major obstacle in submitting frequently will be lack of connectivity in certain environments.

Privacy-preservation We evaluated the performance of a privacy-preserving setup using our proof-of-concept implementation. As the main functionality of Icelus is independent on how the (relative) positions are provided, these benchmarks are carried out separately from the main server. It would not require significant engineering effort, apart from storing encrypted locations, to create a combined implementation.

For fragments, trinkets and beacons, the computational load is very similar to the encryption benchmarks presented in Section 8. For the site and the hub, this setup will incur a noticeable overhead. The experiments were conducted with requests corresponding to an example user with 8 devices, requiring 18 distance computations.

The experiments used the DGK encryption scheme [18] with a key-size of 2048 bits. The machines used was a normal workstation with an Intel i7-4790 CPU running at 3.60GHz with 16 GB RAM, and a MacBook pro with an 3.1 GHz Intel Core i7 CPU and 16 GB RAM. The experiments indicate that the time needed to complete the protocol is is about 2.2 seconds when executed between the university campus and a home network in the geographic vicinity, or about 0.81 seconds with minimalistic network delay where only the workstation was used. The rest of the time needed by the hub to compute the confidence score is identical to that of a non-privacy-preserving solution.

9 Related Work

Utilizing a user's location for authentication and augmenting security decisions, like deciding whether to permit raising a user's privileges, is conceptually described in

a position paper by Denning et al. [19]. Unlike this proposal, they propose an approach based on geodetic signatures, that is, signatures that tie a user or terminal to a particular physical location. The increasing popularity of mobile phones has lead to approaches that use them to establish user location and perform fraud detection. Park et al. [46] propose a mechanism where the bank sends a message to the user's phone when he performs a transaction, including the details of the transaction and the location of the POS.

More recently, Marforio et al. [41] use the trusted platform module (TPM) found on smartphones to sign GPS coordinates, preventing a compromised device from supplying forged location data. In contrast, this proposal is more robust as it uses the entire interconnected world instead of a single device to establish user location and augment authentication.

Location-based authentication has also been explored in the context of indoor "smart spaces" [7, 10, 27, 47, 64]. The common denominator of such systems is attempting to identify that a particular user is physically present in a room, usually through the use of proximity sensors, to protect an asset from unauthorized access and, less related to this proposal, customizing services (e.g., displays) according to user preferences. In the same area, the work of Al-Muhtadi et al. [7], which defines a context-aware security scheme for smart spaces, does include the notion of confidence that is affected by the various sensors being present in a space and the type of authentication a user performs. Unlike this proposal, research in this area focuses on indoor spaces and relies on infrastructure that can immediately detect a user entering one.

Also relevant to this paper are publications on predicting the user's location [9, 11, 15, 21, 36, 38, 49, 53], relying primarily on a single device and GPS, and implicit authentication based on learning the user's behavior patterns [33, 34, 50, 55, 57]. It should be noted that these methods focus on a *single device*, typically a smartphone or a portable computer.

Related to our location privacy-preserving protocol, there is large body work on location privacy [35, 59] and biometric authentication [22] where similar problems are investigated. Steps 1–3 are closest to the work of Erkin et al. [22] who perform blinded outsourced multiplication for privacy-preserving face recognition. Steps 4–6 are closest to a building block of the Louis protocol by Zhong et al. [67] to privately decrypt the result of a distance computation.

10 Conclusion

We have presented a new approach that uses the IoT to establish user location and use it as an additional factor of authentication. Location-based authentication has been explored before [31, 41], however, this work is the first to propose using something as pervasive as the IoT to locate users and model their movement. Undeniably, smartphones will still play an important part, as their proximity to their owner is high most of the time, but using the numerous devices that are part of the IoT, we *are* able to locate users more robustly. A major advantage of this approach is that, in the future, it can operate even more effectively, as more IoT devices become broadly available and integrated in the daily life of users. Today, our evaluation with readily available devices shows that our approach exhibits a low error rate and has negligible impact on the performance of the tested devices. Finally, collecting location information in a central location, even under the ownership of the user, must have undoubtedly raised concerns. We described a privacy-preserving protocol that can alleviate this concerns, and argue that progress in the field of privacy-preserving computation, as well as, information-flow tracking [45] and SGX [54] will further diminish the risks.

Acknowledgments This work was partly funded by the European Community under the ProSecuToR project and the Swedish research agencies SSF and VR.

References

1. Fingerprint hack. "`http://www.instructables.com/id/How-To-Fool-a-Fingerprint-Security-System-As-Easy-/`".
2. User perceptions of security, convenience and usability for ebanking authentication tokens. *Computers & Security*, 28(1–2):47 – 62, 2009.
3. Google cloud messaging (GCM): An evaluation. Metadata Blogspot, November 2014. `http://muratbuffalo.blogspot.com/2014/11/google-cloud-messaging-gcm-evaluation.html`.
4. Find your phone, keys, anything. Tile, September 30 2016. `https://www.thetileapp.com`.
5. 2015 LexisNexis Risk Solutions. Merchants contend with increasing fraud losses as remote channels prove especially challenging. True Cost of Fraud(SM) Study, September 2015. `https://www.lexisnexis.com/risk/downloads/assets/true-cost-of-fraud-2015-study.pdf`.
6. F. A, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2011.

7. J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas. Cerberus: a context-aware security scheme for smart spaces. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 489–496, 2003.
8. C. R. Alexandra Dmitrienko, Christopher Liebchen and A.-R. Sadeghi. When more becomes less: On the (in)security of mobile two-factor authentication. In *Proceedings of Financcial Cryptography and Data Security (FC)*, March 2014.
9. D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.
10. P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proceedings of IEEE INFOCOM*, March 2000.
11. P. Baumann, W. Kleiminger, and S. Santini. The influence of temporal and spatial features on the performance of next-place prediction algorithms. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 449–458, New York, NY, USA, 2013. ACM.
12. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.
13. J. Bonneau, S. Preibusch, and R. Anderson. A birthday present every eleven wallets? the security of customer-chosen banking PINs. In *Proceedings of the International Conference on Financial Cryptography and Data Security (FC)*, pages 25–40, 2012.
14. Chaos Computer Club (CCC). Chaos Computer Club breaks Apple TouchID. `http://ccc.de/en/updates/2013/ccc-breaks-apple-touchid`, September 2013.
15. Y. Chon, H. Shin, E. Talipov, and H. Cha. Evaluating mobility models for temporal prediction with high-granularity mobility data. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 206–212, 2012.
16. B. Cooley. Cadillac rolls out in-car Internet access. c|net, 2009. `http://www.cnet.com/news/cadillac-rolls-out-in-car-internet-access/`.
17. N. T. Courtois. The dark side of security by obscurity and cloning MiFare Classic rail and building passes anywhere, anytime. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, July 2009.
18. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for online auctions. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer, 2007.
19. D. E. Denning and P. F. MacDoran. Location-based authentication: Grounding cyberspace for better security. *Computer Fraud & Security*, 1996(2):12 – 16, 1996.
20. A. K. Dey, K. Wac, D. Ferreira, K. Tassini, J.-H. Hong, and J. Ramos. Getting closer: An empirical investigation of the proximity of user to their smart phones. In *Proceedings of*

the 13th International Conference on Ubiquitous Computing (UbiComp), pages 163–172, 2011.

21. T. M. T. Do and D. Gatica-Perez. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, 12:79–91, 2014.
22. 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, 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings*, volume 5672 of *Lecture Notes in Computer Science*, pages 235–253. Springer, 2009.
23. P. Fourez and Mastercard International Inc. Location controls on payment card transactions, patent no. WO/2011/022062. `http://patentscope.wipo.int/search/en/WO2011022062`, 2011.
24. A. Ghosh. Fraudsters steal $13m from over 1,400 ATMs in Japan in less than three hours. International Business Times, May 2016. `http://www.ibtimes.co.uk/hacker-group-steals-13m-over-1400-atms-japan-less-three-hours-1561435`.
25. O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
26. P. A. Hallgren, M. Ochoa, and A. Sabelfeld. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In A. A. Ghorbani, V. Torra, H. Hisil, A. Miri, A. Koltuksuz, J. Zhang, M. Sensoy, J. García-Alfaro, and I. Zincir, editors, *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, pages 1–6. IEEE, 2015.
27. J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, August 2001.
28. J. Huang. 60% still have old credit cards as oct. 1 EMV card deadline looms. USA TODAY, September 30 2015. `http://krebsonsecurity.com/2014/10/replay-attacks-spoof-chip-card-charges/`.
29. IDC. Always connected - how smartphones and social keep us engaged. IDC Research Report, Sponsored By Facebook. `http://www.nu.nl/files/IDC-Facebook%20Always%20Connected%20%281%29.pdf`.
30. Insteon. Insteon hub. `http://www.insteon.com/insteon-hub/`.
31. M. Jakobsson, E. Shi, P. Golle, and R. Chow. Implicit authentication for mobile devices. In *Proceedings of the 4th USENIX Conference on Hot Topics in Security (HotSec)*, 2009.
32. A. Kalamandeen, A. Scannell, E. de Lara, A. Sheth, and A. LaMarca. Ensemble: cooperative proximity-based authentication. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 331–344. ACM, 2010.
33. H. Khan, A. Atwater, and U. Hengartner. *17th International Symposium Research in Attacks, Intrusions and Defenses (RAID)*, chapter A Comparative Evaluation of Implicit Authentication Schemes, pages 255–275. Springer International Publishing, Cham, 2014.

34. H. Khan, A. Atwater, and U. Hengartner. Itus: An implicit authentication framework for android. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, pages 507–518, New York, NY, USA, 2014. ACM.
35. J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.
36. J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Ubiquitous Computing*, pages 243–260. Springer, 2006.
37. S. L. Lau and K. David. Movement recognition using the accelerometer in smartphones. In *Future Network and Mobile Summit*, pages 1–9, June 2010.
38. L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5):311–331, 2007.
39. Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008:197, 2008.
40. S. Mare, A. Molina-Markham, C. Cornelius, R. Peterson, and D. Kotz. ZEBRA: Zero-effort bilateral recurring authentication. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2012.
41. C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. Capkun. Smartphones as practical and secure location verification tokens for payments. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.
42. Microsoft. Event hubs. Microsoft Azure. `http://azure.microsoft.com/en-us/services/event-hubs/`.
43. D. Naccache, R. Géraud, H. Ferradi, and A. Tria. When organized crime applies academic results: a forensic analysis of an in-card listening device . *Journal of Cryptographic Engineering*, pages 1–11, Oct 2015.
44. 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.
45. V. Pappas, V. P. Kemerlis, A. Zavou, M. Polychronakis, and A. D. Keromytis. CloudFence: Data flow tracking as a cloud service. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, October 2013.
46. F. Park, C. Gangakhedkar, and P. Traynor. Leveraging cellular infrastructure to improve fraud prevention. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 350–359, December 2009.
47. N. B. Priyantha. *The Cricket Indoor Location System*. PhD thesis, MIT, 2005.
48. K. B. Rasmussen, M. Roeschlin, I. Martinovic, and G. Tsudik. Authentication using pulse-response biometrics. In *Proceedings of NDSS*, February 2014.
49. I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong. On the levy-walk nature of human mobility. *IEEE/ACM transactions on networking (TON)*, 19(3):630–643, 2011.

50. O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos. Progressive authentication: Deciding when to authenticate on mobile phones. In *USENIX Security Symposium*, pages 301–316, Bellevue, WA, 2012. USENIX.
51. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 32(4):169–178, 1978.
52. V. Roth, K. Richter, and R. Freidinger. A PIN-entry method resilient against shoulder surfing. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 236–245, 2004.
53. S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell. Nextplace: a spatio-temporal prediction framework for pervasive systems. In *Pervasive computing*, pages 152–169. Springer, 2011.
54. F. Schuster, M. Costa, C. Fournet, C. M. Peinado, G. Mainar-Ruiz, and M. Russinovich. VC3: Trustworthy data analytics in the cloud using SGX. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2015.
55. S. F. Shahandashti, R. Safavi-Naini, and N. A. Safa. Reconciling user privacy and implicit authentication for mobile devices. *Comput. Secur.*, 53(C):215–233, Sept. 2015.
56. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
57. E. Shi, Y. Niu, M. Jakobsson, and R. Chow. Implicit authentication through learning user behavior. In *Proceedings of the International Conference on Information Security (ISC)*, pages 99–113, 2011.
58. starbug. Fingerprint biometrics hacked again. Chaos Communication Congress (31C3), December 2014. `http://www.ccc.de/en/updates/2014/ursel`.
59. M. Terrovitis. Privacy preservation in the dissemination of location data. *SIGKDD Explorations*, 13(1):6–18, 2011.
60. The Telegraph – UK. Three quarters of cars stolen in France 'electronically hacked'. `http://www.telegraph.co.uk/news/worldnews/europe/france/11964140/Three-quarters-of-cars-stolen-in-France-electronically-hacked.html`, October 2015.
61. US AirForce. GPS Accuracy. "`http://www.gps.gov/systems/gps/performance/accuracy/`".
62. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. *IACR Cryptology ePrint Archive*, 2009:616, 2009.
63. R. Verdult, F. D. Garcia, and B. Ege. Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In *Supplement to the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 703–718, Washington, D.C., 2015.
64. R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, January 1992.
65. E. W. Weisstein. Moving Average from mathworld–a wolfram web resource.
66. A. C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

67. 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 Proof of Privacy-preservation

1.1 Background

In the following we recall briefly some fundamental concepts.

Definition 1 (Negligible functions). *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.*

Definition 2 (Indistinguishability). *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* $X \stackrel{c}{\equiv} Y$ *if for a probabilistic polynomial time (PPT) adversary* $\mathcal{A}$ *the function* $\delta(n)$ *is negligible:*

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

Definition 3 (Semantic Security). *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 8 is negligible. The game is won if an attacker can construct the procedures* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *such that* $b = b'$ *with non-negligible probability. If a cryptosystem is* IND-CPA *secure, it is also* multiple message IND-CPA *[12].* Multiple message IND-CPA *is formalized by the game* MM-IND-CPA *in Figure 8, where* k *is polynomially bounded by the security parameter. Essentially, this means that any any ciphertext or order of ciphertexts is computationally indistinguishable from their plaintexts.*

1.2 Privacy against semi-honest adversaries

Our privacy definition follows the standard definitions of secure multi-party computation in the semi-honest adversarial model [25,39], but is here simplified for the case

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

Fig. 8. IND-CPA and MM-IND-CPA

with two parties. For two parties A and B, where A has inputs $\overrightarrow{x}$ and B inputs $\overrightarrow{y}$, the framework formalizes the output of a protocol as $f(\overrightarrow{x}, \overrightarrow{y}) = (g(\overrightarrow{x}, \overrightarrow{y}), h(\overrightarrow{x}, \overrightarrow{y}))$. The function f is called the *functionality* of the protocol,

The functions g and h are functions describing all outputs presented to A and B from the execution of the protocol, respectively. f, giving the tuple of the parties' joint output, is called the *functionality* of the protocol.

Definition 4 (privacy). Privacy *within the standard framework (for deterministic functionalities) holds when the overall knowledge of each party after the execution of the protocol, called the party's* view*, can be computed from the inputs and outputs of that party. This is called that the view can be* simulated*. That is, for the two-party case with A and B as described above, one must show that:*

$$\{S_A(\overrightarrow{x}, g(\overrightarrow{x}, \overrightarrow{y}))\} \stackrel{c}{\equiv} \{view_A(\overrightarrow{x}, \overrightarrow{y})\}$$
$$\{S_B(\overrightarrow{y}, h(\overrightarrow{x}, \overrightarrow{y}))\} \stackrel{c}{\equiv} \{view_B(\overrightarrow{x}, \overrightarrow{y})\}$$

where S_A and S_B are the the simulators for A and B, respectively.

1.3 Instantiation for the proposed solution

For our purposes, let A be the hub, B be the site, and f be the functionality of the sub-protocol described in Section 6, g returns only the distance between the two points, and that h returns nothing. The explicit hub inputs for the hub are $p = (b_x, b_y, b_d, [\![x_1]\!], [\![y_1]\!], [\![x_2]\!], [\![y_2]\!])$ and the site has no explicit inputs. Both parties also have randomness as inputs for each encryption, but this is omitted here as it is trivial to simulate.

The protocol has two round-trips. During the first round-trip the site learns the blinded value $x' = x_1 - x_2 + b_x$ and $y' = y_1 - x_2 + b_y$, and the hub learns two ciphertexts. During the second round-trip the site learns the blinded $d' = d^2 + b_d$, and the hub learns d^2. The view of the two parties can be given as:

$$view_{hub} = (p, [\![x'^2]\!], [\![y'^2]\!], d')$$
$$view_{site} = (x', y', d')$$

Theorem 1. *The protocol privately discloses the distance for the hub according to Definition 4.*

Proof. To prove that Theorem 1 holds, we need to show that there exist two functions S_{hub} and S_{site} such that:

$$\{S_{hub}(p, g(p, \{\}))\} \stackrel{c}{\equiv} \{(p, [\![x'^2]\!], [\![y'^2]\!], d')\}$$
$$\{S_{site}(\{\}, h(p, \{\}))\} \stackrel{c}{\equiv} \{(x', y', d')\}$$

Blinded values are indistinguishable from a random sample in $\mathcal{M}$ when the blinding used is unknown. Thus, we define

$$S_{site} = (\alpha, \beta, \gamma)$$

where α, β, and γ are independent and uniformly random variables in $\mathcal{M}$.

Ciphertexts are easy to simulate for any semantically secure cryptographic system for any principal not holding the private key. In fact, for any list of plaintexts an arbitrary list of ciphertexts of the same length can be used, as per MM-IND-CPA in Definition 3. Since $g(p, \{\}) = d$, to create a simulation of d', one can simply use $d^2 + b_d$. The simulator for the hub is thus easily defined as

$$S_{hub}(p, g(p, \{\})) = (p, E(0), E(0), d^2 + b_d)$$

Paper **Five**

Privacy-Preserving Location-Proximity for Mobile Apps

Simonas Stirbys, Omar Abu Nabah, Per Hallgren and Andrei Sabelfeld

Location Based Services (LBS) have seen alarming privacy breaches in recent years. While there has been much recent progress by the research community on developing privacy-enhancing mechanisms for LBS, their evaluation has been often focused on the privacy guarantees, while the question of whether these mechanisms can be adopted by practical LBS applications has received limited attention. This paper studies the applicability of Privacy-Preserving Location Proximity (PPLP) protocols in the setting of mobile apps. We categorize popular location social apps and analyze the trade-offs of privacy and functionality with respect to PPLP enhancements. To investigate the practical performance trade-offs, we present an in-depth case study of an Android application that implements InnerCircle, a state-of-the-art protocol for privacy-preserving location proximity. This study indicates that the performance of the privacy-preserving application for coarse-grained precision is comparable to real applications with the same feature set.

Published in the 25th Euromicro International Conference on Parallel,
Distributed and Network-based Processing
PDP 2017, St. Petersburg, Russia, March 6-8, 2017

1 Introduction

Location Based Services (LBS) have seen a tremendous growth in recent years. A single resource lists over 2900 services at the time of writing [1]. The growth is boosted by the increasing spread of mobile devices, as Internet usage by mobile devices has come to dominate over desktop both by the number of users [2] and time spent [3]. Thanks to these developments, LBS-based mobile applications (or apps) have come to be a lucrative and thriving market.

LBS in mobile applications lets users accomplish a variety of tasks, such as planning a route from one location to another or obtaining information about entertainment venues in the vicinity. By obtaining the location of their users, LBS are able to provide a personalized experience to their users. Unfortunately, location disclosure endangers the privacy of the user, opening up for a plethora of attacks. These attacks are typically classified into external and internal.

The most intuitive kind of attacks are *external*, where the attacker has a black-box view of the system and can act as an ordinary user. External attacks have been seen in many widely used commercial applications such as Foursquare [4], Tinder [5] and Grindr [6]. These attacks often rely on *trilateration* techniques to precisely position users based on multiple distance queries. In these situations, the service provider is a Trusted Third Party (TTP) while the information being disclosed among users needs to be limited.

Secondly, *internal* attackers have full access to the system, which makes the typical internal attacker the LBS providers themselves. The smartphone app Uber, connecting passengers with private drivers, has been the subject of much privacy debate. Uber and its employees have been allegedly involved in privacy-violating activities from stalking journalists and VIPs to tracking one-night stands [7]. Given how powerful they are, internal attackers are significantly harder to protect against.

In an effort to mitigate attacks such as those mentioned above, there has been substantial progress on privacy-enhancing LBS [8–10]. Some approaches separate some parts of the system [11] in order to make it harder for an attacker to gain full white-box access to the service. For these approaches, the same foundational issue remains: the user needs to trust (a part of) the service, thus not addressing internal attacks. Other promising tracks have emerged using cryptographic techniques [12, 13] where the user retains control of their data through (homomorphic) public-key cryptography (homomorphic encryption is detailed further in Section 3).

While these studies address increasingly more powerful attackers, their evaluation has been often focused on the privacy guarantees. At the same time, the ques-

tion of whether these mechanisms can be adopted by practical LBS applications has received limited attention.

The focus of this paper is on the location proximity problem, i.e., the problem of computing whether one user is within a distance from another user. The privacy goal is to reveal the proximity and nothing else about the location of the users. Cryptographic approaches can provably protect against internal attackers, while disclosing only proximity mitigates the external attacker. Such solutions are referred to as *privacy-preserving location-proximity (PPLP)* [14–18, 12, 13, 19] protocols.

Note that different existing solutions protect different parts of user's data. Many approaches provide k-anonymity [11, 10], where the location of the user is indistinguishable among a set of users. The primary objective of these solutions is to protect the identity from the attacker, while allowing them to learn k distinct possible locations of the user. For the scope of this study, only solutions where the location of the user is secret are considered.

This paper studies the applicability of PPLP in the setting of mobile apps. We categorize popular location social apps and analyze the trade-offs of privacy and functionality with respect to PPLP enhancements.

To investigate the practical performance trade-offs, we present an in-depth case study of an Android application that implements InnerCircle [19], a state-of-the-art protocol for privacy-preserving location proximity. This study indicates that the features of PPLP fits several scenarios of real-world LBS and that the performance of such protocols is, for coarse-grained precision, comparable to real applications. To summarize the main contributions of the paper:

1. We evaluate to what extent a state-of-the-art protocol can be applied to mobile applications without limiting their functionality. The study uses popular location-based social apps from the Google Play Store.
2. We investigate performance trade-offs by performance measurements of an implementation of InnerCircle [19], a state-of-the-art privacy-preserving location-proximity protocol in an Android application. The study compares the performance of the implementation to real-world applications

The paper is organized as follows. Section 2 studies the applicability of privacy-preserving proximity-testing protocols to real-world LBS by investigating the privacy vs. functionality trade-offs. Section 3 presents necessary background for the InnerCircle protocol. Section 4 describes the architecture of the Android-based implementation of InnerCircle. Section 5 studies the performance of the protocol and compares it with the performance of the real-world apps. Section 6 discusses the related work. Section 7 offers concluding remarks.

2 Applicability

This section studies the applicability of privacy-preserving proximity-testing protocols to real-world LBS. First, a new set of *features* for LBS is outlined. These features can be used to assign an LBS into a *category*. A privacy-preserving protocol is able to serve a fixed set of categories.

2.1 LBS Features and Categories

We identify features and categories of location-based services in order to aid the applicability analysis of privacy-preserving protocols for LBS. First, mobile LBS applications vary based on whose location information they provide to the user, herein called the *target type* feature: venues, acquaintances, and strangers. Second, the applications also vary based on the precision of the location information provided, called the *precision* feature: exact location, precise distance, or a boolean proximity result. Using these application features we will determine to what extent a given privacy-preserving mechanism is applicable to each application.

Important to note about the venue *target type* is that in most cases, a venue's location is normally not secret. Thus, for most common applications, there is little to gain by using privacy-preserving protocols towards a venue, as in this case the instigator can be told the venue's coordinates and then run all computations locally. Although there is no need for a privacy-preserving protocol to handle the venue's position in this case, the privacy of the requester's location needs to be protected by the implementation as to prevent location leaks to internal attackers such as via the IP address.

We define three app categories: *Point-of-Interest based (PoI), Friend-Finding (FF),* and *People-Discovery (PD)* apps. PoI apps are common venue-locator applications, e.g. where people wish to meet each other or find a shop of some kind. Friend-Finding apps are for keeping track of the whereabouts of close friends and family. People-Discovery apps are for locating new people to interact with.

Table 1 reflects what kinds of applications belong to which category. As expected, the PoI applications disclose the precise location of the venue. This is also the case for Friend-Finding applications, though one could imagine scenarios where users would not want their precise location known even to friends and family, e.g. when buying a gift. Surprisingly, many applications that facilitate interaction between strangers also disclose the exact location of the users to each other.

For any privacy-preserving proximity-testing protocol to be adopted, the application must have proximity precision. Further, it cannot be a venue target type,

Table 1. Categorizations for Location-Based Services

Precision \ Target Type	Venues	Acquaintances	Strangers
Exact Location	PoI	FF	PD
Precise Distance	–	–	PD
Proximity Boolean	–	–	PD

as then the location can be publicized instead. The services which could most easily adopt privacy-enhancing technologies are thus the Friend-Finding and People-Discovery, both with proximity precision. However, some applications might in their current state reveal more location information than strictly necessary. As such, applicability of a privacy-preserving protocol is grouped into three classes:

- Not Applicable: the mechanism sets overly strict limits on the information disclosure to support the features of the application.
- Partly Applicable: to incorporate the mechanism the application would require minor modifications to its features but would still be able to maintain its core purpose.
- Applicable: the mechanism can easily be incorporated into the mobile application without hampering the functionality of the mobile application.

2.2 Real-World Applications

This section examines how real-world mobile applications utilize LBS to determine if their functions can be supported by a privacy-preserving protocol. We focus on analyzing popular applications, as indicated by top hits from searching for "location social networking" on Google Play Store. Examining the applications has revealed several trends, discussed below. Table 2 summarizes the results, but before looking at the results the different applications are outlined.

MeetUp a PoI application that allows users to find meet-up venues and meet new people with similar interests. The user is able to search for meet-up events by specifying a radius and receiving exact location of the venue. As highlighted earlier, PoI applications cannot easily adopt privacy-preserving technologies, making them Not Applicable.

FourSquare a PoI venue finding application which allows users to search for various entertainment venues, providing their exact location. PPLP can not be incorporated by FourSquare without significant changes in the application, and are Not Applicable.

Family Locator is another Friend-Finding application. It allows users to locate their family members and friends and see their exact location. Modifying the application to only display proximity boolean would go against the intent of the application and as such the PPLP is deemed Not Applicable.

Badoo and *Singles Around Me* are dating applications (PD) allowing users to find matches in their area, displaying their location on a map. In order to implement PPLP, they would need to forego this feature which would be a relatively important change in its functionality. Therefore PPLP is deemed Not Applicable for these application.

LINK is a PD app that allows users to connect with others and form groups based on interests. LINK provides its users with the exact distance between two strangers. Similarly to LINK, *SKOUT* enables strangers to search for others based on various criteria and displays to instigator the target's precise distance. Although such functionality is not directly supported by PPLP, a minor change in the application's features would fix the issue. Hence we deem PPLP to be Partly Applicable for these applications.

MeetMe is a People-Discovery application which enables their users to chat with strangers with similar interests in their area. *Tagged* is another People-Discovery app allowing users to find and chat with people in vicinity. In both cases, proximity checks occur mainly between strangers and only proximity boolean is revealed to the users, maintaining all other location information concealed. As these features are well within the limits of PPLP, the protocol is deemed to be Applicable to these to applications.

Table 2. Applicability of PPLP to Popular Mobile Applications

Application	Category	Not Applicable	Partly Applicable	Fully Applicable
MeetUp	PoI	X		
FourSquare	PoI	X		
Family Locator	FF	X		
Badoo	PD	X		
Singles Around Me	PD	X		
LINK	PD		X	
SKOUT	PD		X	
MeetMe	PD			X
Tagged	PD			X

As visible in Table 2, PPLP protocols are of limited use to applications which focus on interaction between people who already know each other as such applications place less importance on location privacy and allow their users to see the precise location of different users on a map. On the other hand, PPLP are very promising for applications that facilitate interaction with strangers before a possible meeting in reality. In such cases proximity between users is important as users want to know in advance if the potential meeting is possible, but at the same time want to keep their location private as they have not built enough trust yet to reveal it. As search on Google Play Store shows that such applications are relatively popular, good location privacy-preserving mechanism applicability means such mechanisms can become quite important in the future of the mobile application market.

3 A Concrete PPLP Protocol

Hitherto, we have only touched upon PPLP protocols in general; the enforcement of such is discussed in this section. There are many published works describing how to accomplish different flavors of PPLP [14–18, 12, 13, 19]. In this work, *InnerCircle* by AnonymousAthors [19] was implemented to evaluate efficiency of a recent PPLP protocol on a smartphone device. InnerCircle is a good representative of state-of-the-art PPLPs as it provides protection against internal attackers while disclosing only location proximity, which is a good countermeasure against external attackers. Further, the authors provide evidence that the protocol could be efficient enough for usage in smartphone applications. Other protocols, such as the work by Sedenka et al. [13] provides the same security guarantees, but requires the use of multiple cryptographic schemes and has several additional round-trips between the parties as compared to InnerCircle. Reducing the number of round-trips proved a good choice, as this can cause a blow-up in communication, as seen in Section 4.

The mobile application produced in this work uses the InnerCircle protocol [19]. This particular protocol was chosen as it preserves location privacy against both internal and external attackers, while completing in a single round-trip. The key concept used in InnerCircle is, as mentioned previously, homomorphic encryption, which avoids the need for TTP. In recent years homomorphic encryption has became a popular choice for creating privacy preserving protocols and as such, it is a good representation of much of the state-of-the-art technology in location-privacy.

The protocol considers two principals, Alice and Bob, where Alice is the instigator. When Alice wants to query Bob to check if they are in each other's proximity, Alice constructs a *location request*. The location request encapsulates Alice's coor-

dinates, encrypted under her public key. Bob uses the information in the location request together with his own coordinates to create a *location response*. A location response is an array which encodes a single boolean value, which can be decoded using Alice's private key. For the full protocol, the reader is referred to the original paper [19]. For the scope of this work, it suffices to view the protocol as consisting of three steps: request construction, response construction to encode the boolean result, and response interpretation to decode the boolean. The encoding step which constructs the lesser than comparison is henceforth referred to as `lessThan()`, while the decoding step where Alice finds out whether Bob is in her proximity is called `inProx()`. As shown in Section 5, the `lessThan()` and `inProx()` methods are the more time-consuming operations in the protocol.

The key concept used in InnerCircle is, as mentioned previously, homomorphic encryption, which avoids the need for TTP and in recent years has became a popular choice for creating privacy preserving protocols [14, 20, 21, 13, 19]. Homomorphic encryption allows for computations to be evaluated on encrypted data. Formally, given the plaintext space $\mathcal{M}$ and the ciphertext space of a homomorphic scheme $\mathcal{C}$ such that encryption is a function $E : \mathcal{M} \to \mathcal{C}$ and decryption is $D : \mathcal{C} \to \mathcal{M}$, for any arithmetic formula $f : \mathcal{M}^k \to \mathcal{M}^k$ it is possible to construct $g : \mathcal{C}^k \to \mathcal{C}^k$ such that $D(g(E(\overrightarrow{m}))) = f(\overrightarrow{m})$. I.e., for any arithmetic formula in the plain it is possible to construct another formula to compute the same in the ciphertexts. There are several flavors of homomorphic encryption. Normally homomorphic encryption signifies Fully Homomorphic Encryption (FHE) schemes [22, 23]. FHE schemes are extremely powerful, and can evaluate any formula as described above, but are rather inefficient. On the other hand, there are schemes that are more limited in what they can compute – such as Additively Homomorphic Encryption (AHE) – but which are far more efficient [24]. The Authors Of [19] present several cryptosystems which can be used to instantiate InnerCircle. In this research we have chosen to use the ElGamal's [25] encryption system using 1024 bit keys since it had a notably fast performance in the original implementation.

Of interest is also how an array is used in InnerCircle to encode a boolean. Of course, using an array requires much more communication, memory and computational resources. However, due to the limitations of AHE, the authors of InnerCircle found this the most efficient approach. In essence, the array a is the result of a less-than operation. To check if $x < y$, one can check if $\exists y_i < y : x - y_i = 0$. The protocol creates the array such that it contains only uniformly random numbers, except for the case when $x < y$, when it contains a single (random) slot which contains the encryption of 0. The decoding step is thus to decrypt the array and check

for the existence of a zero. Further, as square roots can not be computed using homomorphic encryption, the square of the distance between Alice and Bob is compared to the square of the radius, which yields an array which is quadratic in r.

4 System Overview

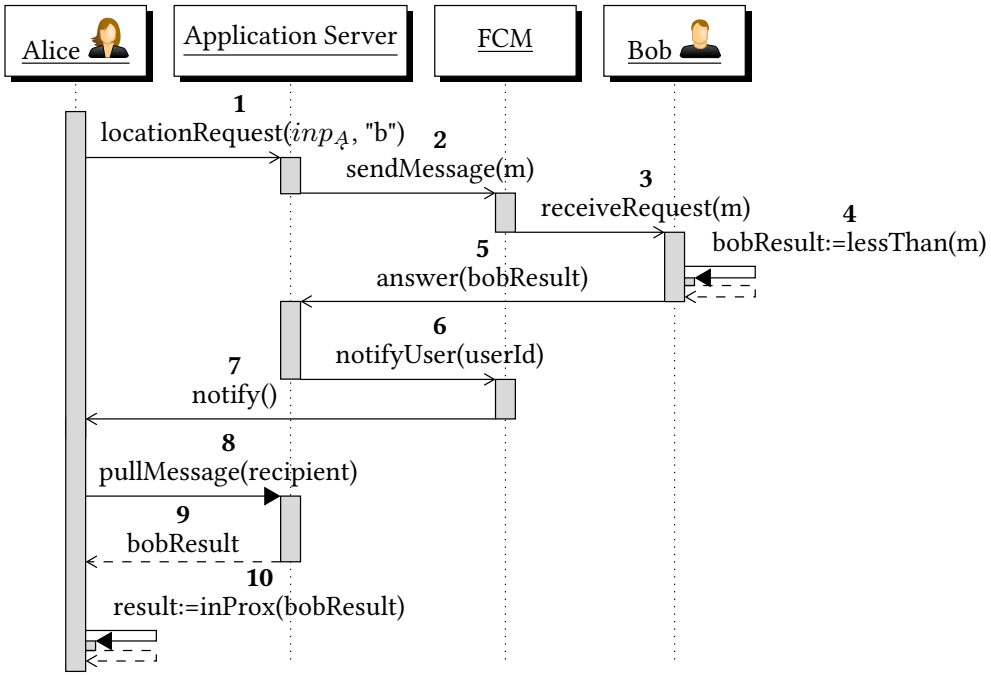

Fig. 1. Sequence diagram of a communication request

This section details the full system resulting from the implementation effort. In brief, the system consists of a mobile application for the Android operating system, an application server, and the messaging service used to send push notifications to the mobiles. Though network attackers are not considered in this work, such can easily be thwarted using HTTPS connections between all parties.

The original InnerCircle protocol finishes in a single round trip. As smartphones do not communicate in a peer-to-peer fashion the resulting implementation uses a larger protocol. The devices send messages to each other via push notifications, e.g. Firebase Cloud Messaging (FCM) on Android or Apple Push Notification Service (APSN) on iOS devices. Further, some of the messages are much too large to be sent via the FCM push service, which limits the size of the messages to 4 kilobytes. To send larger messages, a combination of the application server and the FCM service was used, where the FCM services were used to notify when a message is available

for on the server. This results in a protocol using 7 messages rather than 2 as intended by the original proposal. This communication overhead is much higher than the original prototype implementation by AnonymousAthors [19].

The resulting application thus involves communication between two clients and two servers. The clients are herein called Alice and Bob as in the original protocol, and the two servers the Application Server, and FCM. Messages exchanged during a single request are shown in Figure 1. Alice generates a location request which is sent to the Application Server (1), then from the Application Server to FCM (2), and from FCM to Bob (3). Bob then creates a proximity result using `lessThan()` (4), which is sent directly to the Application Server (5). The server notifies FCM (6), which in turn notifies Alice that the answer is ready to be retrieved (7). Alice then fetches the answer from the Application Server (8 & 9). When Alice retrieves the answer, she interprets ti using `inProx()` which will tell her if Bob is in her proximity (9).

InnerCircle assumes a euclidean plane, but the Android GPS interface provides longitude and latitude which have to be converted into Cartesian coordinates. The search radius and the coordinates input to InnerCircle are specified in an arbitrary distance unit. Thus, converting from GPS to the distance unit allows for discretization to take place, and the unit of the resulting Cartesian coordinate system can correspond to a millimeter, a yard, a meter or a kilometer. As the performance of InnerCircle is proportional to the radius, this is a useful tool to trade precision for efficiency.

5 Performance

This section presents the performance benchmarks performed in this study. First follows a brief description of the setup, after which efficiency both in terms of CPU and network usage is presented and discussed.

Testing was performed on two smartphone devices connected to a WiFi network, with the application server hosted on the same network. Each device is able to act as Alice as well as Bob to measure how the different phones handle both roles of the protocol. The devices used were two Samsung Galaxy S6 (model SM-G920F). The model was released in April 2015 and has a 64-bit Exynos 7 Octa 7420 system-on-chip, consisting of four 2.1 GHz Cortex-A57 cores, and four 1.5 GHz Cortex-A53 cores, and 3 GB of LPDDR4 RAM.

The protocol was executed 25 times with radius 25, 50, 75 and 100. The outcome total time taken to execute a request was measured as well as the CPU time spent computing by each party. CPU time measures the cryptographic parts of the proto-

col, with the two larger parts being `lessThan()` and `inProx()`. Encryption of Alice's coordinates and distance computation by Bob are included in the total CPU time but not displayed individually, as they are both relatively quick methods. Total Time represents the amount of time taken for the application to display the results to the user and includes the CPU time of both users as well as the network delay. Measurement starts when the user presses the locate button, and ends when the answer is available on the user's phone. The source code for both the application server and the smartphone app has been made publicly available[1].

5.1 CPU Performance

Table 3. Benchmark results (in milliseconds)

Radius	Total Time	CPU time `inProx()`	CPU time `lessThan()`	CPU time Total
25	4318.8	1500.2	1542.1	3124.1
50	11508.1	4978.0	5114.5	10174.2
75	21911.0	10235.4	10061.2	20376.4
100	35736.5	17355.1	16453.1	33887.0

The results of the benchmarks can be seen in Table 3, which shows the average time consumed for the collected data. As seen from Table 3, the time for protocol executions increases drastically with increase in radius. Mapping the average protocol execution times onto a line chart in Figure 2 suggests that the increase in time is more than linear, though less than the expected quadratic increase due to optimizations detailed in the original paper [19].

Table 4. Efficiency Results for Real Applications in Seconds

MeetUp	Foursquare	Badoo	Singles Around Me	LINK	SKOUT	MeetMe	Tagged
1.73	1.65	2.9	7.34	2.14	1.77	2.19	3.17

The performance of the Google Play applications was measured using a stopwatch, starting from the moment user opts to check location proximity to the moment the application displays its results to the user. The applications were tested 10

[1] https://bitbucket.org/innercircleandroid/

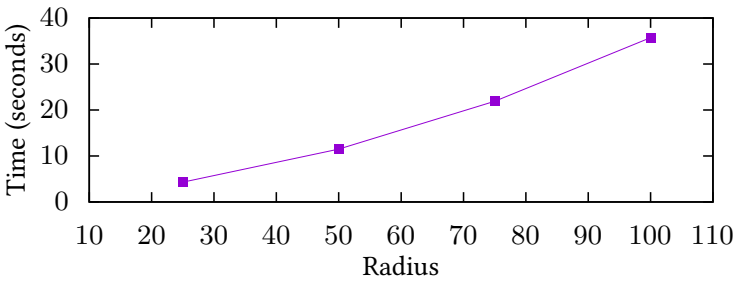

Fig. 2. Averages of execution times for protocol in different ranges

times each. The results can be seen in Table 4. These benchmarks can be compared to the total time in Table 3, allowing to determine if the efficiency of the protocol is sufficient enough to be successful in the market of mobile applications.

5.2 Network Usage

Additionally, network usage of the protocol is measured. Network usage is relevant for two reasons. First, users are often out of range of free network access, which means network usage has a financial cost. Secondly, SMC solutions, such as homomorphic encryption, are often limited either by computational or communications resources. Determining which is the limiting factor of the protocol is vital contribution of this work. Network usage is measured by testing the protocol 10 times at distances of 25, 50, 75 and 100 units and recording Bob's answer size in bytes.

Table 5. Average network usage in kilobytes

25	50	75	100
143.242	489.307	1025.198	1740.382

The protocol on average used 143 and 1740 kilobytes when transferring messages for 25 and 100 distance unit proximity checks respectively. The results of network usage can be seen in Table 5. As real applications would transfer unencrypted coordinates, network usage would be minimal and irrelevant. As such, network usage of real applications was not measured.

5.3 Discussion

The results shows the protocol yields an answer within 36 seconds for a radius of 100. Narayanan et al. [12] implemented a novel protocol in an Android application with an execution time of 46 seconds. Their results are comparable to our implementation, although their solution would likely perform better for larger radiuses. However, time frames over half a minute are unacceptable for most practical scenarios. Furthermore, it is reasonable to assume that Bob will not always be stationary in his position while the answer is generated on his phone. Given that on foot, at 5 km/h, it takes 72 seconds to cover a distance of 100 meters (more so by transportation vehicle), it is plausible that Bob's position can change to in/out of range before Alice receives an answer, invalidating the result.

On the other hand, using InnerCircle with smaller radiuses is definitely feasible in a mobile application; it took only 4.3 seconds to calculate proximity for proximity requests for a radius of 25 units. This is within the time frames of real applications, which provide output to the users in the range of 1 to 7 seconds. However, it is unlikely that Google Play application's response times are affected by the radius specified by the users. The majority of mobile applications focus on rather large distances. Thus, the distance unit used by InnerCircle must be tweaked to accommodate these, and should not be chosen as, for instance, 1 meter. Even though an implementation of InnerCircle would fall short of practical applicability if both high precision and large radius are needed, our study shows that state-of-the-art location-proximity protocols are efficient enough for integration in most mobile applications with more restricted precision than the applications are currently using.

As a concrete example, consider using InnerCircle to enhance the privacy of the previously mentioned Tagged application. Tagged allows users to check for other users within 100 kilometers. To utilize InnerCircle at such distances, the application would need to use a distance unit of approximately 4 kilometers to be able to use a radius of 25. The expected time for a proximity check would be only 4.3 seconds, which is comparable to the current time of the Tagged application.

The negative side of using a discretized plane where the unit is rather large is that it introduces an imprecise edge at the circle which denotes Bob's proximity to Alice. However, this error is fairly small relative to the proximity radius. Using the previous example, Alice is able to check the proximity of Bob in the radius of approximately 100 kilometers with the error range being 4 kilometers, which is an error of 4%. Although current applications most likely are checking proximity with higher accuracy at the same range, we believe the significance of the error is small in comparison to the radius.

In regards to network usage, the proximity result is at most 1.7 megabytes for proximity checks up to 100 distance units. Such sizes are comparable to average size of images, which the users rarely take into consideration when browsing Internet on their mobile devices. As such, we believe the protocol's network usage would be insignificant when incorporated into an application. Users could check proximity many times without much concern. This speaks favorably towards the protocol's implementation in mobile applications.

6 Related Work

The relevance of research on location-privacy has seen some debate, with studies showing mixed results on whether location-privacy is important to users of LBS or not. Barkhuus and Dey [26] compared the two scenarios of location-tracking services and location-aware services and performed an experimental case study with 16 participants. The participants had more privacy concerns regarding location tracking services compared to location-aware services, but in general were not overly concerned about privacy of their location data. Nevertheless, Barkhuus and Dey recommended focusing on developing services around location-aware concept. In case of location-tracking services, the researchers believe such services can still be acceptable as long as users have the option to turn-off the tracking capability at any time.

Xu and Gupta [27] developed a model to examine the impact of privacy concerns on intention to use LBS. They found that performance expectancy had a positive impact on participants' intention to use LBS and effort expectancy was positive only for inexperienced users, but privacy concerns had no direct effect. Interestingly, privacy concerns negatively impact performance and effort expectancy, thus indirectly affecting user decision to use LBS mobile services. This implies that privacy concerns are relevant to at least a limited extent to user of LBS applications.

Zickuhr [28] studied use of LBS in mobile apps by Americans. The findings show that the use of such applications is rapidly growing, from 55% of smartphone owners using LBS applications in 2011 to 74% of smartphone owners using LBS in 2012. Furthermore, taking into account that smartphone ownership itself quickly grew from 35% of adults in 2011 to 46% in 2012, it is safe to assume that the importance of LBS privacy concerns, even if relevance is currently debatable, will grow in the coming years.

6.1 Privacy-Preserving Technologies

There is significant literature on both protecting users' privacy against internal and external attackers. For internal attackers, there are several generic techniques not tied to LBS. For external attackers on LBS, there are two core tracks [29]. The first idea is to minimize each individual disclosure. For instance, by disclosing distances instead of positions, etc. Secondly, a good countermeasure is to discretize the location data by dividing the plane into a grid, such that many coordinates in a grid-cell are mapped to the same location. The grid cells need to be large enough that the imprecision is sufficient to provide privacy. While the first often allows a service to remain unchanged while providing better protection, the second can provide strict guarantees of how much information the attacker is able to learn.

Generic Privacy-Preservation For internal attackers, there are a number of different techniques. One popular strategy is the "k-anonymity model" [30–32], which hides the user among similar other users. This makes the original user indistinguishable from the rest of the population and thus anonymous. However, all efficient techniques for k-anonymity require a third party to be set up, which again opens up for internal attackers at this new party.

A generic approach to hide sensitive data from service providers is to utilize Secure Multi-party Computation (SMC), which is a research field of considerable size. SMC enables multiple parties to compute on private data without revealing their inputs. The ability to compute functions without revealing inputs allows for private data to remain confidential while being handled by 3rd parties, which completely removes the need for trusting a third party. There are three tracks in the literature that achieve SMC, each with its own large community: Secret Sharing (SS) [33], Garbled Circuits (GC) [34] and Homomorphic Encryption (HE) [22]. SS-based techniques show very promising performance, and are seeing some commercial use [35]. However, they typically require a set of non-colluding servers, which makes it unsuitable when the goal is to not put any trust in the service provider(s). Techniques based on GC have seen promising performance utilizing the Intel Advanced Encryption Standard Instructions through protocols tailored for this particular instruction set. As most mobile devices use ARM processors, it is unlikely that the performance results can be extrapolated to mobile devices. Further, GC offer a one-off solution, where any results (except the output) should not be reused in further computations.

As previously detailed, homomorphic encryption makes it possible to perform mathematical operations on encrypted data. The ground-breaking result by Gentry [23] presented the first Fully Homomorphic Encryption (FHE) scheme, which

is capable of computing arbitrary arithmetic formulas. Following Gentry's work, there's been numerous improvements for FHE [36–38]. However, so far there is no FHE scheme that is comparable in efficiency to schemes that are just additive or multiplicative. There have been many works that utilize homomorphic encryption to create privacy-preserving protocols in areas such as location-privacy and biometric authentication [20, 13, 19, 21, 14].

Privacy-Preservation in LBS Puttaswamy et al. [39] present a new technique for location privacy by coordinate system transformations, called LocX. Each user has a secret for which it's coordinate system is translated, and a set of friends. The secrets are distributed to each user's friends, such that only the user's friends may understand how coordinates are mapped. A prototype has been developed and it showed that it can be used in commercial applications with minimum overhead. However, unlike other protocols mentioned in this section, the user's exact location is revealed to all users with the secret, which forces the users to limit their social circle to users they trust with their location.

Further, to generate dummy data and present this to the LBS is a viable option to hide the user's location. Zhou et al. [40] propose a system called TISSA. TISSA allows users to choose what data an application can access. In case an application demands access to data that the user is unwilling to provide, the system sends dummy data as substitute, keeping the real data private. The system was tested in Android OS and successfully prevented leakage of information to restricted applications and caused no significant slow down to performance of the phone. However, using only dummy inevitably prevents the application from functioning properly. Kido et al. [41] propose a system which sends LBS providers real user data as mixed with dummy data. As the LBS providers cannot distinguish between real and fake data, the anonymity of the user is preserved. However, the solution causes large communication overhead as all users need to send many additional messages with dummy data for each real query.

There are not many works that provide an in-depth discussion of PPLP on Mobile Devices. Narayanan et al. [12] provides use cases where LBS mobile applications could be used and how their proposed protocol would relate to such applications. However, it is debatable whether the use cases themselves are realistic examples of LBS use and sufficient proof that the protocol could be applicable enough to be used in general applications.

7 Conclusion

This study furthers the knowledge of how well currently existing cryptographic privacy-preserving protocols apply to real-world mobile apps. To this end, we have road-mapped popular location-based social maps and identified scenarios where privacy-preserving location proximity is desired. The category of People-Discovery apps turns out to be a particularly promising fit. We conclude that the protocol can be fruitfully applied to a number of popular applications, in particular for the ones that facilitate meetings in real life between strangers, such as meet-up and dating apps.

Further, we have implemented InnerCircle, a state-of-the-art privacy-preserving location proximity protocol and integrated it in an Android app. With respect to performance, we arrive at the conclusion that InnerCircle on Android matches real applications at radius values of 25 and 50 units while values at 75 units and above are not yet within a reach. The average network usage for 25 units is 143 KB and for 100 units is 1740 KB respectively. With less precise coordinates the protocol can check the radius of 100 kilometers, using 25 unit radius in only 4 seconds, which shows that the protocol is efficient enough for implementation in real applications.

Acknowledgments This work was partly funded by the European Community under the ProSecuToR project and the Swedish research agency VR.

References

1. AngelList, “Location based services startups,” Sep. 2014, https://angel.co/location-based-services.
2. A. Lella, “Number of Mobile-Only Internet Users Now Exceeds Desktop-Only in the U.S.” https://www.comscore.com/Insights/Blog/Number-of-Mobile-Only-Internet-Users-Now-Exceeds-Desktop-Only-in-the-U.S, Apr. 2015.
3. K. Dreyer, “Mobile Internet Usage Skyrockets in Past 4 Years to Overtake Desktop as Most Used Digital Platform,” http://www.comscore.com/Insights/Blog/Mobile-Internet-Usage-Skyrockets-in-Past-4-Years-to-Overtake-Desktop-as-Most-Used-Digital-Platform, Apr. 2015.
4. D. Coldewey, “"Girls Around Me" Creeper App Just Might Get People To Pay Attention To Privacy Settings,” http://techcrunch.com/2012/03/30/girls-around-me-creeper-app-just-might-get-people-to-pay-attention-to-privacy-settings/, Mar. 2012.
5. M. Veytsman, “How i was able to track the location of any tinder user,” http://blog.includesecurity.com/2014/02/how-i-was-able-to-track-location-of-any.html, Feb. 2014.

6. C. Paton, "Grindr urges LGBT community to hide their identities as Egypt persecutes nation's gay community," http://www.independent.co.uk/news/world/africa/grindr-urges-lgbt-community-to-hide-their-identities-as-egypt-persecutes-nations-gay-community-9757652.html, Sep. 2014.
7. C. Bessette, "Does Uber Even Deserve Our Trust?" http://www.forbes.com/sites/chanellebessette/2014/11/25/does-uber-even-deserve-our-trust/, Nov. 2014.
8. J. Krumm, "A survey of computational location privacy," *Personal and Ubiquitous Computing*, vol. 13, no. 6, 2009.
9. M. Terrovitis, "Privacy preservation in the dissemination of location data," *SIGKDD Explorations*, vol. 13, no. 1, 2011.
10. E. Magkos, "Cryptographic approaches for privacy preservation in location-based services: A survey," *IJITSA*, vol. 4, no. 2. [Online]. Available: http://dx.doi.org/10.4018/jitsa.2011070104
11. N. Talukder and S. I. Ahamed, "Preventing multi-query attack in location-based services," in *WISEC 2010*. [Online]. Available: http://doi.acm.org/10.1145/1741866.1741873
12. 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*. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/11/pdf/1_3.pdf
13. J. Sedenka and P. Gasti, "Privacy-preserving distance computation and proximity testing on earth, done right," in *ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*. [Online]. Available: http://doi.acm.org/10.1145/2590296.2590307
14. G. Zhong, I. Goldberg, and U. Hengartner, "Louis, lester and pierre: Three protocols for location privacy," in *Privacy Enhancing Technologies, 7th International Symposium, PET 2007*. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75551-7_5
15. L. Siksnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen, "A location privacy aware friend locator," in *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009*. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02982-0_29
16. L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu, "Private and flexible proximity detection in mobile social networks," in *Eleventh International Conference on Mobile Data Management, MDM 2010*. [Online]. Available: http://dx.doi.org/10.1109/MDM.2010.43
17. D. Freni, C. R. Vicente, S. Mascetti, C. Bettini, and C. S. Jensen, "Preserving location and absence privacy in geo-social networks," in *19th Conference on Information and Knowledge Management, CIKM 2010*. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871480
18. 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.*, vol. 20, no. 4, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00778-010-0213-7
19. P. A. Hallgren, M. Ochoa, and A. Sabelfeld, "Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol," in *PST*. IEEE, 2015, pp. 1–6.

20. Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Privacy Enhancing Technologies*, 2009.
21. A. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *Information, Security and Cryptology - ICISC 2009*. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14423-3_16
22. R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 32, no. 4, pp. 169–178, 1978.
23. C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009, pp. 169–178.
24. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT 1999*. [Online]. Available: http://dx.doi.org/10.1007/3-540-48910-X_16
25. T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology, CRYPTO 1984*. [Online]. Available: http://dx.doi.org/10.1007/3-540-39568-7_2
26. L. Barkhuus and A. K. Dey, "Location-based services for mobile telephony: a study of users' privacy concerns," in *Human-Computer Interaction INTERACT '03: IFIP TC13 International Conference on Human-Computer Interaction, 2003, Zurich, Switzerland.*
27. H. Xu and S. Gupta, "The effects of privacy concerns and personal innovativeness on potential and experienced customers' adoption of location-based services," *Electronic Markets*, vol. 19, no. 2-3. [Online]. Available: http://dx.doi.org/10.1007/s12525-009-0012-4
28. Kathryn Zickuhr, "Three-quarters of smartphone owners use location-based services," May 2012, http://www.pewinternet.org/files/old-media/Files/Reports/2012/PIP_Location_based_services_2012_Report.pdf.
29. I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis, "Where's wally?: Precise user discovery attacks in location proximity services," in *Proceedings of the 22nd ACM SIGSAC CCS, 2015*. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813605
30. M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *MobiSys 2003*. [Online]. Available: http://www.usenix.org/events/mobisys03/tech/gruteser.html
31. B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity," *IEEE Trans. Mob. Comput.*, vol. 7, no. 1. [Online]. Available: http://dx.doi.org/10.1109/TMC.2007.1062
32. C. Wu, C. Huang, J. Huang, and C. Hu, "On preserving location privacy in mobile environments," in *Ninth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2011*. [Online]. Available: http://dx.doi.org/10.1109/PERCOMW.2011.5766939
33. A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176
34. A. C. Yao, "Protocols for secure computations (extended abstract)," in *23rd Annual Symposium on Foundations of Computer Science, 1982*. [Online]. Available: http://dx.doi.org/10.1109/SFCS.1982.38

35. D. Bogdanov, M. Jõemets, S. Siim, and M. Vaht, "How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation," in *Financial Cryptography and Data Security - 19th International Conference, FC 2015*. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-47854-7_14
36. Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *USENIX Security*, 2011.
37. A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Symposium on Theory of Computing Conference, STOC 2012*. [Online]. Available: http://doi.acm.org/10.1145/2213977.2214086
38. C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *CRYPTO (1)*, 2013.
39. K. P. N. Puttaswamy and B. Y. Zhao, "Preserving privacy in location-based mobile social applications," in *Eleventh Workshop on Mobile Computing Systems and Applications, HotMobile 2010*. [Online]. Available: http://doi.acm.org/10.1145/1734583.1734585
40. Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on android)," in *Trust and Trustworthy Computing - 4th International Conference, TRUST 2011*. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21599-5_7
41. H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *Proceedings of the International Conference on Pervasive Services ICPS 2005*. [Online]. Available: http://dx.doi.org/10.1109/PERSER.2005.1506394

Paper **Six**

PrivatePool

Privacy-Preserving Ridesharing

Per Hallgren, Claudio Orlandi and Andrei Sabelfeld

Location-based services have seen tremendous developments over the recent years. These services have revolutionized transportation business, as witnessed by the success of Uber, Lyft, BlaBlaCar, and the like. Yet from the privacy point of view, the state of the art leaves much to be desired. The location of the user is typically shared with the service, opening up for privacy abuse, as in some recently publicized cases. This paper proposes PrivatePool, a model for privacy-preserving ridesharing. We develop secure multi-party computation techniques for endpoint and trajectory matching that allow dispensing with trust to third parties. At the same time, the users learn of a ride segment they can share and nothing else about other users' location. We establish formal privacy guarantees and investigate how different riding patterns affect the privacy, utility, and performance trade-offs between approaches based on the proximity of endpoints vs. proximity of trajectories.

Published in the IEEE 30th Computer Security Foundations Symposium
CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017
Revised version

1 Introduction

Location-based services (LBS) have seen tremendous developments over the recent years. These services have revolutionized transportation business, as witnessed by the success of Uber [53], Lyft [31], and the like. These technologies leverage the idea of *ridesharing*. The up-and-coming service BlaBlaCar [2] epitomizes the simplicity of ridesharing: a user may advertise that they are traveling between two points, but can take a passenger user or ride with another user. The obvious benefit for the users is to reduce the cost of travel.

Motivation Yet from the privacy point of view, the state of the art leaves much to be desired. The location of the user is typically shared with the service, opening up for privacy abuse. The ridesharing app Uber, connecting passengers with private drivers, has been the subject of much privacy debate. Uber and its employees have been allegedly involved in privacy-violating activities from stalking journalists and VIPs to tracking one-night stands [1].

Beyond ridesharing, digitalization of transportation systems and development of self-driving cars [54] opens up further opportunities, which is expected to drive collection of unprecedented amounts of data. With the advent of self-driving cars, such as the Tesla Model 3 [55], the market will sport a fully digitalized car with autopilot functionality in the hands of consumers. In the near future consumers will hold technology that can provide automated commuting services. To optimize such services, the daily itinerary of end users is needed, such that users who travel to and from the same approximate area can be transported together. When this occurs, users are expected to provide the service provider with their private location data. This raises alarming privacy concerns.

Overall, the sensitivity of the private location information in ridesharing applications poses a major challenge: *how to preserve privacy without hampering the functionality of the ridesharing services?*

Privacy-preserving ridesharing We propose PrivatePool, a novel model for privacy-preserving ridesharing. Our goal is to provide the unhampered functionality of normal ridesharing applications without compromising privacy through means of *secure multi-party computation (SMC)* [3, 11], where participants can jointly compute a function based on private inputs.

This work focuses on *flexible* ridesharing, similar in that respect to BlaBlaCar we accommodate users that are willing to divert from their original path. The economical and environmental savings have impact on whether ridesharing is feasible.

This can be manifested by a maximum deviation from each user's path and a minimum overlap of the trajectories for desirable ridesharing. Intuitively it is harder for the user to deviate in the middle of the trip than at the beginning or end, which means that the deviation allowed is not constant. For instance, if a ride is shared over a larger distance, such as between cities, it would likely be acceptable for the users to use public transport to coordinate within the origin or destination city, but a rendezvous near the center of the trip is much more restricted.

In contrast to BlaBlaCar, however, we are interested in ridesharing with no trust to third parties. We envision that our approach will accommodate a decentralized version for ridesharing services that will break away from the traditional full disclosure of user location. As such, our work is a step in the direction of building theoretical foundations for such a decentralized service, that – in the long run – has potential to lead to practical systems for solving problems like the ones publicized in [1].

To the best of our knowledge, this paper presents the first model for privacy-preserving ridesharing. To accommodate flexible privacy-preserving ridesharing, we draw on two key building blocks: one is based on *proximity* of the journeys' start- and endpoints and the other one is based on *intersection* of the routes between the start- and endpoints.

There has been much recent work on secure multi-party computation of *proximity testing* [57, 48, 47, 7, 32, 35, 45, 15, 42], where the goal is to compute whether two parties are close to each other without revealing their relative distances and positions to each other or to any third party. However, proximity testing by itself does not solve the ridesharing problem. First, we need a method to securely extend proximity testing to yield a match when *both* the respective start and endpoints are within desirable proximity. A naïve application of proximity protocols on the start/endpoints would violate privacy: if for two rides the start points are far but the endpoints are close, the naïve approach would reveal the proximity of the endpoints in the absence of a shared ride. Second, flexible ridesharing in some scenarios necessitates considering trajectories and not only the start- and endpoints.

Indeed, since we are interested in flexible ridesharing, the start/endpoints might be actually far enough while there is still a large segment of the route that can be shared by the users. A typical example is an intercity ride that starts and ends in different parts of the origin and destination cities. In this scenario, we draw on the *private set intersection (PSI)* [6]. By applying PSI to sets representing trajectories/routes, we can compute whether there is a sufficient overlap to warrant a shared ride. Again, PSI by itself does not solve the ridesharing problem either. Recall that we

only want to reveal a match when there is a sufficient segment that overlaps between the users' trajectories. This motivates the need for a *threshold private set intersection (T-PSI)* protocol, enabling us to achieve private and flexible trajectory matching. As a side remark, we note that the term threshold set intersection, or over-threshold set-union, has also been used in the literature to describe the scenario when one wants to disclose all elements among n users' private input sets which occurs in the input of at least a threshold number t users [24, 25]. This scenario is different from the one considered in the paper.

Finally, achieving the above goals would only make sense if it yields techniques that provide utility, respect privacy, and have feasible computation overhead. We thus set out to evaluate these techniques on a collection of realistic ridesharing patterns. In the evaluation, it is also our goal to compare our approach with respect to generic secure multi-party techniques, with the focus on the state of the art technique of garbled circuits [56].

Contributions We develop the first model for privacy-preserving ridesharing. On the policy side, the model accommodates flexible ride sharing, allowing to be parametric in how long the users are willing to travel to meet up for a shared ride and for how long a ride they require in a successful match. On the enforcement side, we develop two independently interesting enforcement mechanisms. It enables ride matching by both the proximity of start- and endpoints and by trajectory matching. For endpoint matching we build on top of existing work utilizing additively homomorphic encryption to create a privacy-preserving protocol. At the core of the trajectory matching, we design a novel threshold private set intersection (T-PSI) protocol, for which we establish rigorous privacy guarantees. The main technical contribution is the definition and construction of a so called *threshold key encapsulation mechanism* (T-KEM). We present an overview of realistic ride patterns and evaluate our mechanisms with respect to these patterns. Benchmarking against the patterns demonstrates that we benefit from the generality of our approach: start/endpoint matching and trajectory matching excel on different patterns. At the same time, our benchmarks also show that our techniques are preferable over a generic approach, as implemented by garbled circuits.

Limitations While the cryptographic techniques evaluated can be called practical out-of-context, a fully-fledged ridesharing system needs more work before it is useful in practice. On the one hand, practically-oriented research results are needed to show how to use SMC in general without information leakage from a running system. As SMC works on the application layer, it is oblivious to information on

other levels such as IP addresses etc. Lacking an outlook for a secure full system, in this first effort for privacy in ridesharing applications several details which could be privacy-sensitive are left for future work. This includes the time of the ride and the identity of the users.

On the other hand, further foundational work is needed in terms of scalability to large numbers of users. The state of the art is making great leaps in this direction, such that is now possible to efficiently compute a fixed function of many users [21]. However, state-of-the art SMC protocols are only efficient for a limited number of users for cases like ours where each party wants to evaluate a different function, as the question "who can I share a ride with?" is context-sensitive. This limitation is shared with the entire line of work on privacy-preserving location proximity protocols (e.g., [57, 35, 45, 15]).

Overview The rest of the paper is organized as follows. Section 2 presents ridesharing concepts, such as feasibility, and discusses ridesharing patterns. Section 3 explains the concepts of proximity testing and private set intersection, key building blocks in our approach. Section 4 present a novel mechanism for threshold private set intersection, which allows us to parameterize the privacy of trajectory matching. Section 5 details the threshold key encapsulation mechanism that lies at the heart of our cryptographic construction. Section 6 presents the experiments that illustrate that different patterns benefit from the different mechanisms in our approach. Section 7 discusses related work. Section 8 concludes.

2 Ridesharing concepts

This section presents the basics of our ridesharing model and characterizes ridesharing feasibility. Based on these, we discuss different patterns where ridesharing is feasible, of which two are studied further in Sections 3, 4 and 6.

2.1 Modeling ridesharing

The model considers users who set out on *trips* as defined in Definition 1, traveling from one point to another. To describe such trips, we consider the map as an undirected, unweighted graph $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. An edge e can be imagined as being a two-way road section, connecting two coordinates.

The model does not include the identities of the users or the time frame in which a user is willing to participate in ridesharing, and does not say anything about

whether or not a service provider is involved in the exchange. Only the abstract view of the two parties exchanging location data is considered, leaving protection of other private information and detailing what data is shared with service providers and third parties for future work.

Definition 1 (Trip). *Given a graph $G = (V, E)$, a* trip T *is an acyclic sequence of consecutive vertices $v_i \in V$, where $v_s = v_0$ is the origin and $v_f = v_{|T|-1}$ is the destination, such that $(v_i, v_{i+1}) \in E$ for all $i \in \{0, \ldots, |T| - 2\}$.*

A set of consecutive vertices in a trip T is called a *segment* as per Definition 2.

Definition 2 (Segment). *Given a graph $G = (V, E)$, a* segment S *of a trip T in G is an acyclic sequence of consecutive vertices $v \in V$, such that S is a subsequence of T.*

Users conduct trips by traversing the graph using the shortest path from their origin to their destination. The set of vertices visited when traversing the shortest path is called the user's *trajectory*. Users are considered as traveling with constant speed for the scope of this work, such that both the spatial and temporal cost of traversing a road section is equivalent. How the graph and the trajectories are constructed are out of the scope of this work. If the application need to take e.g. traffic congestion into account, the length metric should be updated to accommodate this.

When utilizing ridesharing, users may deviate from their trajectory, for the purpose of one user aligning their trip to the other user's trajectory. That is, the first user A pays an extra cost, extending their trip such that it includes a part of another user B's trajectory, while B travels exactly along their trajectory. Given any protocol implementing ridesharing, one can switch roles and rerun the protocol to symmetry between A and B. Users are restricted to sharing a ride during a single segment, which leads to five segments of significance during A's trip, called the *ridesharing segments* of the trip, as illustrated in Figure 1. These segments are:

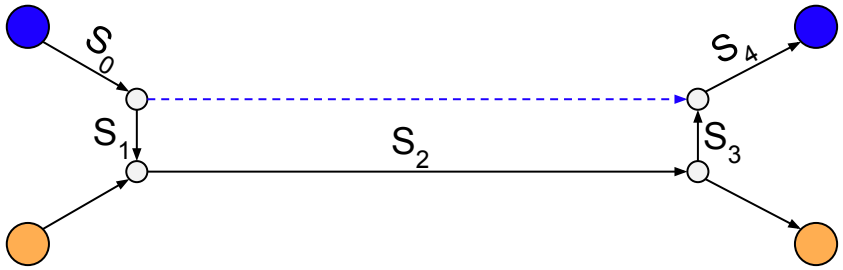

Fig. 1. The five common ridesharing segments

1. S_0, the start segment, is a part of the trajectory and includes at least $v_s \in T$.
2. S_1, where the user is traveling to the ridesharing segment, is a part of neither user's trajectory.
3. S_2 is the ridesharing segment, the maximal segment where the two users can share a ride.
4. S_3 is the counterpart of S_1, where the user is traveling from the ridesharing segment.
5. S_4, similarly to S_0, is a part of the trajectory, where the last vertex is $v_f \in T$.

In short, S_0 and S_4 are parts of the trajectory, which would be also without ridesharing. S_1 and S_3 are the additional path that the user travels, leaving their trajectory and aligning with the other user's trajectory. Finally, S_2 is the part of the trip where the users share a ride. To make the representation of a trip more concise, Definition 3 specifies the length of a segment S as $l(S)$, assuming that we have a Euclidean distance defined for each edge. Note that $l(S) \neq |S|$.

Definition 3 (Segment length). *Given a segment S for some trip through a graph $G = (V, E)$, let $l(S) = \sum_{i=0}^{|S|-2} d\left(S[i], S[i+1]\right)$, where $S[j]$ is the jth vertex in S and $d(p_1, p_2)$ is the Euclidean distance between the two points p_1 and p_2.*

2.2 Ridesharing feasibility

When ridesharing comes with a low enough cost and high enough benefit for both parties, it is called *feasible* ridesharing. Feasibility is modeled with two parameters.

First, an upper limit as to how much a user is willing to extend the trip (i.e., deviate from their trajectory) before and after S_2. As previously highlighted, this distance may depend on how far the user has traveled from their endpoints (i.e., how long S_0 and S_4 are). This is captured using a deviation function $\Delta_T(v)$, where T is the user's trajectory and $v \in T$ is a vertex along the trajectory. The evaluation of $\Delta_T(v)$ at a vertex v gives us the user's *deviation limit* at that stage of their trip, which is a measure of how flexible the user is. Ridesharing is feasible only when $l(S_1) < \Delta_T(v_0) \wedge l(S_3) < \Delta_T(v_1)$, where S_1 and S_3 are ridesharing segments of T, $v_0 \in T$ is the first vertex in S_1 and $v_1 \in T$ is the last vertex in S_3.

Secondly, we model a lower limit of the length of the ridesharing segment as a distance threshold, called t, simply called the *threshold*. This enables the model to be used in cases where the trip's economic or environmental cost is to be reduced by some factor for ridesharing to be feasible. Concisely, ridesharing is feasible only if $l(S_2) > t$.

Finally, the formalization of a feasible ridesharing scenario is given in Definition 4. Using $\Delta(\cdot)$ and t, most preferences of a user can be modeled. Note that no restriction on $\Delta(\cdot)$ has been made, it could be an arbitrary function, and may be defined on a per-user and per-query basis, such that A specifies the deviation function while making each query.

Definition 4 (Ridesharing feasibility). *For any fixed threshold t and deviation function Δ, given two trajectories P_A and P_B for users A and B in $G = (V, E)$, ridesharing is feasible for A along a segment $S = \{p_s, \ldots, p_f\}$ of P_B if and only if $l(S) > t$ and there exist two points $P_A[i]$ and $P_A[j]$, with $i < j$, such that:*

$$d(P_A[i], p_s) < \Delta_{P_A}(P_A[i])$$
$$\wedge\, d(P_A[j], p_f) < \Delta_{P_A}(P_A[j])$$

Another important measure is how much A's total trip is extended by utilizing ridesharing. This is bounded by Δ, as per Theorem 1.

Theorem 1 (Maximum trip extension). *If ridesharing is feasible for A for a trip T with ridesharing segments $S_i, i \in \{0..4\}$ and where $v_0 \in T$ is the first vertex in S_1 and $v_1 \in T$ is the last vertex in S_3, the total trip will have a length shorter than $l(T) + 2(\Delta_T(v_0) + \Delta_T(v_1))$.*

Proof. Let the skipped segment of A's original trip be $S_s = T \setminus (S_0 \cup S_4)$. Now, assume by contradiction that the total trip is extended by at least $2d$ with $d = \Delta_T(v_0) + \Delta_T(v_1)$ such that $2d \leq l(S_1) + l(S_2) + l(S_3) - l(S_s)$, which gives us Equation 1.

$$l(S_s) + 2d \leq l(S_1) + l(S_2) + l(S_3) \tag{1}$$

By construction, from the first vertex of S_2 to the first vertex of S_s, the maximum distance is $\Delta_T(v_0)$. Similarly, the distance from the last vertex of S_s to the last vertex of S_2 is at most $\Delta_T(v_1)$, which gives Equation 2.

$$l(S_1) + l(S_3) < d \tag{2}$$

Using Equation 2, we we can substitute $l(S_1)$ and $l(S_3)$ in Equation 1, and see that $l(S_s) + 2d < l(S_2) + d$. Now we can apply Equation 2 again, on the left-hand side, to arrive at $l(S_s) + l(S_1) + l(S_3) + d < l(S_2) + d$, or simply $l(S_1) + l(S_s) + l(S_3) < l(S_2)$. This implies that the shortest path from the first vertex of S_2 to the last vertex of S_2 is via S_1, S_s, and S_3. Therefore, S_2 is not a part of any shortest path in G, and thus not a part of B's trajectory. □

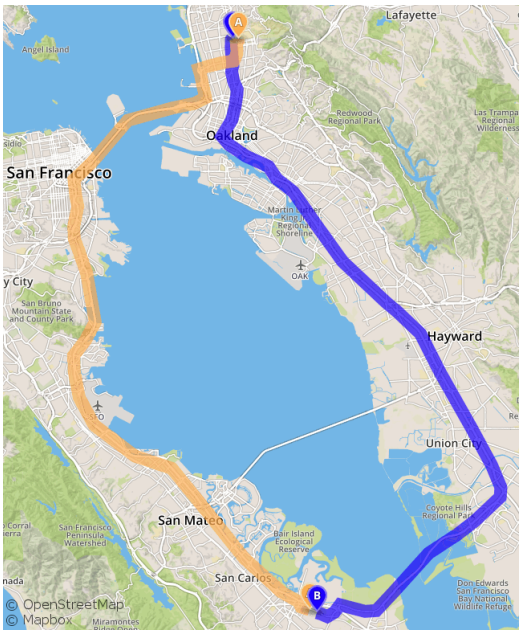

Fig. 2. The paths barely intersect while the endpoints are close

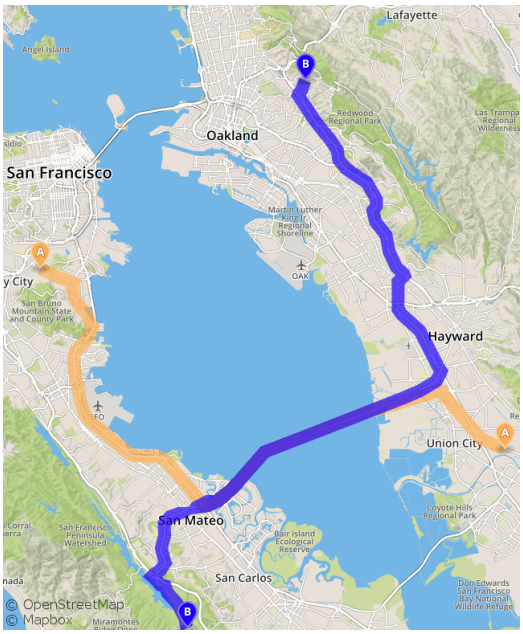

Fig. 3. The paths intersect substantially while the endpoints are far

2.3 Ridesharing patterns

This section discusses distinct *ridesharing patterns* which can be captured with the model.

The *proximity-based* pattern is a convenient ridesharing pattern. In this pattern, there is only a short commute before and after the ridesharing segment, making the effort for changing means of transport low. Such scenarios are easy to find in sparsely connected areas, which is a common situation close to bodies of water, such as rivers with few bridges or lakes. One example is the San Francisco Bay Area as depicted in Figure 2 which shows two trips from Berkley to Redwood City. In this example, the trajectories intersect only an insignificant part of the trip, but where ridesharing clearly is possible as the origins and destinations are only 10 minutes walking distance from each other.

On the other hand, many applications are likely interested in detecting ridesharing patterns where the traveled distance is minimized, which is captured by the *intersection-based* pattern. Consider for instance the one depicted in Figure 3, where the extra distance traveled to enable ridesharing is not relevant. Both users would either way travel along the ridesharing segment. In this case the endpoints are far apart, but the overlap between the two trajectories is roughly half the trip, which is

a reasonable distance to utilize ridesharing. This pattern is likely to be found if users travel via common junction points, e.g. along major highways.

Both the western and the eastern route are acceptable to each party in Figure 2. This highlights a case that is not implicitly captured by the described model. In the case when B could go both the western and the eastern route, but where the eastern one is negligibly shorter, but A cannot take the eastern route (A is e.g. going from downtown San Francisco to Redwood City). However to capture such cases it suffices to do a separate matching for each acceptable route. An alternative approach is to let A and B meet in the middle, as if both are willing to deviate by some distance, more rides can be shared. This setting can be trivially captured by adding B's deviation limit to that of A for every vertex.

While the above patterns are common, we note that there are yet other scenarios, for instance several scenarios would match a *hybrid* pattern of the above two as depicted in Figure 4. In this case the origins (similar for the destinations) are close but an overlap occurs much later. Correspondingly for the Bay Area example in Figure 2, assuming the origins are the south endpoints, if either user would have continued their journey farther north, only the origin endpoints would be closer than the deviation limit, and the intersection would be smaller than t (for reasonable parameters).

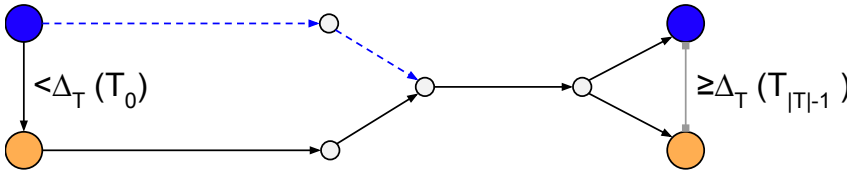

Fig. 4. Ridesharing pattern with small overlap and far endpoints

Having presented the ridesharing concepts and patterns, we are now ready to describe our mechanisms for achieving privacy-preserving ridesharing.

3 Privacy-preserving ridesharing

This section details how to privately detect the concrete patterns given above. As will be shown, the two patterns depicted in Figure 5 and Figure 6 are amenable to SMC. The first pattern can be realized using privacy-preserving proximity testing with a few modifications and the second pattern is exactly the use case for private set intersection.

3.1 Privately detectable patterns

Given the patterns above, the question remains how to design an algorithm to efficiently detect ridesharing scenarios – and further, to design algorithms that are amenable to SMC. The goal of privacy-preserving ridesharing is to disclose the ridesharing segment only if ridesharing is feasible, and no information otherwise.

Two of the distinct patterns turn out to be susceptible to SMC techniques. These two patterns result in enforcing either of the two criteria $t \geq l(T) - \Delta_T(T_0) - \Delta_T(T_{|T|-1})$ and $\Delta_T(\cdot) = 0$ for proximity-based and intersection-based patterns, respectively (where again t is the threshold and Δ_T is the deviation function). That is, in the first case the shared trip is as long as the full trip, except for possibly the cost of A traveling to and from B's starting point as illustrated in Figure 5. In the second case, the shortest paths intersect, and there's no extra distance to travel to enable ridesharing as illustrated in Figure 6.

A straightforward "bruteforce" algorithm that attempts finding the longest possible ridesharing segment is to find the first vertex $v_f \in A$ which is closer than $\Delta_{P_A}(v_f)$ to any point on B's trajectory and last vertex $v_l \in A$ which is closer than $\Delta_{P_A}(v_l)$ to any point on B's trajectory. Ridesharing is then feasible if between v_f and v_l if $d(v_f, v_l) > t$. However, such an algorithm does not have any apparent efficient implementation using SMC.

Instead, we now outline algorithms for proximity-based and intersection-based ridesharing patterns, which are amenable to SMC. Proximity-based patterns are detected by checking that the start and endpoints are close. As seen in Section 3.2, this can be achieved with SMC with private proximity testing. Intersection-based patterns can be detected by computing the intersection $I = P_A \cap P_B$, and conclude that ridesharing is feasible when $l(I) > t$. As seen in Section 4, this is achieved through our new primitive T-KEM.

For the hybrid pattern (and others), there appear to be no apparent efficient privacy-preserving solution. One could imagine detecting proximity endpoints and subsequently checking for intersection. However, such an approach would leak endpoint proximity even if it is impossible to share a ride. Of course, generic SMC solutions could be used to achieve the sought functionality, but as we will see in Section 6.2 such solutions are as of yet not efficient enough for most applications.

How many scenarios do not match the patterns detailed above, and how relevant they are for practical applications, is hard to judge without empirical data. To shed some light on the matter, we have carried out experiments on real-world data from the New York City taxi services, as detailed in Section 6.1. The experiments show that using both approaches can achieve up to 92% of the effectiveness of the entire model,

and that both endpoint-based matching and intersection-based matching excel on different rides.

3.2 Proximity-based Ridesharing

For the first pattern, depicted in Figure 5, the users A and B want to check whether their start- and endpoints are closer than a fixed limit r ($\Delta_A(\cdot) = \Delta_B(\cdot) = r$). To check if two points are close is often called *proximity testing*, for which there are several privacy-preserving solutions with different strengths and weaknesses, such as the amount of privacy provided, the number of roundtrips, and performance requirements [35, 45, 15].

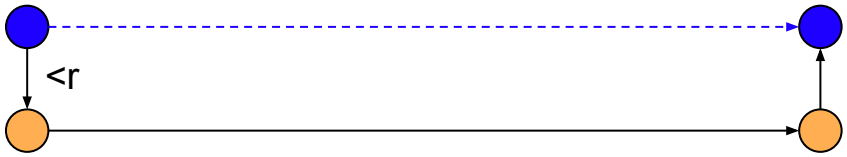

Fig. 5. Ridesharing pattern with maximum ridesharing segment

However, in addition to checking the proximity of two endpoints, a construction is needed that allows to disclose a result only if both endpoints are close to each other. This is not trivial to achieve with an arbitrary proximity testing solution. Luckily, there are several solutions based on homomorphic encryption, where it's common to represent a positive response as encryption of 0, and a negative response as an encryption of a uniformly random value. This enables the two results to be multiplied to create an "or" operator, or added to create an "and" operator. Thus, when the start and endpoint results have been computed, they can be added to produce the final result.

We pick the InnerCircle approach by Hallgren et al. [15] as a starting point because it allows for the lowest round-trip time amongst protocols in the literature. The protocol is also among the better when it comes to other properties such as number of roundtrips and the level of privacy provided. The construction requires an unfortunate restriction in terms of precision, where discretization up to 10 or 100 meters is required. This imprecision is likely not mission-critical in a ride-sharing application and thus the boost in performance as compared to other approaches outweighs this drawback.

As with other recent efficient approaches [35, 45], it works with *additively* homomorphic encryption utilizing well-studied cryptosystems. Further motivating this choice is that the line of work by Hallgren et al. is amenable to composition while protecting against malicious adversaries [14] with few modifications.

The construction is briefly detailed here to highlight the necessary adjustments. The user A holds the private key, for which B has the corresponding public key. A queries B for proximity by issuing a request with a triplet $(x_A, y_A, x_A^2 + y_A^2)$. B computes the squared euclidean distance as:

$$D = x_A^2 + y_A^2 + x_B^2 + y_B^2 - 2\left(x_A x_B + y_A y_B\right)$$

B can do all of these computations in the encrypted domain as they are only linear operations and thus supported by an additively homomorphic encryption system. Next, B needs just to compare D with the radius value r, which is done by encoding the comparison in a set

$$\{(D - i)\,\rho_i | i \in \{0..r^2\}\}$$

Where each ρ_i is an independent random number in the plaintext space. The set is sent to A in a random order, such that A only can deduce whether $\exists i < r^2 : i = D$, which is equivalent to $D < r^2$.

Now back to the ridesharing setting, where a procedure is executed in three steps. First, B computes an array which encodes the proximity result. Secondly, for B to be able to combine the proximity results for the origin and destination, the arrays need to reduced to a single results. This can be done using successive "or" operations, via multiplications as mentioned earlier. Using additively homomorphic means that multiplications of ciphertexts cannot be computed locally. A common workaround is to send blinded values to A, who can decrypt, multiply, and encrypt the result before sending it back [27]. The blinding is then removed and B now has two ciphertexts encrypting either 0 or a uniformly random number. Though the multiplications incur an overhead, consecutive outsourced multiplications will introduce only a logarithmic number of round trips [16]. As a third and final step, B executes the "and" operation of the two ciphertexts, and the final result is sent back to A.

3.3 Intersection-based Ridesharing

The second pattern, as shown in Figure 6, is for users who want to minimize the deviation and find a trip such that $\Delta_T(\cdot) = 0$. That is, the user is looking for trips

that overlap perfectly with their own trip, which as highlighted previously is solvable using PSI. Recent research has yielded ad-hoc solutions for PSI [39] which are highly efficient. However, applying PSI directly would always disclose the intersection, regardless of its size. For instance, with two orthogonal trajectories which only intersect at a single point, disclosing this point does not provide information useful for ridesharing, but leaks privacy-sensitive location-information. Thus, as outlined in Section 4, we define a novel ad-hoc protocol for threshold-conditioned PSI which does not have these fallacies.

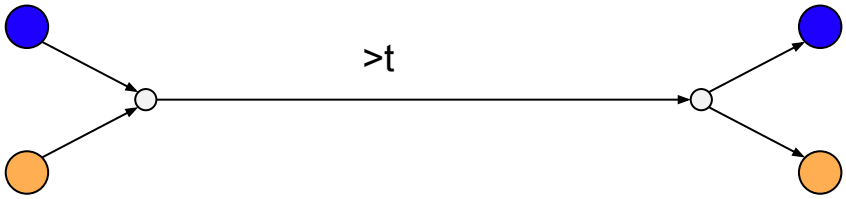

Fig. 6. Ridesharing pattern with minimum cost

There are also efficient solutions utilizing generic circuits to achieve PSI [17]. These are by construction amenable to composition with other methods, and are therefore an interesting comparison to our novel scheme. We outline a performance comparison in Section 6.2.

3.4 Practical applicability of SMC

The above mentioned techniques share restrictions that apply to SMC in general. As highlighted in Section 1, these are both in terms of scalability in the number of users, and in terms of complete system-wide enforcement.

There is much work to be done before privacy of location-data can be guaranteed when considering a running application on existing operating systems for mobile devices. This can be seen through various side-channel attacks on location data [29, 49, 34, 38, 33], including vectors such as power consumption and carrier signal strength. While anonymization techniques such as onion routing may be used to escape threats from internet infrastructure providers, cellular network providers provide further complications which are harder to tackle with existing hardware. Theoretical models for security, such as the one used in this work, try to minimize the trust that is needed to be placed in a party. Instead of the user completely trusting the service provider, one allows the user to place trust in technology/cryptography.

Though there are no current solution that frees the user from trusting anything but cryptography as per the above issues, reducing the amount of trust the user has to place on a service provider or other users is still of tangible value.

In terms of scalability, there are no solutions that are asymptotically comparable to that of using a trusted third party. While there are many solutions for secure *multiparty computations*, such that any number of parties can jointly compute a function without the communication, the restriction is to compute a *single* function. These solution are very useful if the function has the same output no matter who issues the query, such as services for auctions or elections, or when only one party is interested in the output, perhaps according to a business agreement. However, when the question is no longer "Who made the highest bid?" or "Who had the most votes?", but rather "Can I share a ride with someone?", the query is not so easily answered. The solutions proposed in this work are both peer-to-peer, which means that a user needs to communicate with every other party to check for ridesharing opportunities, and thus requires $\mathcal{O}(n)$ parties to communicate. This may make the solution hard to apply in various scenarios, but even with recent and very promising works focusing specifically on scalability, each party needs to communicate for every evaluation of a function [21, 13], which again requires $\mathcal{O}(n)$ communication.

There are many more issues to consider when trying to build a system utilizing SMC. One such issue is secure key distribution. We leave this issue, as well as identity management in general, out of scope and subject to future work. Further, the adversary considered in this work is passive, while many applications need to be secure against active adversary. Both the technique for endpoint matching and for set intersection can be adjusted to give heightened security against active adversaries at the cost of performance [14, 43]. As seen in Section 6.2, the choice of a weaker adversarial model allows us to cater for many applications, while this may not be the case if using a slower (though more secure), version.

4 Threshold PSI

In this section we describe our solution to the problem of securely evaluating the threshold private set intersection (T-PSI) between the sets of two parties, called the *sender* and the *receiver*, in the presence of a passive adversary and static corruption. The sender and receiver have as inputs two sets A and B respectively, both of size n. At the end of the protocol the receiver should learn $(A \cap B)$ if $|A \cap B| \geq t$ for some predetermined threshold t or nothing otherwise.

The sender learns nothing. Though the aim with this work is privacy-preserving ridesharing, we hypothesize that the T-PSI construction is useful in many other areas. For instance, it could be applied to dating applications; If Alice requires her future husband to share at least two of her hobbies, it makes little sense to reveal any of Alice's hobbies to Bob if they share only one.

Before proceeding further, let us outline some basic concepts as often used in the SMC literature in Definition 5, Definition 6 and Definition 7.

Definition 5 (Negligible functions). *A function* $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ *is* 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.*

Definition 6 (Indistinguishability). *The two random variables* $X(n, a)$, $Y(n, a)$ *(where* n *is a security parameter and* a *represents the inputs to the protocol) are called* computationally indistinguishable *and denoted* $X \stackrel{c}{\equiv} Y$ *if for any probabilistic polynomial time (PPT) adversary* $\mathcal{A}$ *the function* $\delta(n)$ *is negligible:*

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

Definition 7 (Semantic Security). *A public key encryption scheme* E *is* semantically secure *or IND-CPA secure if the advantage of any PPT adversary of winning the below game against* $\mathcal{C}$ *is negligible. The game is won if the attacker outputs* $b' = b$*.*

1. $\mathcal{A}$ *outputs two different plaintexts* (m_0, m_1)*.*
2. $\mathcal{C}$ *flips a random* $b \in \{0, 1\}$
3. $\mathcal{C}$ *outputs* $E(m_b)$
4. $\mathcal{A}$ *outputs b'*

Now for a brief outline of simulation-based proofs, for which our definition follows from common definitions of secure multi-party computation in the passive (or honest-but-curious) adversarial model [10, 30], but is here simplified for the case with two parties. For two parties Alice and Bob, where $Alice$ has inputs $\overrightarrow{x}$ and Bob inputs $\overrightarrow{y}$, the model formalizes the output of a protocol as $f(\overrightarrow{x}, \overrightarrow{y}) = (g(\overrightarrow{x}, \overrightarrow{y}), h(\overrightarrow{x}, \overrightarrow{y}))$. The function f is called the *functionality* of the protocol. The functions g and h are functions describing all outputs presented to Alice and Bob from the execution of the protocol, respectively.

Definition 8 (Privacy). Privacy *for deterministic functionalities hold when the overall knowledge of each party after the execution of the protocol, called the party's* view*,*

can be computed from the inputs and outputs of that party. This is called that the view can be simulated. *That is, for the two-party case with Alice and Bob as described above, one must show that:*

$$\{\mathcal{S}_{Alice}(\overrightarrow{x}, g(\overrightarrow{x}, \overrightarrow{y}))\} \stackrel{c}{\equiv} \{view_{Alice}(\overrightarrow{x}, \overrightarrow{y})\}$$
$$\{\mathcal{S}_{Bob}(\overrightarrow{y}, h(\overrightarrow{x}, \overrightarrow{y}))\} \stackrel{c}{\equiv} \{view_{Bob}(\overrightarrow{x}, \overrightarrow{y})\}$$

where $\mathcal{S}_{Alice}$ and $\mathcal{S}_{Bob}$ are the simulators for Alice and Bob, respectively.

4.1 Phasing&Co.

The last few years have seen a large improvement in the efficiency of protocols for PSI, most notably the Phasing protocol [40] and its improvements [39, 26][1]. For the sake of exposition we choose to abstract as much as possible the properties of the Phasing protocol to clearly communicate the main changes which are necessary to turn this into a T-PSI protocol. From a very high-level point of view, the Phasing protocol proceeds in two stages:

Phasing – Stage 1: Masks Generation In the first stage the sender and the receiver compute sets of random strings called *"masks"* based on their input sets. In particular, at the end of the first stage, the sender learns (for each element $a \in A$) a set of random masks $M_a = (m_1, \ldots, m_j)$. Similarly, for each $b \in B$ the receiver learns a set of random masks $M_b = (m_1, \ldots, m_k)$ under the constraint that, if $a = b$, then $|M_a \cap M_b| = 1$. At the same time, choosing masks of appropriate lengths ensures that if $a \in A$ but $a \notin B$ then $\forall b \in B$, $M_a \cap M_b = \emptyset$. The first stage of Phasing can be used directly in our final construction, to reuse the subroutine we write for short:

$$\mathsf{Phasing}(A, B) \rightarrow (U, (V, R_V))$$

Where $U = \{M_a | a \in A\}$ is output to the sender, and $V = \{M_b | b \in B\}$ and R_V is output to the receiver. R_V is a reverse-mapping needed by the receiver to find which element in B is masked by a given mask. We define a function to find the reverse mapping, $\mathsf{RevMask}(\mathsf{M_b}, \mathsf{R_V}) \rightarrow b$, such that for any element $b \in B$ it returns the item b masked by M_b.

[1] The term "Phasing" first appeared in [39] but here we use it as a general term for the whole family of protocols.

Phasing – Stage 2: Computing the Intersection In the second stage of the Phasing protocol the sender sends the set $U = \{M_a | a \in A\}$ to the receiver, who can then compute the intersection of U and each M_b to determine whether b is in the intersection. We define the procedure Isect to reuse in the final protocol:

Procedure $\mathsf{Isect}(U, V, R_V)$
  $Z \leftarrow [\,]$
  for $M_i \in V$ **do** :
    if $U \cap M_i = 1$ **then**:
      $Z \leftarrow Z :: \mathsf{RevMask}(\mathsf{M_i}, \mathsf{R_V})$
  return Z

Security of Phasing For completeness, we show how to construct simulators for our high-level view of Phasing, called $\mathcal{S}_S^{\text{PHG}}$ and $\mathcal{S}_R^{\text{PHG}}$ for the both the sender and receiver, respectively. The claims follow from the proofs of the concrete instances of phasing protocols [40, 39, 26]. The privacy of the overall protocol follows from the following two properties about the first stage of Phasing:

- in the view of the receiver for the first stage the masks which are not in the intersection are computationally indistinguishable from uniform random strings.
- the sender does not learn anything about the set B of the receiver during the first stage. Formally, the view of the sender (and in particular $M_a \forall a \in A$) can be simulated without access to the input B.

The proof of the Phasing protocol [40] implicitly contains the proof of these two properties.

Since the sender does not receive any messages during the second stage and the first stage satisfies the property that it is possible to simulate M_a without access to B, then the protocol is secure against passively corrupt senders. Formally, there exists a simulator for the sender $\mathcal{S}_S^{\text{PHG}}(A) \rightarrow U$ that takes as input the input set A and produces a set of masks U which is indistinguishable from the view of the sender in a real protocol execution.

To argue the security of Phasing against a passively corrupt receiver, the authors construct a simulator that produces a set U to contain the desired intersection with V (consistent with the input B and the output V), and otherwise populates the set U with uniformly random strings. In a nutshell, the simulator for the receiver $(U, V) \leftarrow \mathcal{S}_R^{\text{PHG}}(B, Z)$ works as follows:

1. The simulator $\mathcal{S}_R^{\text{PHG}}(B, Z)$ parses its input and checks that $Z \subseteq B$ and aborts otherwise;

2. The simulator samples uniformly random masks from $\{0,1\}^\lambda$ for the receiver $V = \{M_b | b \in B\}$.
3. The simulator constructs the set of masks for the sender U in the following way: for every $b \in Z \cap B$, the simulator picks a random mask from M_b and adds it to U. Then, the simulator fills U with $n - |Z|$ random masks.

This simulation is indistinguishable from the view of the receiver in a real execution of the protocol. We now extract a property which is needed later in our construction and which is satisfied using $\mathcal{S}_R^{\text{PHG}}$ above:

Definition 9 (Phasing – Sender's Privacy). *We define a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

1. *$\mathcal{A}$ outputs two sets (A, B)*
2. *$\mathcal{C}$ flips a random coin $b \in \{0,1\}$*
3. *If $b = 0$, $\mathcal{C}$ output $\mathsf{Phasing}(A, B)$*
4. *If $b = 1$, $\mathcal{C}$ outputs $\mathcal{S}_R^{PHG}(B, A \cap B)$*

We say that the Phasing *protocol satisfies sender's privacy if for all PPT $\mathcal{A}$ the probability that $\mathcal{A}$ guesses b correctly is at most a negligible factor away from $1/2$.*

In the following, we will make use of the fact that the output of Phasing is indistinguishable from random. In particular this implies that it is indistinguishable from the T-KEM.Gen method (which is defined below), which allows to compose T-KEM and Phasing in Section 4.3.

Efficiency of Phasing For completeness, we also recall that in [40] the Phasing subroutine is implemented using a form of oblivious pseudorandom function (OPRF) which is in turn constructed efficiently thanks to oblivious transfer (OT) extension [19]. A trivial implementation of the Phasing subroutine using the OT induced OPRF would lead to the output of the sender U to have size n^2 (and the output of the receiver V to have size n). Combining the OPRF with clever data structures (e.g., Cuckoo hashing), the authors of Phasing managed to turn the size of the output of both parties to be $\mathcal{O}(n)$ ($|U| \leq 2.4n$ for reasonable sizes of n), which greatly improves in the overall complexity of the protocol.

4.2 Threshold Key Encapsulation Mechanism (T-KEM)

To construct a T-PSI protocol from the Phasing protocol family we introduce a new cryptographic tool which we choose to call *threshold key encapsulation mechanism* or T-KEM. A T-KEM is defined as follows:

Definition 10 (T-KEM). *A T-KEM scheme is defined by the three algorithms called* $(\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ *with the following syntax:*

- *The generation algorithm*
$$\mathsf{Gen}(1^\lambda) \to u$$
on input a security parameter λ*, outputs random values* u *from* $\{0,1\}^\lambda$*.*
- *The key encapsulation algorithm*
$$\mathsf{Encap}(U, t) \to (k, H)$$
on input a set of strings $U = \{u | u \in \{0,1\}^\lambda\}$ *and a threshold* t*, outputs a random key* $k \in \{0,1\}^\lambda$ *and a "hint"* H*.*
- *The key decapsulation algorithm*
$$\mathsf{Decap}(V, H) \to k'$$
on input a set of strings $V = \{v | v \in \{0,1\}^\lambda\}$ *and a hint* H *outputs either* $k' = k$ *or some failure symbol* $\perp$*.*

The idea of a T-KEM is that the two parties both have a set of random numbers, from which they want to derive a common key if and only if the two sets overlap by a threshold number of t items. As an example, one can imagine a single set being generated at random and then stored on two separate unreliable storage units. Both parties retain one storage unit. Later on, the two users want to establish a secure channel between them by using the material on the storage units. As long as the storage is not "too faulty" i.e., the intersection between the two (now different) sets of keys is larger than some threshold t, then the two parties will be able to derive the same key. In the PSI setting, the keys will be generated using the Phasing scheme, which will guarantee the necessary overlap only if the initial sets have a large enough overlap.

We say that a T-KEM is *correct* if, for all V, U s.t. $|V \cap U| \geq t$ the following probability is at most negligible

$$\Pr[\mathsf{Decap}(V, H) \neq k | (k, H) \leftarrow \mathsf{Encap}(U, t)]$$

We define security of T-KEM with the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Definition 11 (T-KEM security). *We say that a T-KEM is* secure *if for all PPT adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *outputs* $b' = b$ *in the following game is at most a negligible factor away from* $1/2$*:*

1. *$\mathcal{A}$ chooses $t' < t$ and outputs t'*
2. *For all $i = 1..t$, $\mathcal{C}$ runs $u_i \leftarrow \mathsf{Gen}(1^\lambda)$*
3. *$\mathcal{C}$ defines $V = \{u_i\}_{i=1..t'}$ and $U = \{u_i\}_{i=1..t}$*
4. *$\mathcal{C}$ computes $(k_0, H) \leftarrow \mathsf{Encap}(V, t)$*
5. *$\mathcal{C}$ samples randomly $k_1 \leftarrow \{0,1\}^\lambda$ and $b \leftarrow \{0,1\}$*
6. *$\mathcal{C}$ outputs (U, H, k_b) to $\mathcal{A}$*
7. *$\mathcal{A}$ outputs b'*

4.3 T-PSI from Phasing and T-KEM

We now describe how to construct a T-PSI scheme using the introduced building blocks, namely the Phasing subroutine, a T-KEM, and any IND-CPA secure symmetric encryption scheme (E_k, D_k) for a key k.

Definition 12 (Threshold PSI functionality). *For any T-PSI protocol, the functionality is given as:*

$$\mathsf{TPSI}(A, B) \to (\perp, Z)$$

where

$$Z = \begin{cases} A \cap B & \text{if } |A \cap B| \geq t \\ \emptyset & \text{otherwise} \end{cases}$$

That is, the sender learns nothing and the receiver learns the intersection if and only if the size of the intersection is greater than t.

Now for a concrete protocol which, as shown later, produces the above functionality. The sender and receiver, with input sets A and B respectively, start by running the Phasing subroutine to calculate U and V. Then, the sender runs TPSI^S using U and the threshold t and sends the output to the receiver. The receiver runs TPSI^R using the received input from the sender as well as V to compute the final result Z.

Procedure $\mathsf{TPSI}^S(U, t)$:
- $(U_0, U_1) \leftarrow \mathsf{split}(\mathsf{U})$
- $(k, H) \leftarrow \mathsf{Encap}(U_0, t)$
- $C \leftarrow E_k(U_1)$
- return (C, H)

Proc. $\mathsf{TPSI}^R(V, R_V, C, H)$:
- $(V_0, V_1) \leftarrow \mathsf{split}(\mathsf{V})$
- $k' \leftarrow \mathsf{Decap}(V_0, H)$
- **if** $k' = \perp$ **then** :
  - return $\emptyset$
- **else** :
  - $U_1 \leftarrow D_{k'}(C)$
  - $Z \leftarrow \mathsf{Isect}(U_1, V_1, R_V)$
  - return Z

Elements in U and V are 2λ bits. The split subroutine enables creation of U_0 and V_0 as the sets containing all the top λ bits from all the masks in U, V and U_1, V_1 to be the sets containing all the bottom λ bits from all the masks in U, V. Splitting the masks in two is needed to avoid circular security issues between the T-KEM and the encryption scheme.

Definition 13 (Phasing/T-KEM T-PSI). *The protocol for Phasing/T-KEM T-PSI proceeds as follows:*

1. *The sender with input A and receiver with input B jointly run*

$$(U, (V, R_V)) \leftarrow \mathsf{Phasing}(A, B)$$

   The sender retains U and the receiver retains (V, R_V).
2. *The sender runs*

$$(C, H) \leftarrow \mathsf{TPSI}^S(U, t)$$

   and sends (C, H) to the receiver.
3. *The receiver concludes the protocol by computing*

$$Z \leftarrow \mathsf{TPSI}^R(V, R_V, C, H)$$

   and outputting Z.

Security of T-PSI As highlighted before, the construction should respect the functionality described in Definition 12. Thus, Theorem 2 captures the security goal of our protocol.

Theorem 2 (Phasing/T-KEM security). *The protocol specified in Definition 13 securely implements the privacy-preserving threshold set intersection functionality (Definition 12) in the presence of passive adversaries*

Proof. The security of the protocol follows from the security of the building blocks. It is trivial to argue security against a passively corrupt sender, since the view of the sender in the T-PSI protocol is exactly the same as in the original Phasing protocol (the sender does not receive any extra messages).

We now argue security against a passively corrupt receiver. To do so, we need to construct a simulator that, having access to the desired input/output of the receiver (e.g., B and Z and in particular not to the sender's input A), constructs a simulated view which is computationally indistinguishable from the view of the receiver in the real protocol. The simulator is constructed with two main cases – when an intersection larger than t is known, and when it is not.

With known threshold: If $|A \cap B| \geq t$, we simply run the simulator of the Phasing protocol. In particular, the simulator runs $\mathcal{S}_R^{\text{PHG}}(B, A \cap B)$ to compute sets of masks (U, V). Now the simulator parses $U = (U_0, U_1)$ and $V = (V_0, V_1)$, computes $(k, H) \leftarrow \mathsf{Encap}(U_0, t)$, encrypts $C = E_k(U_1)$ and adds (C, H) to the simulated view. Any adversary that can distinguish between this view and the view in the real protocol can be used to break the security of the Phasing subroutine using the following reduction: the reduction chooses an arbitrary set B' such that $A \cap B = A \cap B'$ and outputs A, B' to the challenger and receives (U, V). Now the reduction completes the view of the protocol by recomputing (k, H, C) and adding (C, H) to the view, which is passed to the adversary against the protocol simulator. By the construction of Definition 9, when $b = 0$ this corresponds to the real view of the protocol execution and when $b = 1$ this corresponds to the simulated run, hence the reduction breaks Definition 9 with the same advantage as the adversary for the overall protocol.

Without known threshold: The more interesting case is of course when $|A \cap B| < t$. In this case the simulator runs the Phasing simulator $\mathcal{S}_R^{\text{PHG}}(B, \emptyset)$ and receives the masks U^*, V^* such that $U^* \cap V^* = \emptyset$. Next the simulator parses $U^* = (U_0^*, U_1^*)$ and $V^* = (V_0^*, V_1^*)$, computes $(k^*, H^*) \leftarrow \mathsf{Encap}(U_0^*, t)$, encrypts $C^* = E_{k^*}(U_1^*)$ and adds (C^*, H^*) to the simulated view. Note that the distribution of the simulated view is different from the distribution in the real protocol. In particular in the real protocol U, V are such that $|U \cap V| = |A \cap B| < t$ while in the simulation $|U^* \cap V^*| = 0$. We argue that the views are computationally indistinguishable anyway. We do so by a sequence of hybrid games, where hybrid 0 is identical to the real protocol. In the first hybrid we replace k with k^* (the simulated key), and therefore we replace $C = E_k(U_1)$ with $C' = E_{k^*}(U_1)$. Any adversary that can distinguish between these two hybrids can be used to break the security of the underlying T-KEM scheme (in the reduction the adversary here has only $|A \cap B| < t$ of the necessary masks). In the second hybrid we replace the encrypted masks U_1 with the simulated masks U_1^*. That is, in the view we replace $C' = E_{k^*}(U_1)$ with $C^* = E_{k^*}(U_1^*)$. Any adversary that distinguishes between these two hybrids can be used to break the IND-CPA security of the encryption scheme (note that the reduction here does not need to know the key). Since the second hybrid is identical to the simulated view, this concludes the proof.

□

5 Implementing T-KEM

This section describes an implementation of the T-KEM algorithms that satisfies the security requirements. Our techniques are based on polynomial interpolation, and can be seen as an extension of the well known Shamir's secret sharing [46] scheme.

Encapsulation Given a threshold t, we implement $\mathsf{Encap}(B, t)$ on an input set B with $|B| = n > t$ in the following way: first we parse every element in B as $(x_i^B, y_i^B) \in \mathbb{F} \times \mathbb{F}$ where $\mathbb{F}$ is an appropriately large (exponential) finite field. Then we define a polynomial p of degree $n-1$ from the n points known to B as $p(x_i^B) = y_i^B$ and the output key as $k = p(0)$

The hint is composed of $n - t$ additional points (x_i^h, y_i^h) with $y_i^h = p(x_i^h)$ as well as a hash of the key $ck = R(k)$ (where R is modeled as a random oracle) which allows to detect correct decapsulation:

$$H = (ck = R(k), \{(x_i^h, y_i^h)\}_{i \in \{1,\ldots,n-t\}})$$

Decapsulation The decapsulation process $\mathsf{Decap}(H, A)$ works as follows: first parse A as $(x_i^A, y_i^A) \in \mathbb{F} \times \mathbb{F}$; then, for every subset $A' \subseteq A$ with $|A'| = t$, define a polynomial p' using the points in H and in A', then compute a candidate key $k' = p'(0)$ and check if $R(k') = ck$. If yes, then output $k = k'$, otherwise proceed with the next subset A'.

The correctness follows from the basic fact that if two polynomials p, p' of degree $n-1$ agree on n points, then $p = p'$ [46].

To show security as per Definition 11, we make the following argument. Security follows from the fact that (in the random oracle model) ck can only be used to verify if $k' = k$ but otherwise leak no information about k. Consider the worst case, in which the adversary has $t' = t - 1$ correct points. Then even given the $n - t$ hints the adversary only has $n-1$ points on a $n-1$ degree polynomial, which means that the best chance of guessing k is $|\mathbb{F}|^{-1}$. Since ck can only be used to verify guesses, then the only way of distinguishing the real k_0 from the random key k_1 (in the security game of Definition 11) is to invert the random oracle, and the probability that an adversary can do this by querying the oracle $q = \mathsf{poly}(\lambda)$ times is bounded by $q/|\mathbb{F}|$ which is negligible if the size of $\mathbb{F}$ is exponential in the security parameter.

In terms of efficiency, the main bottlenecks in the solution described above is to construct the hint during encapsulation, and how to reconstruct the key k from the receiver's set and the hints during decapsulation. How to accomplish encapsulation

and decapsulation in the context of ridesharing is detailed in Appendix 1. Encapsulation is rather straight-forward with a running time of $\mathcal{O}(n^3)$. Interestingly, for the corner case when any overlap of the input sets are sequential, as is the case for ridesharing applications, decapsulation can be optimized to give an asymptotic running time of $\mathcal{O}(n^2)$.

6 Experiments

This section details experiments to assert both the effectiveness and efficiency of our two-faceted approach. First, we present a study of taxi trips in New York City. This study compares how many trips the two privacy-preserving approaches find in comparison to a plaintext implementation. Secondly, the efficiency of the two approaches are evaluated, and compared to a set intersection protocol implemented in a generic SMC framework using garbled circuits.

6.1 Study of Taxi rides

To determine the effectiveness of the privacy-preserving approaches, we have evaluated our approach on real-world data of taxi rides in New York City, as publicly provided by the New York City Taxi and Limousine Commission (TLC) [51]. We have used this data to find typical movements in a dense city, assuming the user would also be willing to travel using other means than by taxi to go from their origin to their destination. Given any two taxi rides, we assign one to be party A, who is willing to deviate from their route, and the other party B who sticks to their shortest route. TLC reveals only the origin and destination of the users. To calculate the route, we have used the open source software Routino [44], utilizing street data from OpenStreetMap [37].[2]

The taxi dataset is very large, but there are surprisingly few rides that happen at the same time such that the users could have shared a cab in practice. We conjecture that the paths computed from the taxi trips are realistic movement patterns, but that the times the rides take place are likely widely different from users prone to use ride-sharing services in order to e.g. commute. Even though taxi trips in New York City do not generalize to mobility patterns in general (it is even likely that users of the yellow non-bookable cabs differ from users of the green pre-booked cabs within New York City alone), this data at least give some indicative intuition for ridesharing

[2] Map data copyrighted OpenStreetMap contributors and available from http://www.openstreetmap.org

in a dense city. Thus, in this study we opt to not look at the time of day at all, and instead view each file in the dataset as a time-frame in which ridesharing is feasible from a timing perspective. Files are grouped month by month, which means we effectively get a relative measure between the different patterns which is roughly averaged over time of day, and weekday, and so forth. Though the magnitude of ridesharing opportunities calculated in the study is an over-approximation, the goal is not to find how many ridesharing opportunities exist, but which pattern detects the most ridesharing opportunities.

As the TLC dataset is vast, the "bruteforce" algorithm (outlined in Section 3) is the bottleneck in the comparison. To make the comparison feasible, only 1000 trips from every even month of 2015 for the green cab dataset was used. The dataset was considered as asymmetric, such that both parties can assume the role of A or B, giving roughly one million possible ridesharing opportunities every month and a total of 6 million for all months.

The number of ridesharing opportunities found when using intersection-based matching, endpoint-based matching, and when brute-forcing all points according to the "bruteforce" algorithm outlined were counted. The bruteforce algorithm captures all rides caught by the model. We assume that for this dataset, users are willing to deviate most the start and end of their trajectory, and not at all in the middle. The intuition is that the user familiar to the areas close to home and/or work, but less comfortable with improvising further along the route. A second-degree polynomial was used to implement the deviation function Δ, as shown in Equation 3.

$$\Delta_T(i) = 4i^2 \frac{r_0}{|T|^2} - 4i \frac{r_0}{|T|} + r_0 \tag{3}$$

In Equation 3, r_0 is how much the user is willing to deviate at the endpoints, and i is the index of a vertex such that $\forall i : v_i \in T$. This gives a "happy-smiley-face" curve with root at $|T|/2$ with $y = r_0$ at $x = 0$ and $x = |T|$. Note that the indices of the positions rather than the positions themselves does lead to some imprecision. But Equation 3 gives an intuitive estimation with the right sign on the derivate in almost all points.

Table 1 illustrates the effectiveness of the two different approaches on this particular dataset. In the table, t says what percentage of the trip must be shared for ridesharing to be deemed feasible, and r_0 is the deviation allowed at the start and endpoints (with $\Delta_T(\cdot)$ as per Equation 3). It shows the number of found ridesharing opportunities by both approaches in PrivatePool (PP), intersection-based matching (IS) and endpoint-based matching (EP) compared to capturing all opportunities when applying the entire model. The results show that effectiveness is heavily re-

liant on the parameters, ranging from over 92% to 14% when utilizing both approaches, 11% to 0.3% for endpoint matching, and 92% to 2.5% when using only intersection. PrivatePool thus captures the vast majority as covered by the model, but which privacy-preserving approach to use depends on the needs of the user. Even though the highest results were achieved with intersection-based matching, the experiments confirm that endpoint matching is sometimes preferable to the intersection-based approach.

Table 1. Effectiveness of privacy-preserving approaches (in percent)

t	$r_0 = 500$			$r_0 = 1000$			$r_0 = 2000$		
	PP	IS	EP	PP	IS	EP	PP	IS	EP
20%	92.6	92.29	0.31	61.3	59.98	1.31	36.91	31.92	4.99
50%	75.57	74.64	0.93	45.73	42.3	3.43	25.26	16.01	9.25
80%	28.76	26.24	2.52	15.06	9.26	5.8	13.57	2.48	11.09

6.2 Benchmarks

To benchmark the different approaches, the two patterns as outlined in Section 2 were investigated, again using OSM and Routino. The shortest path for several likely routes from intra-city rides to trips over 1000 km, were used to determine the size of the input set to use for the benchmarks. As seen in Table 2, the standard deviation is very large, meaning that the road sections are of very varying size. Further, it is visible that for shorter trips, the road segments are shorter, likely because a highway contains long road sections.

The implementations measure only the CPU time, and leaves out I/O operations entirely. Benchmarks were done on a single machine with 16GB RAM and an Intel i7-4790 CPU at 3.60GHz.

Implementation details For the different approaches, separate implementations were used. For endpoint matching, the implementation is built on top of the original python code for proximity testing by Hallgren et al., whereas for the T-PSI a new C++ prototype for T-KEM was developed for integration into the work by Pinkas et al.[40], but should also be compatible with the most efficient Phasing subroutine from the work of Kolesnikov et al. [26]. For a comparison to generic circuit-based approaches, a Java implementation (called FastGC) for PSI by Huang et al. [17] was

Table 2. Road segment lengths from OpenStreetMap data

Number of sections	Distance(m)	Average (m)	Standard deviation (m)
230	12013	52.46	81.87
123	19801	162.30	160.13
626	143165	229.06	217.20
1856	610581	329.15	285.69
2420	623177	257.62	231.94
2142	735975	343.75	305.62
3440	931475	270.86	308.35
2868	974107	339.77	299.89

used. For each implementation there are different perks and drawbacks, and in many ways the comparison is unfair (e.g. by programming language). However, regarding performance the results firmly position the different solutions asymptotically relative to each other. Exactly how they are configured is detailed in the following.

For the end-point matching, we are able to achieve very good results for coarse grained precision. The implementation is using 1024 bit keys for the Paillier cryptosystem, and a "radius value" of 4, with an imprecision of 500 meters. This means that proximity is checked in a 2 km range, where however both false negatives and false positives are possible up to 500 meters.

As the data received from OpenStreetMap does not correspond to equal-sized chunks (road sections have different lengths), the data needs to be processed to be a good fit for the two PSI solutions. For an application that needs to be precise in the intersection threshold t, the road sections would need to be split into sections as long as the distance unit of OpenStreetMap. This would greatly increase the size of the input set for the PSI solutions. For the scope of this work, these smaller input sizes suffices to evaluate the efficiency relative to the other approaches, and as will be seen below the used inputs are enough to see roughly for which size of the input the different solutions are useful. For our benchmarks, the input set corresponds to using the start coordinate of each road section. Though this much imprecision often would cause the intersection to be either half or twice as long as the intended threshold, plaintext preprocessing can be used to attune precision and mitigate this issue once parameters are fixed for a certain application.

The T-KEM implementation uses $\lambda = 128$, utilizing the Intel Intrinsic [12] instruction set for operations in a field of size 2^{128} and t was set to 80% of the total set size. On the sender side, parallelization is trivial, and the sender thus computes hints in parallel using OpenMP [36]. Though the benchmarks shown do not include the

Phasing subroutine, the full original protocol needs under 0.3 seconds for 4096-bit set sizes [40], rendering it almost negligible in the comparison.

The GC solution for PSI uses 80-bit security and 32-bit strings for the input elements. For the scope of this work it was chosen to not include the threshold computation. The addition of the extra comparison should yield significantly slower results for the generic solution. However, the asymptotic result shown in the following section should not change – for large sets the generic solution should outperform our threshold version. To determine how much larger the set needs to be for a threshold GC solution to outperform our T-PSI protocol is left for future work.

Notably, there are more efficient implementations for generic SMC which should be used for further investigations. The recent work by Pinkas et al. [41] show results which are significantly better than those of Huang et al. With their setup for 128-bit security on a LAN network, a set intersection between 256 elements was performed in 0.304 seconds and 4096 elements in 1.647 seconds, which means that they may outperform our T-PSI solution also for very small input sets.

Results Benchmarks using the three prototypes show that all are feasible enforcements in some situations, see Figure 7. Table 3 shows the average time taken to check for ridesharing opportunities using endpoint-matching, T-PSI, and FastGC set intersection. Experiments were run 25 times to minimize noise, with a less than 10% standard deviation.

Table 3. Benchmarks (seconds)

Number of	End-point	T-PSI			FastGC
sections	match (r=4)	Sender	Receiver	Total	PSI
32	0.352	0.014	0.008	0.022	2.485
64	0.367	0.046	0.031	0.077	2.915
128	0.361	0.174	0.187	0.362	3.918
256	0.363	0.695	1.388	2.083	5.593
512	0.372	2.778	10.79	13.57	8.665
1024	0.354	11.12	85.67	96.78	15.100
2048	0.353	44.34	683.8	728.1	30.263
4096	0.371	176.8	5450	5627	62.154

The end-point matching performs independently of the size of the set, however suffers greatly with improved precision. At $r = 25$ (80 meter precision) the implementation finishes in 9.2 seconds, and at $r = 100$ (20 meter precision) in 124.43

seconds. At this stage, the derivate tapers off (for small r the increase in running time is close to quadratic, for large r it's close to linear), but large values of r are out of reach for practical applications with the chosen approach.

For the intersection-based solutions, the FastGC implementation is only preferable over T-PSI for very large sets, and though it outperforms our threshold variant for paths longer than 500 items as per Table 3, both implementations are arguably too slow to be used in practice at this stage. Thus, the T-PSI implementation is preferable over FastGC for any application where the user cannot tolerate large delays, such as consumer-facing applications. Overall, the benchmarks are very encouraging results for using SMC techniques. Though indeed some configurations are very impractical, for most settings there are alternatives that yield satisfying results. The endpoint matching and intersection approaches are both very efficient for low-precision applications in the general case, and T-PSI can be used with full precision for shorter trips. Both could be used in many applications, and for any consumer-facing such as ridesharing they outperform generic techniques by orders of magnitude.

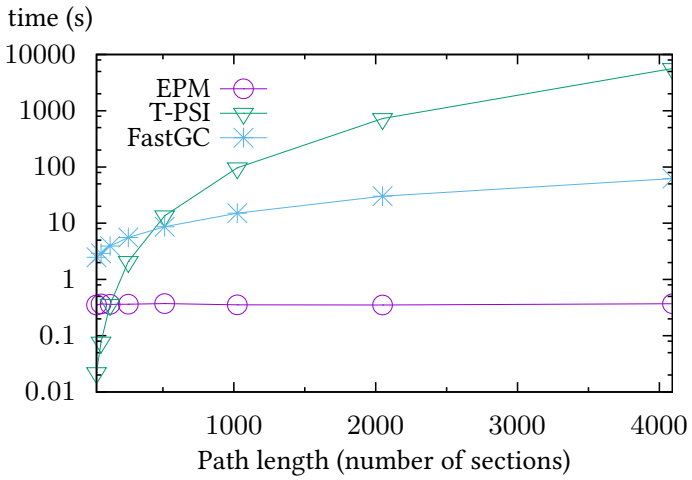

Fig. 7. Visualization of the benchmark results

Much important work within SMC is of a more theoretical nature, and is thus not able to show that SMC is applicable to at least some real-world scenarios, e.g. using T-PSI for short trips. These experiments show how to use SMC as a part of a product in the emerging area of digital transportation services. Indeed, though all evaluated approaches need to use a degraded precision in the general case, the achieved running time is well within reach of practical applications with acceptable precision for many settings.

7 Related work

As discussed in recent surveys [28, 50], location privacy is an increasingly important topic. Relating to the privacy of a user's trajectory, much prior research relates to re-identification [8, 4], whereas our work focuses on minimizing disclosure of location data. The focus of hiding the identity of a user stems from the common fact that many existing parties has a trace of a user's whereabouts and wants to publish statistical facts about crowds. E.g. a provider of public transport which uses electronic tickets often knows where a user enters and exits a vehicle, and might want to contribute anonymized data for urban planning [23]. Instead, this work focuses on the case where a service provider, who anyways needs to know the identity of their customers, wants to minimize the data disclosure to e.g. reduce the risk of security breaches or to conform to emerging regulations, such as the data protection regulation by the European Union [52].

There is extensive literature on the problem of how to test for the proximity of two points as provided by two different users, without revealing more than the proximity result [57, 48, 47, 7, 32, 35, 45, 15]. However, most work considers a single proximity result in isolation. There are some exceptions, such as the work by Hallgren et al. [16] to account for moving parties. In the spirit of Hallgren et al., we utilize multiplication of two proximity results to check whether both are positive. There are several ad-hoc solutions which are not amenable to this approach [48, 47, 7, 32]. Further work uses additive homomorphic encryption for location proximity [57, 35, 45, 15].

The first problem considered for SMC was the millionaires problem, where two parties wanted to find which was the greater of two integers [56], from which equality testing [5] and subsequently set intersection are natural steps [6]. Set intersection is an important primitive needed to build many larger applications, as discussed in several works [17, 40, 39]. Set intersection has been implemented both using homomorphic encryption [18, 22], garbled circuits [17], and ad-hoc solutions [40, 39, 26]. Of these, the ad-hoc solutions perform better than the other techniques for small input sizes, with garbled circuit-based solutions being asymptotically slower for large input sets [40].

8 Conclusions

We have presented an approach to specifying and enforcing privacy in ridesharing applications. Our investigation of ridesharing patterns has elucidated the benefits

of a two-fold approach: (i) ride matching based on the proximity of start- and endpoints of the rides and (ii) ride matching based on the overlap of the ride trajectories. We have therefore developed privacy-preserving mechanisms for (i) start/end point matching and (ii) private trajectory matching. For the former, we have built on recent work on location proximity based on homomorphic encryption. For the latter, we have designed a novel protocol for threshold private set intersection (dubbed T-PSI), based on Shamir's secret sharing scheme.

We have evaluated the effectiveness of our approach on the real-world data from the New York City Taxi and Limousine Commission, confirming that the endpoint-based and intersection-based mechanisms are both useful.

We have prototyped and benchmarked these mechanisms and contrasted them with a general-purpose approach based on garbled circuits. The evaluation shows that for any application that requires termination under 10 seconds, both of our mechanisms outperform the generic garbled-circuit approach. The benchmarks also indicate that precision can be traded to achieve running time in under a second using either homomorphic encryption or the T-PSI protocol.

Future Work Our results open up for promising future work tracks. On the theoretical side, we plan to investigate multi-key protocols for ridesharing. If successful, this will allow us to boost the scalability of the approach by significantly simplifying the key distribution phase. On the practical side, we plan to build a fully-fledged ridesharing app, drawing on the infrastructure developed in our prototype.

Moreover, we're keen to conduct further investigation to find more efficient schemes for T-PSI. The simple scheme outlined in the following could outperform our solution based on T-KEM, where Alice has a trajectory T^A of size m and Bob a trajectory T^B of size n, for any given a threshold t.

Note that this solution only works when the input data is sorted as in our case for ridesharing. We need that if $\exists i, j : T_i^A = T_j^B \wedge T_{i+t}^A = T_{j+t}^B$ then it also holds that $\forall u \in \{i, \ldots, i+t\}, v \in \{j, \ldots, j+t\} : T_u^A = T_v^B . \forall u \in \{0, \ldots, t\} : T_{i+u}^A = T_{j+u}^B$. The construction proceeds as follows:

1. Alice prepares a set on the following form,

$$(T_1^A, T_t^A), (T_2^A, T_{t+1)}^A, \ldots, (T_{m-t+1}^A, T_m^A)$$

2. Bob similarly prepares a set on the form

$$(T_1^B, T_t^B), (T_2^B, T_{t+1)}^B, \ldots, (T_{n-t+1}^B, T_n^B)$$

3. The parties run a standard PSI protocol on these sets.

Each element is a pair in the original set denoting a segment of length t. Thus, if the PSI returns at least one element, the parties may rideshare along the segment represented by this element. It's easy to see that shared segments of length less than t are not disclosed. All that remains is to find the maximum segment to use for ridesharing, but this can be done in the plain by the receiver of the PSI by taking the earliest and latest coordinate in any returned pair.

Acknowledgments Thanks are due to Peter Druschel and Rijurekha Sen for inspiring discussions on the scenario of private ridesharing. This work was partly funded by the European Community under the ProSecuToR project, COST Action IC1306, Danish Independent Research Council and the Swedish research agency VR.

References

1. C. Bessette, "Does Uber Even Deserve Our Trust?" http://www.forbes.com/sites/chanellebessette/2014/11/25/does-uber-even-deserve-our-trust/, Nov. 2014.
2. "BlaBlaCar - Trusted carpooling," https://www.blablacar.com/.
3. D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols (extended abstract)," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, J. Simon, Ed. ACM, 1988, pp. 11–19. [Online]. Available: http://doi.acm.org/10.1145/62212.62214
4. R. Chen, B. C. M. Fung, and B. C. Desai, "Differentially private trajectory data publication," *CoRR*, vol. abs/1112.2020, 2011. [Online]. Available: http://arxiv.org/abs/1112.2020
5. R. Fagin, M. Naor, and P. Winkler, "Comparing information without leaking it," *Commun. ACM*, vol. 39, no. 5, pp. 77–85, 1996. [Online]. Available: http://doi.acm.org/10.1145/229459.229469
6. M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 1–19. [Online]. Available: \url{http://dx.doi.org/10.1007/978-3-540-24676-3_1}
7. D. Freni, C. R. Vicente, S. Mascetti, C. Bettini, and C. S. Jensen, "Preserving location and absence privacy in geo-social networks," in *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, and A. An, Eds. ACM, 2010, pp. 309–318. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871480
8. G. Ghinita, "Private queries and trajectory anonymization: a dual perspective on location privacy," *Trans. Data Privacy*, vol. 2, no. 1, pp. 3–19, 2009. [Online]. Available: http://www.tdp.cat/issues/abs.a018a09.php

9. A. A. Ghorbani, V. Torra, H. Hisil, A. Miri, A. Koltuksuz, J. Zhang, M. Sensoy, J. García-Alfaro, and I. Zincir, Eds., *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*. IEEE, 2015. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7193739
10. O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
11. O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, A. V. Aho, Ed. ACM, 1987, pp. 218–229. [Online]. Available: http://doi.acm.org/10.1145/28395.28420
12. S. Gueron and M. E. Kounavis, "Intel® carry-less multiplication instruction and its usage for computing the gcm mode," https://software.intel.com/sites/default/files/managed/72/cc/clmul-wp-rev-2.02-2014-04-20.pdf, Apr. 2014.
13. S. Halevi, Y. Lindell, and B. Pinkas, "Secure computation on the web: Computing without simultaneous interaction," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, ser. Lecture Notes in Computer Science, P. Rogaway, Ed., vol. 6841. Springer, 2011, pp. 132–150. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22792-9_8
14. P. A. Hallgren, M. Ochoa, and A. Sabelfeld, "Bettertimes - privacy-assured outsourced multiplications for additively homomorphic encryption on finite fields," in *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*, ser. Lecture Notes in Computer Science, M. H. Au and A. Miyaji, Eds., vol. 9451. Springer, 2015, pp. 291–309. [Online]. Available: \url{http://dx.doi.org/10.1007/978-3-319-26059-4_16}
15. ——, "Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol," in *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, A. A. Ghorbani, V. Torra, H. Hisil, A. Miri, A. Koltuksuz, J. Zhang, M. Sensoy, J. García-Alfaro, and I. Zincir, Eds. IEEE, 2015, pp. 1–6. [Online]. Available: \url{http://dx.doi.org/10.1109/PST.2015.7232947}
16. ——, "Maxpace: Speed-constrained location queries," in *2016 IEEE Conference on Communications and Network Security, CNS 2016, Philadelphia, PA, USA, October 17-19, 2016*, 2016.
17. Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. [Online]. Available: http://www.internetsociety.org/private-set-intersection-are-garbled-circuits-better-custom-protocols
18. B. A. Huberman, M. K. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *EC*, 1999, pp. 78–86. [Online]. Available: http://doi.acm.org/10.1145/336992.337012

19. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, 2003, pp. 145–161. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45146-4_9
20. J. Jung and T. Holz, Eds., *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*. USENIX Association, 2015. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15
21. F. Kerschbaum, "Adapting privacy-preserving computation to the service provider model," in *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, Vancouver, BC, Canada, August 29-31, 2009*. IEEE Computer Society, 2009, pp. 34–41. [Online]. Available: http://dx.doi.org/10.1109/CSE.2009.261
22. ——, "Outsourced private set intersection using homomorphic encryption," in *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, H. Y. Youm and Y. Won, Eds. ACM, 2012, pp. 85–86. [Online]. Available: http://doi.acm.org/10.1145/2414456.2414506
23. H. Kikuchi and K. Takahashi, "Zipf distribution model for quantifying risk of re-identification from trajectory data," in *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, A. A. Ghorbani, V. Torra, H. Hisil, A. Miri, A. Koltuksuz, J. Zhang, M. Sensoy, J. García-Alfaro, and I. Zincir, Eds. IEEE, 2015, pp. 14–21. [Online]. Available: http://dx.doi.org/10.1109/PST.2015.7232949
24. L. Kissner and D. X. Song, "Private and threshold set-intersection," http://www.dtic.mil/dtic/tr/fulltext/u2/a461119.pdf, 2004.
25. ——, "Privacy-preserving set operations," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 241–257. [Online]. Available: http://dx.doi.org/10.1007/11535218_15
26. V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious PRF with applications to private set intersection," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 818–829. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978381
27. V. Kolesnikov, A. Sadeghi, and T. Schneider, "From dust to dawn: Practically efficient two-party secure function evaluation protocols and their modular design," *IACR Cryptology ePrint Archive*, vol. 2010, p. 79, 2010. [Online]. Available: http://eprint.iacr.org/2010/079
28. J. Krumm, "A survey of computational location privacy," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, 2009.
29. J. Krumm and E. Horvitz, "LOCADIO: inferring motion and location from wi-fi signal strengths," in *1st Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2004), Networking and Services, 22-25 August 2004, Cambridge,*

MA, USA. IEEE Computer Society, 2004, pp. 4–13. [Online]. Available: http://dx.doi.org/10.1109/MOBIQ.2004.1331705

30. Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *IACR Cryptology ePrint Archive*, vol. 2008, p. 197, 2008. [Online]. Available: http://eprint.iacr.org/2008/197
31. "Lyft," https://www.lyft.com/.
32. 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.*, vol. 20, no. 4, pp. 541–566, 2011. [Online]. Available: \url{http://dx.doi.org/10.1007/s00778-010-0213-7}
33. Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, J. Jung and T. Holz, Eds. USENIX Association, 2015, pp. 785–800. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/michalevsky
34. K. Muthukrishnan, B. van der Zwaag, and P. J. M. Havinga, "Inferring motion and location using WLAN RSSI," in *Mobile Entity Localization and Tracking in GPS-less Environnments, Second International Workshop, MELT 2009, Orlando, FL, USA, September 30, 2009. Proceedings*, ser. Lecture Notes in Computer Science, R. Fuller and X. D. Koutsoukos, Eds., vol. 5801. Springer, 2009, pp. 163–182. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04385-7_12
35. 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. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/11/pdf/1_3.pdf
36. OpenMP Architecture Review Board, "OpenMP application program interface version 4.0," May 2013. [Online]. Available: http://www.openmp.org/wp-content/uploads/OpenMP4.0.0.pdf
37. "Openstreetmap," http://www.openstreetmap.org/.
38. W. R. Ouyang, A. K. Wong, and C. A. Lea, "Received signal strength-based wireless localization via semidefinite programming: Noncooperative and cooperative schemes," *IEEE Trans. Vehicular Technology*, vol. 59, no. 3, pp. 1307–1318, 2010. [Online]. Available: http://dx.doi.org/10.1109/TVT.2010.2040096
39. B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, J. Jung and T. Holz, Eds. USENIX Association, 2015, pp. 515–530. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas
40. B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *Proceedings of the 23rd USENIX Security Symposium, San Diego,*

CA, USA, August 20-22, 2014., K. Fu and J. Jung, Eds. USENIX Association, 2014, pp. 797–812. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas

41. ——, "Scalable private set intersection based on OT extension," *IACR Cryptology ePrint Archive*, vol. 2016, p. 930, 2016. [Online]. Available: http://eprint.iacr.org/2016/930
42. I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis, "Where's wally?: Precise user discovery attacks in location proximity services," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, I. Ray, N. Li, and C. Kruegel, Eds. ACM, 2015, pp. 817–828. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813605
43. P. Rindal and M. Rosulek, "Improved private set intersection against malicious adversaries," in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, ser. Lecture Notes in Computer Science, J. Coron and J. B. Nielsen, Eds., vol. 10210, 2017, pp. 235–259. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-56620-7_9
44. "Routino," http://www.routino.org/.
45. J. Sedenka and P. Gasti, "Privacy-preserving distance computation and proximity testing on earth, done right," in *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, S. Moriai, T. Jaeger, and K. Sakurai, Eds. ACM, 2014, pp. 99–110. [Online]. Available: http://doi.acm.org/10.1145/2590296.2590307
46. A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176
47. L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu, "Private and flexible proximity detection in mobile social networks," in *Eleventh International Conference on Mobile Data Management, MDM 2010, Kanas City, Missouri, USA, 23-26 May 2010*, T. Hara, C. S. Jensen, V. Kumar, S. Madria, and D. Zeinalipour-Yazti, Eds. IEEE Computer Society, 2010, pp. 75–84. [Online]. Available: \url{http://dx.doi.org/10.1109/MDM.2010.43}
48. L. Siksnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen, "A location privacy aware friend locator," in *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009, Aalborg, Denmark, July 8-10, 2009, Proceedings*, ser. Lecture Notes in Computer Science, N. Mamoulis, T. Seidl, T. B. Pedersen, K. Torp, and I. Assent, Eds., vol. 5644. Springer, 2009, pp. 405–410. [Online]. Available: \url{http://dx.doi.org/10.1007/978-3-642-02982-0_29}
49. T. Sohn, A. Varshavsky, A. LaMarca, M. Y. Chen, T. Choudhury, I. E. Smith, S. Consolvo, J. Hightower, W. G. Griswold, and E. de Lara, "Mobility detection using everyday GSM traces," in *UbiComp 2006: Ubiquitous Computing, 8th International Conference, UbiComp 2006, Orange County, CA, USA, September 17-21, 2006*, ser. Lecture Notes in Computer Science, P. Dourish and A. Friday, Eds., vol. 4206. Springer, 2006, pp. 212–224. [Online]. Available: http://dx.doi.org/10.1007/11853565_13

50. M. Terrovitis, "Privacy preservation in the dissemination of location data," *SIGKDD Explorations*, vol. 13, no. 1, pp. 6–18, 2011.
51. The City of New York, "Taxi and Limousine Commission trip data," http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml, 2016.
52. The European Parliament and the Council of the European Union, "Article 25: Data protection by design and by default," http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf, Apr. 2016.
53. "Uber technologies inc." https://www.uber.com/.
54. Volvo Car Group, "Volvo cars and uber join forces to develop autonomous driving cars," https://www.media.volvocars.com/global/en-gb/media/pressreleases/194795/volvo-cars-and-uber-join-forces-to-develop-autonomous-driving-cars, 2016.
55. Wikipedia, "Tesla model 3 — wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Tesla_Model_3&oldid=750392389, 2016.
56. A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. IEEE Computer Society, 1986, pp. 162–167. [Online]. Available: \url{http://dx.doi.org/10.1109/SFCS.1986.25}
57. G. Zhong, I. Goldberg, and U. Hengartner, "Louis, lester and pierre: Three protocols for location privacy," in *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, ser. Lecture Notes in Computer Science, N. Borisov and P. Golle, Eds., vol. 4776. Springer, 2007, pp. 62–76. [Online]. Available: \url{http://dx.doi.org/10.1007/978-3-540-75551-7_5}

1 Interpolation optimizations

Before describing our optimizations, we recall the basics of polynomial interpolation.

For any given set of n coordinates, there exists a unique interpolating polynomial of degree $n+1$. When using a polynomial over a field, as in our context, an attacker gains no advantage from knowing $n-1$ points. *Lagrange interpolation* is used to evaluate a polynomial $p(x)$ of degree $n+1$ for any x, given n arbitrary points. For the input set (x_i, y_i), with $i \in \{0, 1, .., n\}$, we have:

$$p(x) = \sum_{j=0}^{n} \left(y_j \prod_{m=0, m \neq j}^{n} \frac{x_m - x}{x_m - x_j} \right)$$

From the above representation, it's easy to see that Lagrange interpolation requires $\mathcal{O}(n^2)$ time.

Decapsulation Here we exploit the following property of our specific application: trajectories are ordered sets of points, we do not need to try *every* set A' of size t, but only sets with consecutive points. In detail, if two trajectories (x_i^A, y_i^A) and (x_j^B, y_j^B) only have an intersection of size t if there exist a starting point s and a difference d such that $(x_{s+i}^A, y_{s+i}^A) = (x_{s+d+i}^B, y_{s+d+i}^A)$ for all $i = 0..t-1$.

For simplicity, consider the case where A contains $n = 2t$ points, and we need to find a sequence of t consecutive points that yields the correct interpolation. There's a total of $t+1$ sequences of the correct length, and A needs to attempt to verify each of them towards the check. Let each such sequence's dimension be denoted separately as

$$\overrightarrow{X^i} = \{x_i^A, x_{i+1}^A, .., x_{i+t}^A\} \cup \{x_0^h, x_1^h, .., x_t^h\}$$

$$\overrightarrow{Y^i} = \{y_i^A, y_{i+1}^A, .., y_{i+t}^A\} \cup \{y_0^h, y_1^h, .., y_t^h\}$$

For the first sequence, Decap computes a normal interpolation in $\mathcal{O}(n^2)$ time. For all following sequences, the interpolation at the x coordinates for X_k can be used to compute the interpolation of X_{k+1}. As shown in the following, this incremental step can be done in $\mathcal{O}(n)$ time. Note that the receiver is only interested in learning $p(0)$. The receiver saves the numerators and denominators from first run in two vectors $\overrightarrow{O^0}$ (over), $\overrightarrow{U^0}$ (under), given as:

$$\overrightarrow{O^0} = \left[\prod_{m=0, m\neq j}^{t} X_m^0 \,\middle|\, j \in \{0..t\} \right] \tag{4}$$

$$\overrightarrow{U^0} = \left[\prod_{m=0, m\neq j}^{t} \left(X_m^0 - X_j^0\right) \,\middle|\, j \in \{0..t\} \right] \tag{5}$$

Since each O_i^0 and U_i^0 take $\mathcal{O}(n)$ time to compute, Equation 4 and Equation 5 are both clearly $\mathcal{O}(n^2)$. The following describes how to compute O_i^{k+1} and U_i^{k+1} from O_i^k and U_i^k in linear time, for any $k \geq 0$. The transformation from $\overrightarrow{O^k}$ to $\overrightarrow{O^{k+1}}$ consists of two steps, the first for $j \in \{0..t-2\}$ and the second for $j \in \{t, .., n-1\}$. For the case when O_{t-1}^{k+1}, we note that is exactly the value O_0^k. For $j \in \{0..t-2\}$, the numerator is calculated as:

$$O_j^{k+1} = O_{j+1}^k \cdot \frac{X_t^{k+1}}{X_0^k}$$

Which intuitively means just including one new factor $X^{k+1}t$ and removing the factor X_0^k. E.g., for $k = 0$, this means removing the first x-coordinate x_0, and including x_{t+1}. For $j \in \{t..n-1\}$, a similar computation is carried out:

$$O_j^{k+1} = O_j^k \cdot \frac{X_t^{k+1}}{X_0^k}$$

Clearly, this is constant time for each of the n indexes, which means that the time for updating from $\overrightarrow{o}^k$ to $\overrightarrow{o}^{k+1}$ is $\mathcal{O}(n)$. For the denominators, the computations are more expensive, however the asymptotic runtime is the same. Similarly as for numerators, we have that for $j \in \{0..t-2\}$:

$$U_j^{k+1} = U_{j+1}^k \cdot \frac{X_t^{k+1} - x_j^{k+1}}{x_0^k - x_j^{k+1}}$$

For $j \in \{t..n-1\}$:

$$U_j^{k+1} = U_j^k \cdot \frac{X_t^{k+1} - X_j^{k+1}}{X_0^k - X_j^{k+1}}$$

As with the numerators, this is constant time per j, and thus runs in $\mathcal{O}(n)$. Now for the value of U_{t-1}^{k+1}, we have to recompute the entire denominator.

$$U_{t-1}^{k+1} = \prod_{m=0, m\neq j}^{t} \left(X_m^{k+1} - X_j^{k+1}\right) \Big| \, j \in \{0..t\}$$

This means that the update for the denominators take $\mathcal{O}(n+n) = \mathcal{O}(n)$. Further, for each $k \in \{0..t\}$, after $\overrightarrow{O^k}$ and $\overrightarrow{U^k}$ have been computed, the evaluation of the polynomial is needed. This is done as:

$$p(0) = \sum_{j=0}^{t} \left(y_j \frac{o_j}{u_j} \right)$$

Which runs in $\mathcal{O}(n)$ time. Thus, for $k = 0$ interpolation requires $\mathcal{O}(n^2 + n) = \mathcal{O}(n^2)$ work, which is followed by t evaluations that run in $\mathcal{O}(n)$ time. In total, the receiver thus spend $\mathcal{O}(n^2 + n^2) = \mathcal{O}(n^2)$ time.

Encapsulation The encapsulation algorithm Encap can not utilize incremental computation during the interpolation process. However, the denominator is in this case fixed, and only needs to be computed once.

$$\overrightarrow{u} = \left[\prod_{m=0, m\neq j}^{t} \left(x_m^B - x_j^B\right) \middle| j \in \{0..t\} \right]$$

The numerator needs complete recomputation for every hint:

$$\overrightarrow{O^x} = \left[\prod_{m=0, m\neq j}^{t} \left(x_m^B - x\right) \middle| j \in \{0..t\} \right]$$

It's possible to achieve a speedup for the numerator by precomputing

$$\overrightarrow{P} = \left[\frac{y_j}{u_j} \middle| j \in \{0..t\} \right]$$

Then, to compute the y-component of every hint, the sender computes

$$p(x_i^h) = y_i^h = \sum_{j=0}^{t} \left(P_j O_j^{x_i^h} \right)$$

However, the improvement of precomputing P does not improve the running time asymptotically.