# Louis, Lester and Pierre: Three Protocols for Location Privacy

Ge Zhong, Ian Goldberg, and Urs Hengartner

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 3G1
{gzhong,iang,uhengart}@cs.uwaterloo.ca

**Abstract.** Location privacy is of utmost concern for location-based services. It is the property that a person's location is revealed to other entities, such as a service provider or the person's friends, only if this release is strictly necessary and authorized by the person. We study how to achieve location privacy for a service that alerts people of nearby friends. Here, location privacy guarantees that users of the service can learn a friend's location if and only if the friend is actually nearby. We introduce three protocols—Louis, Lester and Pierre—that provide location privacy for such a service. The key advantage of our protocols is that they are distributed and do not require a separate service provider that is aware of people's locations. The evaluation of our sample implementation demonstrates that the protocols are sufficiently fast to be practical.

## 1 Introduction

The ubiquity of cellphones has led to the introduction of location-based services, which let cellphone users benefit from services that are tailored to their current location. For example, individuals can learn about interesting nearby places or get directions to a target location. Location privacy is of utmost concern for such location-based services, since knowing a person's location can leak information about her activities or her interests. Therefore, a person's location should be revealed to other entities only if this release is strictly necessary and authorized.

The potential of location-based services, together with rising interest in social-networking applications, has led to the introduction of buddy-tracking applications. For example, Boost Mobile, a US cellphone service targeted at young people, offers the Loopt Service [14], which alerts users of nearby friends. The drawback of the Loopt Service is that it is bound to a particular cellphone network and wireless technology. MIT's iFIND project [15] works around this problem by introducing a distributed buddy-tracking application, where a person's WiFi device determines its location and shares this information with the person's friends. While it is possible to exploit this approach for alerting people of nearby friends, its disadvantage is that the friends always learn each other's location, regardless whether they are actually nearby; that is, the approach may

reveal more information than desired. What we really want is a distributed buddy-tracking application where users (and their devices) can learn information about their friends' locations if and only if their friends are actually nearby. In the rest of this paper, we call this problem the *nearby-friend problem*.

We present three protocols—Louis, Lester and Pierre[1]—for solving the nearby-friend problem. The Louis protocol requires a semi-trusted third party that does not learn any location information. The Lester protocol does not need a third party, but has the drawback that a user might be able to learn a friend's location even if the friend is in an area that is no longer considered nearby by the friend. However, this can happen only if the user is willing to invest additional work. The Pierre protocol does not have this disadvantage at the cost of not being able to tell the user the precise distance to a nearby friend.

Our protocols can run on wireless devices with limited communication and computation capabilities. The Louis protocol requires four communication steps, whereas the Lester and Pierre protocols require only two steps. Furthermore, the evaluation of our sample implementation shows that the cost of running our protocols is comparable to the cost of setting up a TLS [7] connection.

The rest of this paper is organized as follows. In section 2, we discuss previous approaches to solve the nearby-friend problem. Our protocols exploit homomorphic encryption, which we review in section 3. We present the Louis, Lester and Pierre protocols in sections 4, 5, and 6, respectively, and compare their features in section 7.

## 2 Related Work

Location cloaking has been a popular approach for providing location privacy [5, 9, 10, 16]. Here, an individual's device or a third party cloaks the individual's location before giving it to the provider of a location-based service. Cheng et al. [5] study location cloaking for a service that alerts people of nearby friends. For each individual, the service provider knows only that the individual is within a particular region, but not where exactly. The authors develop a metric for describing the quality of an answer received from the service. This metric allows an individual to trade off privacy for better answer quality. A drawback of this approach is that the service provider learns some location information. Our protocols do not require such a third party. (In the Louis protocol, the third party does not learn any location information.) Furthermore, if a friend is nearby, our protocols will always return a positive answer and there is no doubt about the quality of the answer.

The nearby-friend problem is an instance of a secure multiparty computation problem, where multiple parties jointly compute the output of a function without learning each other's inputs. We next examine two previous approaches based on secure multiparty computation that are applicable to solving the nearby-friend problem.

[1] Our protocols are named after three former residents of 24 Sussex Drive, Ottawa.

Køien and Oleshchuk [12] present a secure two-party protocol for the point-inclusion problem. The protocol allows Alice to learn whether a point chosen by Bob is in a polygon determined by Alice, without Bob revealing the point to Alice and without Alice revealing the polygon to Bob. We could exploit this protocol for letting Alice know whether Bob is nearby. Namely, Alice determines the circle around her current location that corresponds to the area that she considers nearby and approximates the circle with a polygon; Bob picks the point that corresponds to his current location. However, Køien and Oleshchuk's protocol has a flaw: Alice can learn Bob's location by choosing a degenerate polygon. For example, if there are only two different edges in the polygon and all the other edges are identical to one of them, Alice will usually be able to solve a system of linear equations to determine Bob's location. Bob cannot detect degenerate polygons, assuming the underlying encryption scheme is semantically secure, so this protocol is not adequate for solving the nearby-friend problem.

Atallah and Du [1] also study the point-inclusion problem. Their protocol lets both Alice and Bob learn whether Bob's point is in Alice's polygon. The protocol is based on solving the secure two-party scalar product problem and the secure two-party vector dominance problem [1]. With the help of a semi-trusted third party, the first problem can be solved in three communication steps [8]. The solution of the second problem is based on solving Yao's millionaire problem [22]. The most efficient constant-round protocol for solving this problem requires six communication steps [3]. With a semi-trusted third party, the problem can be solved in three communication steps [4]. Our Louis protocol, which needs a semi-trusted third party, lets Alice know in four communication steps whether Bob is nearby and requires one additional step to inform Bob of this result. The Lester and Pierre protocols each require two communication steps to let Alice learn whether Bob is nearby. To let Bob know whether Alice is nearby, these protocols also require one additional step. In summary, to achieve the same result as Atallah and Du's protocol, our protocols require fewer communication steps and the Lester and Pierre protocols do not need a third party at all.

## 3 Homomorphic Encryption

Our protocols use the techniques of public-key cryptography, but we require the cryptosystems used to have a special algebraic property: that they are *additive homomorphic*. An additive homomorphic cryptosystem is one in which, given $\mathcal{E}(m_1)$ and $\mathcal{E}(m_2)$, one can efficiently compute $\mathcal{E}(m_1 + m_2)$. Our protocols use two of these systems, which we review here.

### 3.1 Paillier

The first of these systems is the Paillier cryptosystem [18]. Like the RSA cryptosystem, a user Alice selects random primes $p$ and $q$ and constructs $n = pq$; plaintext messages are elements of $\mathbb{Z}_n$. Unlike RSA, however, ciphertexts are elements of $\mathbb{Z}_{n^2}$. Alice picks a random $g \in \mathbb{Z}_{n^2}^*$ and verifies that $\mu =$

$(L(g^\lambda \bmod n^2))^{-1} \bmod n$ exists, where $\lambda = \text{lcm}(p-1, q-1)$ and $L(x) = (x-1)/n$. Alice's public key is then $(n, g)$ and her private key is $(\lambda, \mu)$.

To encrypt a message $m$, another user Bob picks a random $r \in \mathbb{Z}_n^*$ and computes the ciphertext $c = \mathcal{E}(m) = g^m \cdot r^n \bmod n^2$. To decrypt this message, Alice computes $\mathcal{D}(c) = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$, which always equals $m$.

Given $\mathcal{E}(m_1) = g^{m_1} \cdot r_1^n \bmod n^2$ and $\mathcal{E}(m_2) = g^{m_2} \cdot r_2^n \bmod n^2$, Bob can easily compute $\mathcal{E}(m_1 + m_2) = \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) \bmod n^2 = g^{m_1+m_2} \cdot (r_1 r_2)^n \bmod n^2$.

Note that if Bob does not trust Alice enough to generate her Paillier modulus correctly, he can insist she prove its validity (that is, that it is the product of exactly two nearly equal primes) [13].

### 3.2 CGS97

Cramer, Genarro and Schoenmakers [6] present the CGS97 scheme. This is a variant on El Gamal, where we have (public) large primes $p$ and $q$ such that $q|p-1$. Plaintexts are elements of $\mathbb{Z}_q$ and ciphertexts are elements of $\mathbb{Z}_p \times \mathbb{Z}_p$. Alice's private key is a random element $a \in \mathbb{Z}_q$ and her public key is $A = g^a \bmod p$.

To encrypt a message $m$, Bob picks a random $r \in \mathbb{Z}_q$ and computes $(c_1, c_2) = \mathcal{E}(m) = (g^r \bmod p, A^{r+m} \bmod p)$. To decrypt this message, Alice finds $A^m = c_2 \cdot c_1^{-a} \bmod p$ and computes $m$ as the discrete log of that value with the base of $A$, mod $p$. Note that this can only be done if $M$, the number of possible plaintext messages, is small. In that event, the Pollard lambda, or "kangaroo", method [19] can find $m$ in time $O(\sqrt{M})$.

Given $\mathcal{E}(m_1) = (g^{r_1} \bmod p, A^{r_1+m_1} \bmod p)$ and $\mathcal{E}(m_2) = (g^{r_2} \bmod p, A^{r_2+m_2} \bmod p)$, Bob can easily compute $\mathcal{E}(m_1+m_2) = (g^{r_1+r_2} \bmod p, A^{r_1+r_2+m_1+m_2} \bmod p)$ by pointwise multiplication mod $p$.

## 4 The Louis Protocol

There are three participants in the Louis protocol: Alice, Bob and Trent. Alice and Bob are friends and Alice wants to know whether Bob is nearby. Alice considers Bob nearby if he is within a circle of some radius $r$ centered around Alice. Alice informs Bob of $r$ and Bob can refuse to participate in the protocol if he considers it to be too large. Trent acts as a third party and helps Alice and Bob decide whether they are nearby. Unlike other protocols for implementing location-based services that exploit third parties [5, 9, 10, 14], the Louis protocol does not allow Trent to learn any location information about either Alice or Bob.

Our protocol consists of two phases. In the first phase, Alice and Bob jointly solve the nearby-friend problem and Alice learns whether Bob is nearby. If this is the case, Alice and Bob inform each other of their locations in the (optional) second phase of the protocol. Alice and Bob cannot learn each other's locations if they are not nearby.

Alice and Bob can misbehave and input fake locations into the protocol. However, the detection of misbehaviour by one of them will likely affect their

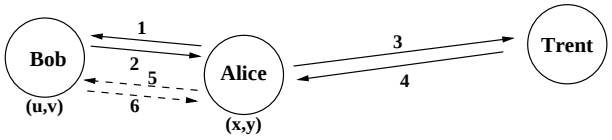


**Fig. 1.** System model of the Louis protocol. The dashed arrows indicate the optional second phase.

friendship, so they are less likely to misbehave. We discuss the detection of misbehaviour by Alice or Bob, and of cheating by the third party Trent in section 4.3.

### 4.1 Protocol Description

We assume that a location can be mapped to two-dimensional coordinates and that the mapping is known to Alice and Bob. Let Alice's location be $(x, y)$ and Bob's be $(u, v)$. By the definition above, they are nearby if $\sqrt{(x-u)^2 + (y-v)^2} < r$. Equivalently, we can check the sign of $d = (x-u)^2 + (y-v)^2 - r^2$. In particular, Bob is near Alice if $d < 0$.

Figure 1 presents the two communication channels used in our system model. The first is between Alice and Bob, and the second is between Alice and Trent. Alice also acts as a relay of the communication between Bob and Trent. The benefit of this approach is to hide Bob's identity from Trent, thus improving privacy. We assume that the two secure communication channels are set up before our protocol begins.

The protocol consists of two phases. The first phase lets Alice determine whether Bob is nearby. If this is the case, the (optional) second phase lets Alice and Bob learn each other's locations. In our protocol, $\mathcal{E}_A(\cdot)$ is the Paillier additive homomorphic encryption function using Alice's public key, $\mathcal{E}_T(\cdot)$ is a (non-homomorphic) public-key encryption function using Trent's public key, $\mathcal{H}(\cdot)$ is a cryptographic hash function, $sig_A(m)$ is Alice's signature on message $m$, and similarly with $sig_T(m)$.

1. **First phase:** Alice determines her location $(x, y)$ and her desired radius $r$, and picks a random salt $s_A$.
   Alice→Bob: $\mathcal{E}_A(x^2 + y^2)$, $\mathcal{E}_A(2x)$, $\mathcal{E}_A(2y)$, $r$, $\mathcal{H}(x \parallel y \parallel s_A)$
2. Bob checks the value of $r$. If he thinks $r$ is too large, he aborts the protocol. Otherwise, he determines his location $(u, v)$, picks a random value $k$ and computes
   $$\mathcal{E}_A(d+k) = \frac{\mathcal{E}_A(x^2+y^2) \cdot \mathcal{E}_A(u^2+v^2) \cdot \mathcal{E}_A(k)}{(\mathcal{E}_A(2x))^u \cdot (\mathcal{E}_A(2y))^v \cdot \mathcal{E}_A(r^2)},$$
   Bob also chooses a random salt $s_B$.
   Bob→Alice: $\mathcal{E}_A(d+k)$, $\mathcal{E}_T(k)$, $\mathcal{H}(u \parallel v \parallel s_B)$, $\mathcal{H}(k)$.
3. Alice decrypts $\mathcal{E}_A(d+k)$.
   Alice→Trent: $d+k$, $\mathcal{E}_T(k)$, $sig_A(d+k)$, $sig_A(\mathcal{E}_T(k))$

| | Alice | Bob | Trent |
|---|---|---|---|
| TLS connection time | $516 \pm 2$ ms | $255 \pm 4$ ms | $256 \pm 2$ ms |
| Computation time | $635 \pm 4$ ms | $175 \pm 4$ ms | $41 \pm 0.6$ ms |

**Table 1.** Runtime of the Louis protocol.

4. Trent decrypts $\mathcal{E}_T(k)$ and verifies Alice's signatures. Next, he computes $d$. If $d < 0$, Trent sets $answer = \text{'YES'}$ else $answer = \text{'NO'}$.
   Trent→Alice: $answer$, $sig_T(answer \parallel sig_A(d+k) \parallel sig_A(\mathcal{E}_T(k)))$.
5. Alice verifies Trent's signature. Next, if $answer == \text{'YES'}$, she knows that Bob is nearby. Alice terminates the protocol if Bob is not nearby or if only the first phase of the protocol is run. Otherwise:
   **Second phase:** Alice reveals her location to Bob:
   Alice→Bob: $answer$, $d+k$, $sig_A(d+k)$, $sig_A(\mathcal{E}_T(k))$, $sig_T(answer \parallel sig_A(d+k) \parallel sig_A(\mathcal{E}_T(k)))$, $x$, $y$, $s_A$.
6. Bob verifies all signatures. He then computes $\mathcal{H}(x \parallel y \parallel s_A)$ and compares the hash value with the one provided by Alice in step 1. He also uses $(x, y)$ to compute $d + k$ and compares it to the value received. If the values do not match, Bob aborts the protocol. Otherwise Bob reveals his location to Alice:
   Bob→Alice: $u$, $v$, $s_B$, $k$.
7. Alice computes $\mathcal{H}(u \parallel v \parallel s_B)$ and $\mathcal{H}(k)$ and compares the values with the hash values provided by Bob in step 2. Alice also computes $d + k$ based on $(x, y)$, $(u, v)$, and $k$ and verifies whether it equals the decrypted value of $\mathcal{E}_A(d + k)$.

Note that our protocol checks whether $d < 0$. In the Paillier cryptosystem, $d$ will be an element of $\mathbb{Z}_n$, so to check this condition, we ensure that $n$ is sufficiently large, and we say $d < 0$ if $n/2 < d < n$.

### 4.2 Measurements

We implemented our protocols using the OpenSSL [17] and NTL [21] libraries. We chose RSA for the non-homomorphic encryption and signature functions. The key sizes of all the cryptographic functions are 2048 bits. Our hash function is SHA-256, and the cipher stack in TLS is AES256 in CBC mode with ephemeral Diffie-Hellman key exchange. (The ephemeral keys can be used in the Lester and Pierre protocols, below.) We evaluated these protocols on a 3.0 GHz Pentium 4 desktop. We ran the protocol one hundred times and measured TLS connection-setup time and overall computation time for each protocol participant. Table 1 shows our results.

With 2048-bit keys, it takes about a quarter second to set up a TLS connection. Alice initiates two TLS connections, which takes about half a second. Trent's computation time is very small. The major burden is on Alice, who takes about 0.6 s; Bob's computation time is less than one third of Alice's. In short, if a mobile device can set up a TLS connection, it should be able to finish the Louis protocol in comparable time or shorter.

### 4.3 Analysis

The Louis protocol can directly detect scenarios where Alice and Bob reveal other locations than the ones they committed to. We next explain how Alice and Bob can discover other kinds of misbehaviour.

*Alice detects misbehaviour by Bob or Trent.* If Alice detects suspicious behaviour, such as not spotting nearby Bob though she was told that he is nearby, and if only the first phase of the protocol has been run, Alice asks Bob to execute the second phase. If Bob refuses, Alice will suspect that Bob misbehaved. Otherwise, Alice proceeds as follows:

If Alice is told by Trent that Bob is nearby, but then fails to spot Bob at his released, nearby location, Alice will realize Bob's misbehaviour. If the released location is not nearby, Alice asks Bob to reveal the random values that he used in his calculations and repeats the calculations. If the results are not identical to the ones released by Bob, Bob must have misbehaved. Otherwise, there was cheating by Trent.

If Alice is told by Trent that Bob is not nearby, but then spots him in her vicinity, she proceeds in a similar way. Namely, if the location released by Bob is not nearby, Bob must have misbehaved. If it is nearby, Alice repeats Bob's calculations, as explained above, to detect cheating by Trent.

Finally, if step 7 in the protocol fails, Alice also repeats Bob's calculations to discover who misbehaved.

*Bob detects misbehaviour by Alice or Trent.* If the second phase of the protocol is not run, Bob does not learn any location information about Alice, which makes it impossible for him to detect misbehaviour. However, Bob can refuse to answer multiple queries from Alice if they arrive within a very short time. These queries could be part of a probing attack, where Alice knows a set of likely locations for Bob and uses each of them for invoking the protocol.

If the second phase of the protocol is run and Bob detects suspicious behaviour, Bob uses mechanisms similar to Alice's to discover misbehaviour.

*Alice or Bob collude with Trent.* Our protocol cannot detect collusion, where Trent tells the value of $d$ to one of the parties. However, Alice and Bob can jointly choose the third party, which reduces the risk of collusion.

## 5 The Lester Protocol

The Louis protocol allows Alice and Bob to learn each other's locations if and only if they are nearby, but it requires the participation of Trent. In our second protocol, Lester, we do away with the need for Trent. However, this comes at some small costs. First, the information disclosure is now only one-way; that is, Alice learns about Bob's location, but not vice versa. Alice and Bob could of course run the protocol a second time, with the roles reversed, to mutually exchange information. (Note that this requires only one extra message, since the resulting two messages from Bob to Alice can be combined.) Second, Alice learns less exact information about Bob; she only learns the distance between them, although this may actually be a benefit, depending on the context.

### 5.1 Protocol Description

This protocol uses the CGS97 cryptosystem of section 3.2. Recall that this cryptosystem has an unusual property: the amount of work Alice must do in order to decrypt a message depends on the number of possible messages. We use this property to our advantage in this protocol.

The Lester protocol is very simple. Let $a$ and $b$ be Alice and Bob's private keys, and $A = g^a$ and $B = g^b$ be their public keys. Note that these keys may be ephemeral; if Alice and Bob are communicating via TLS [7], for example, they can use the key pairs from an ephemeral Diffie-Hellman key exchange. Alice and Bob can each calculate $C = A^b = B^a$. Alice sends Bob $\mathcal{E}_A(x^2 + y^2)$, $\mathcal{E}_A(2x)$, $\mathcal{E}_A(2y)$. Bob picks a *workfactor* $t$ (see below) and a random salt $s$ of length $t$, and sends to Alice $t, \mathcal{E}_A(b \cdot (D \cdot 2^t + s))$, where $D = (x-u)^2 + (y-v)^2$ is the square of the distance between Alice and Bob. Alice receives this message, and can calculate $A^{b \cdot (D \cdot 2^t + s)} = C^{D \cdot 2^t + s}$.

If Alice wants to learn whether Bob is closer than some threshold distance $r$ away, she uses the kangaroo method [19] to determine $D \cdot 2^t + s$ if it is in the range $[0, r^2 \cdot 2^t]$. This can be done in time $O(r \cdot 2^{t/2})$ and space $O(t \log r)$. Other methods to calculate discrete logarithms, such as baby-step-giant-step [20], can solve this problem with the same runtime, but with exponentially larger space requirements. If this step is successful, shifting the result by $t$ bits yields $D$. The effect of Bob including a factor of $b$ in his response to Alice is that Alice's discrete logarithm calculation is to the base of the ephemeral $C$ rather than $A$. This prevents Alice from doing a certain amount of reusable precomputation derived from a predetermined base.

Bob should choose $t$ so that he is comfortable with the amount of work Alice would have to do in order to discover the distance between them. This will likely depend on things Bob knows about his friend Alice, such as the computational capacity of Alice's cellphone.

### 5.2 Measurements

The runtime of the Lester protocol is dominated by Alice's computation of the discrete log of $C^{D \cdot 2^t + s}$ to the base of $C$. In Figure 2, we plot this time against the workfactor value $t$, chosen by Bob. For fixed $r$, we expect this runtime to scale as $2^{t/2}$ and the log plot shows that this is indeed the case. This gives Bob a fair amount of control over the amount of work Alice will need to do to find the distance between them: in our setup, if $t = 20$, Alice needs only about a quarter of a second, and if $t = 40$, Alice needs a few minutes of computation time. If this is not enough, Bob could choose even larger values, and the exponential nature of the runtime means he can make Alice work a very long time with only a small increase in $t$.

We measured Bob's computation time, on the other hand, to be $175 \pm 2$ ms, comparable to that of the Louis protocol, and this value is independent of $t$.

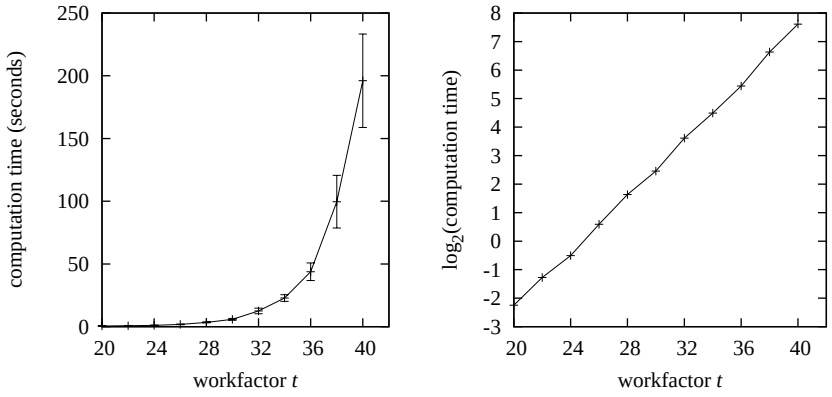


**Fig. 2.** Alice's computation time in the Lester protocol.

### 5.3 Analysis

This protocol has no way to detect if Alice or Bob use incorrect locations as their input. This could allow Alice to confirm a guess of Bob's location simply by entering that guess as her own location and seeing if the protocol successfully finds Bob to be very nearby. Alice could also check specific ranges of large values of $D$. For example, if locations are measured in metres, she could check whether Bob is between 10000 and 11000 m away for about the same cost as checking whether he is between 0 and 4600 m away. Of course, the former ring represents a much more widely spread out geographical area, and knowing only that Bob is in that ring probably gives less useful information to Alice. An exception is when Alice knows a few places that Bob is likely to be: his home, his work, etc.; she can then confirm those guesses with minor difficulty. Note that Bob has a little bit of extra power: not only can he choose a large $t$ if he suspects Alice is probing for his exact location, but he can also effectively refuse to participate in the protocol, without letting Alice know. He does this by returning an *unconditional negative*; that is, an encryption of a random value instead of the correct response. This makes it extremely probable that Alice's discrete log computation will fail. If Bob wants to be extra careful, he should be sure to avoid revealing he has done this to side channels, such as timing differences [11]. Conversely, he could return an *unconditional positive* by returning an encryption of a small number rather than the result of his calculation. If Alice cares, she can prevent the latter by adding a random value $k$ to her $x^2 + y^2$ and dividing Bob's response by $C^{k \cdot 2^t}$. Of course, as in the Louis protocol, Alice is likely to notice if Bob claims to be nearby but is not.

Another downside of this protocol is that Bob only has very coarse control over the threshold distance; he can choose how much work Alice would have to do in order to discover that he was, say, 500 metres away, but with only twice as much work, Alice could discover that Bob was 1000 metres away. A minor modification to the Lester protocol, however, can make Alice's work be quadratic

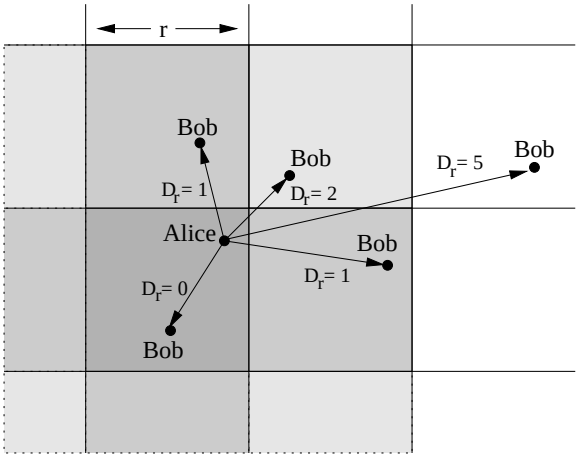


**Fig. 3.** Grid distances in the Pierre protocol. The $x$ and $y$ distances between Alice and Bob are measured in grid cells (integral units of $r$), and $D_r = (\Delta x_r)^2 + (\Delta y_r)^2$. Alice can determine whether Bob is in the dark grey, medium grey, or light grey area, but no more specific information than that.

in the threshold distance instead of linear. Instead of the CGS97 cryptosystem, the Boneh-Goh-Nissim cryptosystem [2] can be used. This protocol has the same properties (additive homomorphic; decryption takes $O(\sqrt{M})$ time) as CGS97, but also allows calculations of encryptions of *quadratic* functions, in addition to linear ones. With this system, Bob could compute $\mathcal{E}_A(D^2 \cdot 2^t + s)$ for a random salt $s$ between 0 and $(2D+1)2^t - 1$, and Alice's work to find the distance to Bob will be $O(r^2 \cdot 2^{t/2})$.

## 6 The Pierre Protocol

Our third protocol, Pierre, solves the above problems with the Lester protocol and gives Bob more confidence in his privacy. On the other hand, if Alice and Bob are nearby, the Pierre protocol will inform Alice of that fact, but will give her much less information about Bob's exact location.

### 6.1 Protocol Description

In this protocol, Alice picks a *resolution distance* $r$, roughly analogous to the threshold distance $r$ in the previous protocols. Alice and Bob then express their coordinates in (integral) units of $r$; that is, if Alice's true position is $(x, y)$, then for the purposes of this protocol, she will use coordinates $(x_r, y_r) = (\lfloor \frac{x}{r} \rfloor, \lfloor \frac{y}{r} \rfloor)$, and similarly for Bob. This has the effect of dividing the plane into a grid, and Alice and Bob's location calculations only depend on the grid cells they are in; see Figure 3.

| | Alice | Bob |
|---|---|---|
| TLS connection time | $256 \pm 3$ ms | $257 \pm 1$ ms |
| Computation time | $384 \pm 4$ ms | $354 \pm 3$ ms |

**Table 2.** Runtime of the Pierre protocol.

This protocol can use either of the homomorphic cryptosystems we have mentioned. It turns out that CGS97 is slightly more efficient, so we will use the notation of that system. As with the Lester protocol, Alice and Bob's public keys can be the ephemeral ones generated during TLS setup.

Alice sends to Bob $r, \mathcal{E}_A(x_r^2 + y_r^2)$, $\mathcal{E}_A(2x_r)$, $\mathcal{E}_A(2y_r)$. Bob picks three random elements $\rho_0, \rho_1, \rho_2$ of $\mathbb{Z}_p^*$ and replies with $\mathcal{E}_A(\rho_0 \cdot D_r), \mathcal{E}_A(\rho_1 \cdot (D_r - 1)), \mathcal{E}_A(\rho_2 \cdot (D_r - 2))$, where $D_r = (x_r - u_r)^2 + (y_r - v_r)^2$ is the square of the distance between Alice and Bob, in integral units of $r$. As in the Lester protocol, if Bob is uncomfortable with Alice's query, either because of her choice of $r$, her frequency of querying, or some other reason, Bob can reply with encryptions of three random values, ensuring Alice will not think he is nearby.

Note that $\rho_0 \cdot D_r = 0$ if Alice and Bob are in the same grid cell and is a random element of $\mathbb{Z}_p^*$ otherwise. Similarly, $\rho_1 \cdot (D_r - 1) = 0$ if Alice and Bob are in adjacent grid cells and random otherwise, and $\rho_2 \cdot (D_r - 2) = 0$ if Alice and Bob are in diagonally touching grid cells and random otherwise.

In CGS97, it is easy for Alice to check whether a received ciphertext $(c_1, c_2)$ is an encryption of 0: this is the case exactly when $c_2 = {c_1}^a \bmod p$, where $a$ is Alice's private key. Therefore, with this protocol, Alice can tell when Bob is in the same, adjacent, or diagonally touching grid cell (and learns which is the case), but she learns no more specific information than that.

### 6.2 Measurements

We measured the computation time of the Pierre protocol using 2048-bit keys for both TLS and CGS97; the results are shown in Table 2. For comparison, we also show the time to set up an TLS connection between Alice and Bob. The computation times shown are for the worst-case situation; that is, Alice and Bob are not nearby.

We can see that the computational cost of the Pierre protocol is only slightly more expensive than setting up TLS; this suggests that the protocol would be reasonable to run on mobile devices.

### 6.3 Analysis

As with the other protocols, we cannot prevent Alice from using an incorrect location in order to try to confirm a guess of Bob's location. However, in the Lester protocol, as mentioned above, Alice can try to verify a number of guesses with a single query to Bob. This is not the case in the Pierre protocol; each

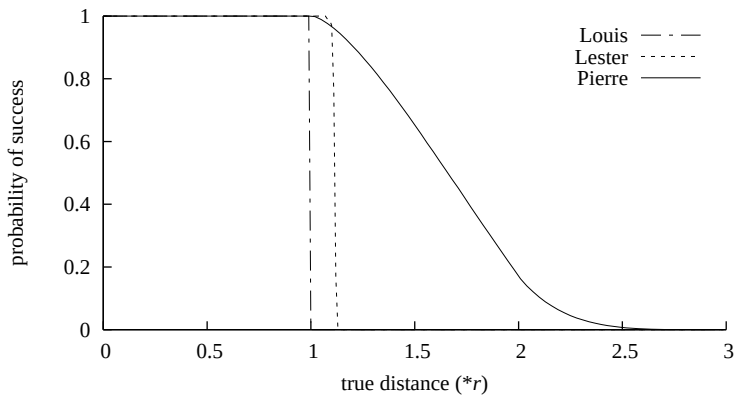


**Fig. 4.** Success probabilities of the three protocols, as a function of the actual distance between Alice and Bob (as a multiple of $r$).

protocol run tells Alice only whether Bob is near the location she entered, and she can extract no other information from Bob's reply.

Like the Lester protocol, the Pierre protocol can gain a minor benefit from using the Boneh-Goh-Nissim cryptosystem. Bob can combine two of his responses and reply with, for example, $\mathcal{E}_A(\rho_1 \cdot D_r \cdot (D_r - 1)), \mathcal{E}_A(\rho_2 \cdot (D_r - 2))$. If the first ciphertext decrypts to 0, then Alice knows that $D_r$ is either 0 or 1, but not which. This gains Bob a small amount of privacy, and at the same time slightly decreases the size of his reply, even taking into account that Boneh-Goh-Nissim is elliptic curve based.

A more dramatic benefit could be gained by using a *ring homomorphic* encryption system; that is, a system in which, given $\mathcal{E}(x)$ and $\mathcal{E}(y)$, one can efficiently compute both $\mathcal{E}(x+y)$ and $\mathcal{E}(x \cdot y)$. With such a system, Bob could reply with the single ciphertext $\mathcal{E}_A(\rho \cdot D_r \cdot (D_r - 1) \cdot (D_r - 2))$. Bob could also include more factors of $(D_r - i)$ inside the encryption while reducing $r$ and be able to more accurately approximate a circle around Alice by using more grid cells of a smaller size. Unfortunately, no secure ring homomorphic cryptosystem is yet known to exist.

## 7 Comparison of the Protocols

In each of our three protocols we say Alice *succeeds* if she discovers Bob is nearby. In some of the protocols, if Alice succeeds, she also learns extra information about Bob's location. We have set up each of our three protocols so that if Alice and Bob are within a distance $r$ of each other, Alice will succeed. In the Louis protocol, the inverse is also true: if Alice and Bob are slightly more than distance $r$ apart, Alice will not succeed. This behaviour does not match realistic use models, however; it is unlikely that Alice will want to learn if Bob is 199 m away, but not if Bob is 201 m away. In our other two protocols, the probability

| Protocol | Louis (first phase only) | Louis (both phases) | Lester | Pierre |
|---|---|---|---|---|
| Extra information learned by Alice | none | Bob's exact location | Bob's exact distance | Bob's grid cell distance |
| Requires third party | ✓ | ✓ | | |
| Bob learns $r$ | ✓ | ✓ | | ✓ |
| Bob learns Alice's location | | ✓ | | |
| Communication steps | 4 | 6 | 2 | 2 |

**Table 3.** Feature comparisons of our three protocols

that Alice succeeds does not fall to 0 as soon as Bob is slightly further than $r$ away; rather, it gradually drops to 0 as Bob gets further, reaching 0 at some outer threshold distance $r_{out}$. That is, if Bob's distance from Alice is less than $r$, Alice will certainly succeed; if his distance is greater than $r_{out}$, Alice will certainly not succeed, and between those values, Alice's probability of success gradually decreases. This seems to fit better with what Alice is likely to want.

In Figure 4 we plot Alice's success probability against Bob's distance from her (in units of $r$), for each of the three protocols. As you can see, all three protocols succeed with probability 1 when the distance is less than $r$. The success probability of the Louis protocol drops immediately to 0 at that point, while the other protocols fall to 0 more gradually. The success probability of the Lester protocol starts dropping slowly as the distance increases past $r$, but then has a rapid decrease to 0 soon after; this is due to the fact that the kangaroo method for finding discrete logarithms has a small chance of succeeding, even if the logarithm in question is outside the expected exponent range. The success probability of the Pierre protocol, on the other hand, decreases to 0 gradually as the distance increases from $r$ to $r_{out} = 2\sqrt{2}r$; this last value is the maximum distance by which Alice and Bob can be separated and still be in diagonally touching cells.

In Table 3 we summarize the properties of our three protocols. For each, we indicate what additional information Alice learns about Bob's location in the event that the protocol succeeds, and whether the protocol requires the participation of a third party. We also indicate whether Bob learns Alice's choice of $r$, whether Bob learns any information about Alice's location, and the number of communication steps.

## 8 Conclusion

We have presented three protocols to solve the nearby-friend problem without requiring a third party that learns location information. Compared to previous work, our protocols require fewer rounds of computation. Moreover, we have demonstrated their feasibility with a sample implementation and its evaluation.

Alerting people of nearby friends is only one of many possible location-based services. A topic of further investigation is what other services can be built with the techniques exploited in this paper.

## Acknowledgments

We thank the anonymous reviewers for their comments. This work is supported by the Natural Sciences and Engineering Research Council of Canada.

## References

1. M. J. Atallah and W. Du. Secure Multi-party Computational Geometry. In *Proceedings of 7th International Workshop on Algorithms and Data Structures*, pages 165–179, August 2001.
2. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *Theory of Cryptography (TCC) '05, Lecture Notes in Computer Science 3378*, pages 325–341. Springer-Verlag, 2005.
3. F. Brandt. Efficient Cryptographic Protocol Design based on Distributed El Gamal Encryption. In *Proceedings of 8th International Conference on Information Security and Cryptology (ICISC)*, pages 32–47, December 2005.
4. C. Cachin. Efficient Private Bidding and Auctions with an Oblivious Third Party. In *Proceedings of 6th ACM Conference on Computer and Communications Security*, pages 120–127, November 1999.
5. R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *Proceedings of 6th Workshop on Privacy Enhancing Technologies (PET 2006), Lecture Notes in Computer Science 4258*, pages 393–412. Springer-Verlag, June 2006.
6. R. Cramer, R. Gennaro, and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Advances in Cryptology—Eurocrypt '97, Lecture Notes in Computer Science 1233*, pages 103–118. Springer-Verlag, 1997.
7. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, `http://www.ietf.org/rfc/rfc4346.txt`, April 2006.
8. W. Du and Z. Zhan. A Practical Approach to Solve Secure Multi-party Computation Protocols. In *Proceedings of 2002 Workshop on New Security Paradigms Workshop*, pages 127–135, September 2002.
9. B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proceedings of 25th International Conference on Distributed Computing Systems (ICDCS 2005)*, June 2005.
10. M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proceedings of First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, May 2003.
11. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology—CRYPTO '96, Lecture Notes in Computer Science 1109*, pages 104–113. Springer-Verlag, August 1996.
12. G. M. Køien and V. A. Oleshchuk. Location Privacy for Cellular Systems; Analysis and Solutions. In *Proceedings of 5th Workshop on Privacy Enhancing Technologies (PET 2005), Lecture Notes in Computer Science 3856*, pages 40–58. Springer-Verlag, May/June 2005.
13. M. Liskov and R. Silverman. A Statistical Limited-Knowledge Proof for Secure RSA Keys. IEEE P1363 working group, 1998.
14. Loopt, Inc. loopt - Live In It. `http://www.loopt.com/`. Accessed February 2007.

15. MIT SENSEable City Lab. iFind. `http://ifind.mit.edu/`. Accessed February 2007.
16. M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006)*, pages 763–774, September 2006.
17. The OpenSSL Project. OpenSSL: The Open Source toolkit for SSL/TLS. `http://www.openssl.org/`. Accessed February 2007.
18. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology—Eurocrypt '99, Lecture Notes in Computer Science 1592*, pages 223–238. Springer-Verlag, 1999.
19. J.M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.
20. D. Shanks. Class number, a theory of factorization, and genera. *Proceedings of Symposia in Pure Mathematics*, 20:415–440, 1971.
21. Victor Shoup. NTL: A Library for doing Number Theory. `http://www.shoup.net/ntl/`. Accessed February 2007.
22. A. C. Yao. Protocols for Secure Computations. In *Proceedings of 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.