Location Privacy via Actively Secure Private Proximity Testing

Janus Dam Nielsen, Jakob Illeborg Pagter, and Michael Bladt Stausholm
The Alexandra Institute, Aarhus, Denmark
Email: {janus.nielsen,jakob.i.pagter,michael.stausholm}@alexandra.dk

***Abstract*—We present a solution which improves the level of privacy possible in location based services (LBS). A core component of LBS is proximity testing of users. Alice wants to know if she is near to Bob (or generally some location). The presented solution support private proximity testing and is actively secure meaning it prevents a number of attacks possible in existing protocols for private proximity testing. We demonstrate that the improved security provided only implies a factor of two penalty on execution time compared to an existing passively secure protocol.**

We also provide a security analysis and discuss the relevance of secure multiparty computation for location based services.

***Keywords*-Social network services, Privacy, Security, Encryption, Cryptographic protocols**

I. Introduction

The geographic location of a person is valuable information in online social networks since it transmits a lot of social information. The places we visit describes who we are in an even more direct way than any statements of political views, taste, rants, family, experience, particular news stories, books we (claim) to read, etc. typically found on social networks, because we actually took the *time* and *effort* to physically move us there. In contrast to the little effort involved in a Twitter message or a Facebook update. Based on the locations visited by a person one may actually deduce some of the other characteristics like where do we shop for what, who do we like to spend time with (and also not), or even political views by attending political meetings.

The geographic location can hence be an *important* and *valuable* aspect of a persons *social life*. And as such should be treated with *caution*, since disclosing it might have unforeseen consequences. Visiting certain locations or persons may be seen as offensive or socially unacceptable to some of your online "friends". You might not want to share with your boss that you are going for a job interview in another organization.

There exists a number of location based services (LBS) providing location sharing. This includes Google Latitude, Facebook places, Foursquare, Loopt, and a large number of smartphone applications.

In recent years there has been considerable research on privacy in LBS. The fundamental problem seem to be that few people would like even their closest friends to know their location all the time, yet will allow distant acquaintances to know their location some of the time.

Many existing solutions allows the user to compose a privacy policy for the sharing of location based on different policy elements like providing access to certain friends or only disclosing your location at certain places. Some policy elements can easily be realized without compromising the location. Others however, are easiest realized by sharing the location with the LBS provider, which then enforces the policy.

Besides the privacy threat from friends, acquaintances, and potentially complete strangers, it is clear that any LBS based upon a centralized server has another potentially large privacy problem, namely the massive collection of data. It will typically be quite easy to deduce the actual identities of different users. As demonstrated by the case of Malte Spitz[1].

In this work we focus on privacy features/principles, where the decision to divulge your location is a function of information you have combined with information others have - in particular, information that you may not want to disclose unless the decision to disclose your location is positive. The most basic feature we can think of is proximity testing, where the decisions to disclose your position is based on your position itself. The basic idea is that if you are "close" to a friend, her knowing your location becomes more relevant because you are more likely to meet physically (or can in fact, arrange physical meetings.).

Our method is based on secure multiparty computation (SMC), which allows a number of parties to jointly compute some function in such a manner that each party does not learn the inputs of the other parties. We use this to prevent two parties from learning the other party's location unless they are in proximity to each others. And also to prevent third parties like the provider of the LBS to learn the location. All of this with strong cryptographic guaranties.

In the next section we present a security analysis of the problem domain including a discussion on the applicability of SMC. Section III discuss proximity testing. In Section IV we describe the cryptographic protocol which computes if two users are in proximity. We evaluate and discuss the protocol by benchmarking a concrete implementation on actual devices in Section V. A description of related work is given in Section VI, and Section VII presents our conclusions along with future work.

[1]http://www.zeit.de/datenschutz/malte-spitz-data-retention

II. Security Analysis

A natural solution for ensuring privacy features is of course to use a trusted server, however as a consequence of the centralized server threat it is natural to look for solutions, in which the server does not learn your location or other private information. The simplest way to achieve this, is to simply not involve a server. Therefore the problem reduces to two entities, wishing to decide if the distance between them is lower than a treshold, without revealing their actual location unless it is below the threshold. The canonical way to solve this problem is using the ideas behind Yao's millionaires problem [1]: Two party computation, a special case of SMC. In the following we briefly discuss how to apply SMC to some other privacy features in LBS—besides proximity testing—for which a server based solution would otherwise be an obvious choice.

Plausible Deniability (PD): An important feature for location based service privacy is PD [2]. PD is the ability to make a white lie about your position without others knowing about it. PD is in itself not something which can be adressed using SMC, but if you want to support it, you cannot automatically reveal your location. For proximity testing, it means setting your threshold to zero (or so low that people would meet you physically anyway).

Access Control (Who-When-Where): The authors in [3] suggest that a user should be able to specify an access control policy based on users and groups of users, times and places. At the same time they follow the advice of [4] to provide auditing functionality, so that the users can see who has requested access and why it was granted or rejected. One could argue that the principle of PD could be relevant here in reverse, meaning that if I request access to your location and my request is denied, that you do not learn this, as the requester could find it embarrassing or similar. In this case an SMC-based lookup in the access control table becomes relevant.

Access control based on places, it is a feature that can easily be answered without SMC if the LBS works in peer-2-peer fashion. However, using a centralized server which you tell your location when you are at one of the predefined places could make for a more efficient solution. However, this would make it harder to enforce PD in a granular fashion. In this another user could query your location from the server who would then perform SMC with you.

Location Entropy (LE): In [5] LE is described as an important design principle for LBS's. The idea is that, the more unique visitors are sighted at a given location, the more likely people are to be willing to reveal that they are at this location. Computing the LE of a location requires a centralized server that register sightings including some kind of unique identifier. In [5] they use a study with 500+ users, but numbers on popular LBS's would be much bigger. Consequently, it is not practical to use SMC with participation of all the clients to put the central server out of the equation. Improved privacy could be achieved by using the approach of [6], where a single server is replaced by a network of servers. Of course, this requires that one can find servers that are likely not to collude.

In general, SMC can be used to replace a trusted centralized server, used to solve any problem, not least location based. Of course, as when using a trusted server, it will rely on users not making the LBS work on fake location data. A problem which is out of scope for this paper.

In terms of the more low-level cryptographic security requirements, it is important to distinguish between (at least) passive and active security. They have the following informal characteristics:

Passive security guarantees a protocol does not leak information to an adversary who might try to learn another party's information, given the messages seen in the protocol. However, the adversary is assumed not to deviate from the protocol.

Active security allows for the adversary to behave arbitrarily during the protocol. In order for a protocol to be active secure, this behaviour must not make the protocol leak information or influence the result for a honest user.

A more formal definition of passive and active security can be found in [7] chapters 7.2.2 and 7.2.3. In the case of a proximity testing protocol, passive security is achieved if we can guarantee that if both parties follow the protocol, they will only learn if they are in proximity of each other and nothing else.

Without active security a number of attacks are available, including, but not limited to:

1 Most protocols start by having both parties sending some input to the other party. At this step a malicious party could wait until it receives the honest party's input and use this input to compute its own. This can in some cases allow the malicious party to choose the outcome of the protocol, e.g. if the malicious party simply copied the honest party's input, it would guarantee a proximity test would always result in a match.

2 At some point both parties will have to perform some operations on the inputted encryptions. At this point a malicious party can send some specially crafted message. This might change the result, i.e. making the honest party believe they are in proximity, even if they are not.

3 Finally both parties have to participate to obtain the result, at this point a malicious user could send some garbage message, preventing the honest party from learning the proper result, without affecting his own result.

As we, in the general case, consider both the centralized server as well as peers/friends we communicate with to be potentially dishonest, it should be clear that active security is preferable over passive security for location based services.

III. Proximity Testing

In this section we analyse different ways to compute the proximity of Alice and Bob in terms of performance and accuracy.

The obvious solution would be to calculate the distance between their positions and decide if the distance is lower than some threshold. This is fairly simple and also gives a very accurate answer. Given the positions $(a_x, a_y), (b_x, b_y)$, the computation can be done using either trigonometric functions (the Haversine formula) or by disregarding the Earths curvature (thereby loosing accuracy) and simply calculating the difference ($\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$).

In principle, it is possible to perform either of the two algorithms privately using SMC. However, computing trigonometric operations, square roots, or comparisons using SMC is prohibitively expensive [6]. Thus, another and less ressource intensive solution is needed.

The position of Alice along with a given range defines a circle, and the problem is to test if Bob is inside or outside the circle. Another solution is to approximate the area of the circle with cells of a grid. A position is then mapped to a cell, having a unique identifier, in the grid. In this way, we can decide whether Alice is close to Bob by comparing the cell Bob is in, to the cells approximating the area around Alice. Using this approach, proximity testing can be reduced to set inclusion as noted by others [8]–[10]. Set inclusion is more tractable in terms of performance for techniques like SMC. In our approach we approximate the circle with 9 (roughly) square cells and do set inclusion by comparing the unique identifier of the cell Bob is in to the unique identifiers of the nine cells around Alice.

The tradeoff is that grid based approaches are not as accurate as directly computing the distance. I.e. if Alice is near to Bob but in a different cell, then they may be deemed far apart, resulting in a false negative. Similarly, if Alice and Bob are in the opposite corners of the same (large) cell, and thus far apart, they are still deemed close since they are in the same cell, resulting in a false positive, as shown in Figure 1a.

One way to improve the accuracy is to use more and smaller cells to approximate the area of the circle, as seen in Figure 1b. This approach reduces the amount of false negatives/positives. However, it is not perfect. Assuming that we use nine square 1x1 km cells to approximate the circle, then it is guaranteed that a person less than 1 km away is always determined to be nearby. However, a person being up to $\sqrt{8}$ km. away, may sometimes be determined to be nearby and other times not, all depending on the precise positions in the cells.

The set of cells can be constructed in a number of ways, depending on what is meaningful in the given application. In some cases it might be useful to distinguish between floors in a building or the border between countries. If only distance

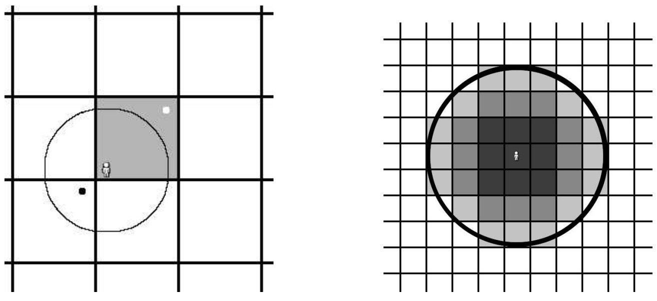
(a) The problem with cells (b) Approximating a circle

Figure 1. The circle in 1a shows the range in which Bob wishes to share his position, while the gray area shows the area covered by his current cell. Depending on Alice's exact position, there might be both false positives (the white dot) and false negatives (the black dot). 1b shows how increasing the number of cells can be used to both approximate a circle.

matters, the plane can be split into cells in a number of ways, using squares, hexagons [8], as well as other shapes, or even layers of grids [10]. While all grid based solutions are inaccurate, certain shapes can approximate a circle using less cells than others.

Since our protocol is *oblivious to the shape of the cell*, we found that the simplests way to split the Earth into cell, is by using the standard latitude and longitude system. The unique id of a cell can, given a persons exact location as latitude and longitude, be computed as the concatenation of the coordinates rounded to the nearest tenth of a degree minute. A cell spans 2 longitude degrees for each latitude degree making the cells roughly quadratic in the test area. This creates a grid of (nearly) square cells, roughly 130x100 m in size. Even smaller cells can be obtained by rounding to the nearest degree minute, second or fraction of a second.

This approach makes the cell sizes vary depending on the latitude, i.e. a cell will be 3 - 4 times larger in Kenya than in Siberia. This is because the absolute length of one degree of longitude difference becomes shorter the further you get from the Equator, whereas one degree of latitude difference always has roughly the same length. This implies that, if Alice and Bob are x m apart and standing on the equator, they are considered nearby. However, if they move straight north at the same speed, they will eventually be considered far apart, eventhough the distance between them remains x m. This is not a problem since the application is supposed to be used around the same latitude.

The size of cells can be made constant by using the protocol to test if the position is close to Equator, and then resize the cells accordingly, e.g. 1 degree latitude to 1 degree longitude near the Equator and 1 to 3 in Siberia.

IV. An Actively Secure 2-Party Protocol for Equality Testing

The proximity of friends is computed by computing the pairwise proximity of the party and each of his friends. The core of the protocol only involves two parties in this way.

The basic idea behind of the core protocol is to first decide if two parties are nearby. This is the case if they are in the equal cells. Equality of cells is done by computing the difference and deciding if it is zero. If two parties are close, they can exchange their exact coordinates using a secure channel, otherwise the protocol stops.

We present an actively secure protocol based on the homomorphic ElGamal cryptosystem [11] combined with a number of zero-knowledge proofs.

To ensure that all parties must participate in the decryption of a ciphertext, a shared private key with a common public key is used.

We setup the cryptosystem as follows:

- *Setup:* We choose G to be a cyclic group of prime order q, with generator g, where the Decisional Diffie-Hellman (DDH) [12] problem is assumed to be hard (and thereby ensuring the security of the system).
- *Private key:* Each party p_i chooses some random $k_i \in \mathbb{Z}_q$ as private key share. Where $\mathbb{Z}_q$ is the set of natural numbers from and including zero and up to but not including q
- *Public key:* Given their private key share k_i, both parties compute and publish $h_i = g^{k_i} \in G$. Each party then computes the common public key as $h = \prod h_i = g^{\sum k_i} \in G$

To *encrypt* a cell id $z_i \in \mathbb{Z}_q$, it is first transformed into $m_i = g^{z_i}$, before the ciphertext $c \in G \times G$, is computed as $c = (g^r, m_i h^r) = (g^r, g^{z_i} h^r)$ where $r \in \mathbb{Z}_q$ is chosen at random. The transformation changes the ElGamal cryptosystem from being multiplicatively homomorphic (i.e. if you multiply two ciphertexts, you get a new encryption of the product of the plaintexts) to being additively homomorphic (i.e. if you multiply two ciphertexts, you get a new encryption of the sum of the zones) with regard to the encoded zones.

To *decrypt* a given ciphertext $c = (\alpha, \beta)$, each party computes and sends a decryption share $d_i = \alpha^{-k_i}$, using their private key share k_i. Given all the decryption shares, the result is obtained as $m_i = g^{z_i} = \beta \prod d_i = \beta \alpha^{-\sum k_i}$. Note that the decryption scheme produces $m_i = g^{z_i}$. Obtaining z_i given g^{z_i} is assumed to be hard for the group G, however we are only interested in the case where $z_i = 0$ i.e. $g^{z_i} = 1$.

The equality testing protocol proceeds as follows for two parties p_1 and p_2, each with cell ids $z_1, z_2 \in \mathbb{Z}_p$:

1) Party p_i create and send encryptions $c_i = (g^{r_i}, g^{z_i} h^{r_i})$
2) Both parties computes the difference locally:

$$\begin{aligned} c &= c_1 c_2^{-1} \\ &= \left(g^{r_1} g^{-r_2}, g^{z_1} h^{r_1} g^{-z_2} h^{-r_2}\right) \\ &= \left(g^{r_1 - r_2}, g^{z_1 - z_2} h^{r_1 - r_2}\right) \end{aligned}$$

   Given $c = (\alpha, \beta)$, party p_i compute and send $c_i = (\alpha^{s_i}, \beta^{s_i})$ for some random $s_i \in \mathbb{Z}_q$
3) After receiving $c_j = (\alpha^{s_j}, \beta^{s_j})$, party p_i compute $c' = (\alpha^{s_j s_i}, \beta^{s_j s_i})$ using the previously generated s_i where $j \neq i$. The ciphertext c' is then jointly decrypted, yielding the result $g^{(z_1 - z_2) s_1 s_2}$. This will either be 1 if z_1 is equal to z_2, i.e. ($g^{0 s_1 s_2} = 1$), or g raised to some unknown power

Note that raising the encryptions to s_i in steps 2 and 3 are necessary to ensure privacy. At step 2, c will be an encryption of $g^{z_\triangle}$, where $z_\triangle$ is the difference in cell ids. The difference can be decrypted at this point, which would leak information. Although finding $z_\triangle$ given $g^{z_\triangle}$ is considered a hard problem in the group G, the possible values for $z_\triangle$ are limited by the number of possible cells and performing an exhaustive search might therefore be feasible[2]. By following step 2 and 3, the parties both get to mask any nonzero difference using some random value of their choosing. In order for a malicious party to obtain information about the difference of the positions, he would have to guess the value used by the honest party. However, the malicious party would not have any way to validate these guesses, perfectly hiding the difference.

The protocol above offers passive security, but as described in Section II active security is required. This is achieved by adding a number of zero knowledge proofs of knowledge (ZKPoK). ZKPoKs allows a person to prove he or she knows some fact, without revealing information about the fact [13].

One example of such a proof, is Schnorr's protocol [14]. Given a prover and verifier, both with knowledge of some values $g, h = g^x$, Schnorr's protocol allows the prover to prove it knows x, without revealing x to the verifier. This is done by having the prover commit to some value r by sending $a = g^r$ to the verifier. The verifier then sends some random challenge e back. The prover finally computes and sends the reply $z = r + ex$ to the verifier, who verifies $g^z = ah^e$. The trick is, that the prover can only compute a valid reply to e, iff it knows both x and r. This way, the verifier learns nothing about x, but since the prover could produce a valid reply, the prover must know x.

In addition to the encrypted values sent in step 1, the parties also performs a ZKPoK, proving they know the encrypted value and the randomness used to produce it. This prevents attack 1 from Section II, because a malicious party cannot just copy the value received.

At step 2, the parties prove that they know s_i. The proof use the result from the previous step as common input to both prover and verifier. By proving knowledge of s_i, they also prove they haven't replaced the encrypted difference with some encryption of their choosing. This makes the parties perform the computations like they are supposed to preventing attacks of type 2 mentioned in Section II.

Finally in the decryption phase, the parties prove that the decryption share is produced using their own key. As in

[2]Earths surface is roughly 500 mio. square km. By making a few assumptions on the other party's location, this can be drastically lowered.

step 2, this also verifies that the decryption share is produced from the result of the previous round. This prevents the third type of attack, where a malicious party learns the correct result and makes the honest party learn an erroneous result.

The proofs are fairly simple on their own, but become quite lengthy to describe when used as subcomponents and are therefore left out. The used proofs are slightly modified versions of Schnorr's protocol, Okamoto's protocol [15] and Chaum-Pedersens [16] protocol for proving equality of discrete logarithms.

The program is linear and the execution time mainly depends on the size of group. The protocol communicates via 6 message sent in 3 rounds.

V. Evaluation

To test what level of accuracy and security is feasible for handheld devices, a prototype implementation was made for Android devices. The prototype can use indoor positioning, using wifi fingerprint of nearby hotspots, or outdoor positions using the built-in GPS. Indoor positioning uses an adjacency matrix to describe the grid layout, while the outdoor positioning computes the current cell as described in Section III.

The protocol was tested on a setup using a HTC Desire and a Google Nexus phone. Both having a 1 GHz processor and communicating on a WiFi connection. Nor the protocol or the security attributes assumes any properties of the network, thus it could as well have been i.e. 3G, Edge or Bluetooth. The transport protocol will of cause impact the execution time.

Since the level of accuracy can be increased by using a set of cells closely approximating a circle (as in Figure 1b). We have benchmarked three different configurations using 9, 25 and 49 comparisons, in order to investigate what accuracy is practical. The results were 2720 ms, 6061 ms and 11656 ms respectively. The measurements have been done with a fixed key size of 160 bits, using a group size of 1248 bits. And averaged over 10 executions of the protocol. The parties spend around 1 sec to generate a session key before the comparisons, this can be eliminated by having all friends precompute their shared/common keys, and is thus not part of the timings.

As we would expect increasing accuracy comes with a cost. While 12 seconds for evaluating proximity might be acceptable in a situation with only a single friend, a user might often want to run the protocol with multiple friends. In order for the application to be able to provide fresh positions, the protocol must be run at fairly short intervals. That means that the level of accuracy that 49 cells would give, is most often infeasible with current mobile hardware. However, using a configuration using few cells is very much feasible. We found that using 9 comparisons and initiating the protocol with 1 minute intervals seems appropriate for 10 to 20 friends. This results in an average CPU workload of 30 to 60%. Although the timings indicate there would be enough CPU time for having more friends and/or higher accuracy/frequency, keep in mind that handheld devices are limited by battery, so CPU power consumption does become an issue. The level of accuracy offered by nine cells is definitely sufficient to be of practical use in situations like meetups at concerts or sport events.

	Our (grid)	Our (hexagons)	NTLHB (hexagons)
Passive	1.4 s/friend	0.46 s/friend	0.46 s/friend
Active	2.7 s/friend	0.9 s/friend	-

Table I
Comparison of our protocol to the NTHLB [8] protocol.

Using cells of another shape, might require less comparisons for an equal (or better) level of accuracy. E.g. the protocols described in [8] uses hexagons and only needs 3 comparisons for their desired level of accuracy.

We made a passively secure version of our protocol by removing the zero-knowledge proofs in order to better compare our results with the result in [8]. The comparison is shown in Table I with execution times for the active and passive versions of our protocol for a square (9 comparisons) and hexagonal (3 comparisons) grid layout. The passive version is *only* a factor 2 faster than the active version. And just as fast as the one in [8]. We would expect a speedup considering the more powerful CPUs used in our setup, but this can be explained by our benchmark including network latency, which is excluded in [8].

The confidentiality of the encrypted positions can be extended from a few years to a couple of decades by using an encryption key of 192 bits (the group will then be 1776 bits). Resulting in a 25% increase in the execution time (3850 ms) for nine comparisons.

The implementation supports gracefull degradation of service. If a user has too many friends compared to the computing power of the mobile device. Then, the application will still attempt to execute the protocol. However, other users may perceive the device as unresponsive, thereby creating false negatives, eventhough they are nearby.

VI. Related Work

We improve on the previous work [8] by adding active security. Our protocol only incurs a factor 2 slowdown compared to [8] when comparing the raw cryptographic primitives under the same conditions. We also map locations on the Earth into a grid. The choice of grid influences the accuracy and number of comparisons needed. We could also have chosen to use layers of grids similar to [10], and have the parties negotiate which layer to use for comparison, and so enabling individual distance preferences. Our solution does not reveal to the LBS if two users are in proximity as opposed to both protocols in [9] where the LBS can compare the values send to the LBS. Also the LBS can collude with Bob to find Alices position or tell Alice that she is far apart from Bob while they are actually nearby. These attacks are not possible in our solution.

VII. Conclusion and Future Work

We have designed and implemented an actively secure protocol for proximity testing. We have shown that it is feasible to execute the protocol on contemporary mobile devices, if the set of friends is not more than 10 or 20. Which is typical in the scenario we target. Our protocol is only a factor of two slower than an existing passively secure solution [8].

Processors in mobile devices seems to become more and more powerful and the numbers of cores increases. This will have a beneficial impact on the number of friends the protocol can handle in reasonable time, because the protocol is CPU bound and parallelises well. The main computational bottleneck is modular integer exponentiation. Thus, a more efficient implementation will give the highest increase in the practical usefulness of the protocol.

Acknowledgment

The authors would like to thank Gert Mikkelsen and Thomas Jakobsen for fruitful discussions. And thanks to the anonymous reviewers for suggestions on improving the paper.

References

[1] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *Foundations of Computer Science*. IEEE, 1986, pp. 162–167.

[2] G. Hsieh, K. P. Tang, W. Y. Low, and J. I. Hong, "Field deployment of *imbuddy* : A study of privacy control and feedback mechanisms for contextual im," in *Ubicomp*, ser. Lecture Notes in Computer Science, J. Krumm, G. D. Abowd, A. Seneviratne, and T. Strang, Eds., vol. 4717. Springer, 2007, pp. 91–108.

[3] N. M. Sadeh, J. I. Hong, L. F. Cranor, I. Fette, P. G. Kelley, M. K. Prabaker, and J. Rao, "Understanding and capturing people's privacy policies in a mobile social networking application," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 401–412, 2009.

[4] G. Iachello, I. E. Smith, S. Consolvo, G. D. Abowd, J. Hughes, J. Howard, F. Potter, J. Scott, T. Sohn, J. Hightower, and A. LaMarca, "Control, deception, and communication: Evaluating the deployment of a location-enhanced messaging service," in *Ubicomp*, ser. Lecture Notes in Computer Science, M. Beigl, S. S. Intille, J. Rekimoto, and H. Tokuda, Eds., vol. 3660. Springer, 2005, pp. 213–231.

[5] E. Toch, J. Cranshaw, P. H. Drielsma, J. Y. Tsai, P. G. Kelley, J. Springfield, L. F. Cranor, J. I. Hong, and N. M. Sadeh, "Empirical models of privacy in location sharing," in *UbiComp*, ser. ACM International Conference Proceeding Series, J. E. Bardram, M. Langheinrich, K. N. Truong, and P. Nixon, Eds. ACM, 2010, pp. 129–138.

[6] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. 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 *Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Dingledine and P. Golle, Eds., vol. 5628. Springer, 2009, pp. 325–343.

[7] O. Goldreich, *Foundations of cryptography. II.* Cambridge: Cambridge University Press, 2004, basic Applications.

[8] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location privacy via private proximity testing," in *NDSS*. The Internet Society, 2011.

[9] 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.

[10] L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu, "Private and flexible proximity detection in mobile social networks," in *Mobile Data Management*, T. Hara, C. S. Jensen, V. Kumar, S. Madria, and D. Zeinalipour-Yazti, Eds. IEEE Computer Society, 2010, pp. 75–84.

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

[12] D. Boneh, "The decision diffie-hellman problem," in *ANTS*, ser. Lecture Notes in Computer Science, J. Buhler, Ed., vol. 1423. Springer, 1998, pp. 48–63.

[13] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *J. Cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[14] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proceedings on Advances in cryptology*, ser. CRYPTO '89. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 239–252. [Online]. Available: http://dl.acm.org/citation.cfm?id=118209.118231

[15] T. Okamoto, "Provably secure and practical identification schemes and corresponding signature schemes," in *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '92. London, UK: Springer-Verlag, 1993, pp. 31–53. [Online]. Available: http://dl.acm.org/citation.cfm?id=646757.705529

[16] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '92. London, UK: Springer-Verlag, 1993, pp. 89–105. [Online]. Available: http://dl.acm.org/citation.cfm?id=646757.705670