CERIAS Tech Report 2004-94
Privacy-Preserving Route Planning
by Mikhail J. Atallah
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Privacy Preserving Route Planning *

Keith B. Frikken
CERIAS and Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
kbf@cs.purdue.edu

Mikhail J. Atallah
CERIAS and Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
mja@cs.purdue.edu

ABSTRACT

The number of location-aware mobile devices has been rising for several years. As this trend continues, these devices may be able to use their location information to provide interesting applications for their owners. Possible applications for such devices include: i) planning a route that brings the owner near a coffee shop or ii) a route that would allow the owner to intersect one of their friends' own route. The difficulty with such computations is that the owners of the devices will not want their devices to be sending their location (or future locations) to some random server to compute the functions. In this paper, we look at computing distance functions of routes in a private manner; we propose using Secure Multi-party Computation (SMC) techniques to solve these computational geometry problems. In this paper we propose protocols for three such problems: i) the distance between a point and a line segment, ii) the distance between two moving points each defined by a parametric equation (with constant velocity), and iii) the distance between two line segments.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce—*Security*

General Terms

Security

Keywords

Privacy, Pervasive Computing, Security Protocols

*Portions of this work were supported by Grants EIA-9903545, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

WPES'04, October 28, 2004, Washington, DC, USA.

1. INTRODUCTION

Suppose Bob is visiting an unfamiliar area, and that Bob has a mobile device that is a route helper that allows Bob to determine if a certain type of store (coffee shop, rare bookstore, etc.) is near a route that he will travel on, but for privacy reasons Bob does not want to reveals his route. Suppose Alice has gathered a large amount of information and that she provides a subscription-based service to access this information. However, Alice cannot simply reveal all of her information to Bob since this reveal all of Alice's proprietary information about certain topics, which is unacceptable because this is how Alice generates revenue (not to mention that Alice's database may be too voluminous to send to Bob). Thus the problem becomes how Bob can know if there is a specific item of interest near his route, without revealing his route and without requiring Alice to reveal all of her information.

A similar application of this would be to help entities with restraining orders placed against them to avoid the person that they are ordered to avoid. Suppose Alice has a restraining order on Bob (perhaps she has caught him being malicious in one too many protocols) that states that he must stay at least 1000 yards from her. Furthermore, suppose that Alice and Bob live in a smaller town, where the chances of them randomly running into each other is not negligible. Clearly, Alice wants to stay away from Bob and Bob would also like to stay away from Alice (as he does not want to cause a scene). Furthermore, Alice and Bob have the right to privacy, and it would be potentially dangerous for Alice to tell Bob where she will be at all times during the day. The problem in this case is that Alice and Bob would like to run a protocol that tells them if their route is safe (i.e., does not get too close to the other party's route) without revealing their own route.

There are also several natural applications for cooperating but mutually distrusting counties or military organizations (so-called "uneasy allies"). Suppose Alice and Bob are two governments who are temporarily cooperating to perform a humanitarian relief (or perhaps a military) operation. Bob has an object (vehicle, convoy, or airplane) that is moving in a space where Alice has certain objects that she wants to hide from Bob. Although Alice and Bob are cooperating, they do not fully trust each other; furthermore the less information is disseminated, the less it is likely that this information will be leaked by an untrustworthy partner (or by a crooked insider employed by an otherwise trustworthy partner). A more extreme example is when both Alice and Bob have moving objects that they want to prevent from getting

too close to each other in order to reduce the likelihood of an accidental collision or of a "friendly fire" accident.

The above problems are all instances of computational geometry problems where the operation being computed is proximity between objects. These objects can take various forms: points, points moving in space (defined by parametric equations), and line segments. Furthermore, the distance between these objects must be computed in a secure fashion. This computation should be done in a way that does not reveal anything other than the result (or what can be computed from the result, as this is unavoidable in any such protocol). A well-known result in the area of Secure Multiparty Computation (SMC) is that any computation can be performed in such a manner. However, these general solutions are expensive. Also, computational geometry problems are difficult in this framework because floating point arithmetic is very expensive, and so operations such as square root and division should be avoided. However, their inverses (multiplication and squaring) are much easier. We assume that the inputs (i.e., point coordinates, coefficients of equations, velocities, etc) are integers – so there is an implied integer "grid". We also assume that all coordinates are non-negative. Furthermore, we initially give protocols for two dimensions, however extending these protocols to higher dimensions is also discussed briefly. In this paper we introduce protocols to solve problems that include the following three:

1. To determine if the distance between a segment and point is smaller than some threshold.

2. To determine if the distance between two points moving with constant velocity, described by parametric equations, stays smaller than some threshold.

3. To determine if the distance between two line segments is smaller than some threshold.

The rest of the paper is organized as follows. In Section 2, we outline related work in this area, and how this paper relates to it. In Section 3, we introduce building blocks and definitions required by our protocols. In Section 4, protocols for the above mentioned problems are given. Finally, in Section 5, a summary of this work is given along with future directions.

2. RELATED WORK

Secure Multi-party Computation (SMC) was introduced in [16], which contained a scheme for secure comparison; suppose Alice (with input a) and Bob (with input b) desire to determine whether or not $a < b$ and without revealing any information other than this result (this is referred to as "Yao's Millionaire Problem"). More generally, SMC allows Alice and Bob with respective private inputs a and b to compute a function $f(a, b)$ by engaging in a secure protocol for public function f. Furthermore, the protocol is private in that it reveals no additional information. By this what is meant is Alice (Bob) learns nothing other than what can be deduced from a (b) and $f(a, b)$. Elegant general schemes are given in [10, 9, 2, 5] for computing any function f privately. However, these general solutions are considered impractical for many problems, and it was suggested in [12] that more efficient domain-specific solutions can be developed.

When developing such protocols, one needs to make sure that the protocols have an advantage over protocols based on the general results. General results in SMC simulate a circuit and require either (depending on the implementation): i) a 1-out-of-2 oblivious transfer (OT) per input wire, constant number of rounds, O(1) invocations of a pseudorandom function per gate, and communication proportional to the number of gates, or ii) an OT per gate and rounds equal to the depth of the circuit. The protocols in this paper have an advantage over such general solutions for several reasons:

1. The circuits require multiplication, and the easiest circuits for k-bit multiplication require $O(k^2)$ gates. There are asymptotic improvements to these circuits, but they come at the cost of large constant factors; the asymptotically best of them (and the worst in terms of having impractically large constant factors) is a circuit of size $O(k \log k \log \log k)$ derived from the textbook Schoenhage-Strassen integer multiplication algorithm (which is itself of mainly theoretical interest, and not used in practice). Our protocols use homomorphic encryption multiplication, which requires $O(k)$ communication. Thus the communication of our protocols is lower.

2. Another savings occurs when the routes are several segments. If Alice has m segments and Bob has n segments, then a general solution will need to have $O(mn)$ versions of the circuit. A nice property of the homomorphic scalar product is that it can be used to do n scalar products with communication proportional to $O(n+m)$. There still will be $O(nm)$ communication to return the results, but it will have a much smaller constant than the general solution.

3. Our protocols are simpler than the general solutions.

In addition to the generic work in the area of SMC, there has been some work in Privacy Preserving Computational Geometry [7, 1]. This work focused on solving three problems: i) the inclusion of a point in a polygon, ii) the intersection of polygons, and iii) the distance between points; solutions were given for all of these problems. The key difference between this and previous work is that this paper focuses on distance protocols rather than on intersection protocols, but some of the protocols overlap. There are two cases where this paper's protocols overlap the work of [7, 1], namely, i) the computation of the distance between points was given in [7, 1], and ii) the determination if two line segments intersect. Instead of placing these protocols in the building block section we place them with the protocols that used them to increase readability, however they are clearly marked as previous work.

3. BUILDING BLOCKS, NOTATIONS, AND DEFINITIONS

3.1 Notations and Definitions

The following are a list of notations that are used within this paper.

1. We use <> to represent vectors.
2. To represent a point in two dimensional space we use the following notation: $P(x, y)$.

3. To represent a line in two dimensions we use $L(A, B, C)$, where the line is described by the equation: $Ax + By + C = 0$.
4. To represent a line segment we use the notation $S(P_1(x_1, y_1), P_2(x_2, y_2))$ to be the segment between the two points P_1 and P_2.
5. To represent a motion of a point along a line with constant velocity we use parametric equations: $x(t) = (v_x)t + x_0$ and $y(t) = (v_y)t + y_0$ to represent the x and (respectively) y coordinates of the moving point. Also, the variable t is in an interval $[s, e]$.
6. We assume that the players in these protocols follow the honest-but-curious (also called semi-honest in the literature) form of behavior, which means that they will follow the protocol steps, but will try to deduce things besides the result. To formalize the notion of security (in the standard way [10]), suppose there is a trusted third party. A protocol that is secure with such a party, would be for Alice and Bob to send their data to this party, who responds with the result. A protocol is secure in the honest-but-curious model if the participants cannot learn anything other than what can be learned in the model with the trusted third party.
7. Sometimes it is desirable to store items in a split manner (i.e., where neither party knows the values, but each party has a share of the value). In this paper, we use three primary types of split data: additive, modular additive, and exclusive-or (XOR). Whenever a value x is split, we denote the respective shares by x' and x'', where $x = (x' + x'', x' + x'' \bmod M, x' \oplus x''$ for additive, modular additive, and XOR split respectively). A problem with additively split data is that it is probabilistically leaky, but it allows for comparison using standard techniques. In this paper, we use only modularly additively split data; we discuss how this is done in more detail in Section 3.4. A note on XOR-split data: in this paper, we use the standard notation and treat 0 as false and 1 as true where $a \oplus b$ is true iff either a or b are true but not both.

3.2 Protocol Building Blocks

In this section we outline various SMC protocols:

1. *Secure Circuit Evaluation*: A well known result in SMC is that circuits can be evaluated with communication equal to the size of the circuit and in constant rounds. Initially, this result was presented in [17], for more details about this work and extensions of this work see the following survey [11]). This can be extended to more than two parties (as in [15]).
2. *Comparison:* Alice and Bob can compare two k-bit integers for the $\leq, <, =, \neq, >, \geq$ predicates where the result is split in one of the previously mentioned ways. This is because the size of such circuits is $O(k)$. More efficient mechanisms (in terms of computation) have been proposed in [3, 8]. Note that [3] requires a non-colluding third party, and [8] gives up accuracy through an accuracy-communication tradeoff. We discuss a method for comparing modular additive split values in Section 3.4.
3. *Dot Product:* This protocol allows Alice and Bob to compute the scalar (i.e., "dot") product of two vectors. It is denoted by $DOTPRODUCT(\vec{v_1}, \vec{v_2})$ where Alice has one vector and Bob has the other (alternatively both vectors are additively split between Alice and Bob). More on the dot product protocol next, in Section 3.3.

3.3 Scalar Product Protocols

Scalar Product protocols were introduced in [7], but this scheme requires several rounds and requires many Oblivious Transfers. Using standard cryptographic techniques it is possible to compute the scalar product in two rounds. Note that for reasons discussed in Section 3.4 we compute the scalar product in a modularly additively split fashion.

This scheme is based on homomorphic encryption. Homomorphic encryption schemes are defined in [14, 6, 13] and have several remarkable properties, including (but not limited to):

1. Semantic Security: Informally, this means that there are many valid encryptions for the value x, and that it is not possible to distinguish two encryptions of the same value.
2. $E(x)E(y) = E(x + y)$. This means that it is possible to add encrypted values; note that the sum is modulo the base of the homomorphic system.
3. $E(x)^c = E(xc)$. This means that it is possible to multiply encrypted values by constants, note that the product is modulo the base of the homomorphic system.

With a semantically-secure homomorphic encryption system, it is obvious that the scalar product can be computed in two rounds.

3.4 On Split Data

In our protocols the intermediate results are stored in a split fashion between the two parties. The reason for this is that we can compose secure protocols for each step to make a secure protocol [4]. More formally, if the intermediate computations are replaced by function calls to trusted oracles, then this would be a secure protocol, and thus the calls to trusted oracles can be replaced by secure protocols. There are two difficulties with such an approach: i) comparing modularly split values and ii) many times the modulus is very large (much larger than the split value) and leads to inefficient protocols. We discuss protocols for comparing split values (Section 3.4.1) and for reducing the modulus of the split value (Section 3.4.2).

3.4.1 Comparing split values

A difficulty with using modular additively split values is comparing the values. While this is very difficult to do efficiently in all cases, when the modulus is more than double the largest possible value being compared (and while one must ensure that this occurs, it is not difficult to choose a large enough modulus to ensure this property) it is roughly twice as difficult (in terms of communication and computation) as standard comparison. The authors are not aware of such a protocol elsewhere.

Input: Alice has two values x' and y' and Bob has two values x'' and y''; all of these values are in the range $[0, M)$. Furthermore the sums $(x' + x'') \bmod M$ (i.e., x) and $(y' + y'') \bmod M$ (i.e., y) are in the range $[0, m)$. It is also known that $M \geq 2m$.

Output: Alice and Bob would like to compute in XOR-split fashion whether or not $(x'+x'') \bmod M \leq (y'+y'') \bmod M$. Note that the protocol below can easily be modified to support other types of comparison.

Notes All arithmetic in the protocols is done modulo M.

1. Alice computes $a \leftarrow y'-x'-m+1$, $b \leftarrow y'-x'$, and $c \leftarrow false$. If $a \geq b$ (i.e., there is wrap-around), then Alice sets her values to $a \leftarrow y'-x'+1$, $b \leftarrow y'-x'+m-1$, and $c \leftarrow true$. Bob computes $d \leftarrow (x''-y'')$.
2. Alice and Bob engage in a protocol using Scrambled Circuit Evaluation to evaluate $(d \geq a) \wedge (d \leq b)$ in an XOR-split fashion, thereby obtaining r_A and r_B. Alice and Bob's respective output is $r_A \oplus c$ and r_B.

Before we prove the correctness of the above protocol, we give two examples which will clarify it. For both of the examples, we suppose that $m = 4$ and the $M = 8$ (satisfying that $M \geq 2m$). In what follows when we define ranges $[a, b]$, we are referring to a range modulo M with the possibility of wrap-around and all arithmetic is done modulo M. For example, if $M = 8$, then the range $[6, 1] \equiv \{6, 7, 0, 1\}$.

Example 1: Suppose $x' = 3$ and $y' = 2$. Alice knows that Bob's values satisfy: $x'' \in \{5, 6, 7, 0\}$ and $y'' \in \{6, 7, 0, 1\}$. The reader can easily verify that, after step 1 of the protocol, (a, b, c) is $(4, 7, false)$ and $(x \leq y) \equiv (x''-y'' \in [4, 7])$.

Example 2: Suppose $x' = 3$ and $y' = 4$. Alice knows that Bob's values satisfy: $x'' \in \{5, 6, 7, 0\}$ and $y'' \in \{4, 5, 6, 7\}$. The reader can easily verify that, after step 1 of the protocol, (a, b, c) is $(2, 4, true)$ and $(x \leq y) \equiv (x''-y'' \in [6, 1])$, which is easily transformed into $(x \leq y) \equiv (x''-y'' \notin [2, 4])$.

Proof of Correctness: Alice knows that:
i) $x'' \in [-x', -x'+m)$ and ii) $y'' \in [-y', -y'+m)$. Suppose the values of x and y are respectively i and j, then $x'' = -x'+i$ and $y'' = -y'+j$. Clearly, $x''-y'' = y'-x'+(i-j)$. From this one can deduce that $(x \leq y) \equiv x''-y'' \in [y'-x'-m+1, y'-x']$ and similarly $(x \leq y) \equiv x''-y'' \notin [y'-x'+1, y'-x'+m-1]$. Since $M \geq 2m$ these ranges must be disjoint and thus at most one of the ranges has wraparound. For a range $[a, b]$ that does not wraparound it is easy to determine if a value z falls in the range by checking if $z \geq a$ and $z \leq b$. The above clearly mimics what the protocol does. **QED**

Proof of Security: We omit a detailed proof as it is obvious based on the composition theorem [4] and a secure protocol for comparison.

3.4.2 Reducing the Base

Another problem with using modularly additively-split values is that many times such values are modulo the base of a homomorphic encryption scheme (i.e., as in scalar product protocol with homomorphic encryption). And if the values are compared with the technique in the previous section, then the cost will be proportional to the number of bits in the homomorphic base, which is typically much larger than the number of bits in the item. We now present a protocol for reducing the base of a split value (this protocol is not used explicitly in our protocols as it is not required, but it will decrease the communication for many of our protocols). We are not aware of this protocol elsewhere.

Input: Alice has a value x' and Bob has a value x''; both of these values are in the range $[0, M)$, but $(x'+x'') \bmod M$ (i.e., x) are in the range $[0, m)$. It is also known that $M \geq 2m$.

Output: Alice and Bob have respective values y' and y'' in the range $[0, m)$, such that $(x'+x'') \bmod M = (y'+y'') \bmod m$.

Steps:

1. Alice generates a random value r chosen uniformly from the range $[0, m)$. Alice computes $a \leftarrow (M \bmod m)$. Alice sets $y' \leftarrow ((x' \bmod m) - r) \bmod m$. Alice creates a list of two values v_0, v_1: If $x' \geq m$, then both values are $(r-a) \bmod m$, else the values are respectively r and $(r-a) \bmod m$.
2. Bob computes the boolean value $i \leftarrow (x'' \geq m)$, where the value is either 0 or 1. Alice and Bob engage in a chosen 1-out-of-2 OT where Bob learns v_i. Bob sets his output to be $y'' \leftarrow (x''+d) \bmod m$.

Proof of Correctness: There are two cases: i) $(x'+x'') \in [0, m)$ (no wraparound) and $(x'+x'') \in [M, M+m)$ (wraparound). In case i, both x' and x'' are smaller than m, and so $y' = (x'-r) \bmod t$ and $y'' = (x''+r) \bmod m$. Thus, $(y'+y'') \bmod m = (x'+x'') \bmod m$, which is correct in this case. In case ii, suppose $(x'+x'') = M+\alpha$, note that the $(y'+y'') \bmod m$ should equal α. Note that the respective outputs will be (since at least one of x' and x'' must be larger than m), $y' = (x'-r) \bmod m$ and $y'' = (x''+r-a) \bmod m$. Note that $(x'+x'') \bmod m = (M+\alpha+r-r-a)$, but since $a = M \bmod m$, this is equal to α. In either case, the claim holds. **QED**

Proof of Security: The main concern is that Alice or Bob could learn something about the value of $(x'+x'') \bmod m$ by engaging in the protocol. Clearly, as long as the OT is a secure protocol, Alice learns nothing as her values are generated from her input. While Bob's view is more complex, it is not difficult to see that every d value could be generated for any value of x', as the value r is chosen uniformly from $[0, m)$. **QED**

4. ROUTE PLANNING PROTOCOLS

In this section we introduce three types of protocols for route planning including: i) the distance between a route (line segments) and a set of fixed objects (points), ii) the distance between two moving objects with constant velocity (parameterized lines), and iii) the distance between a route (line segments) and another route (line segments).

4.1 Points and Line Segments

Suppose Alice has a set of points each associated with a distance and Bob has a route which consists of a set of lines. Furthermore, Alice and Bob want to know if Bob's route gets too close to a point in Alice's set. Note that the protocols can easily be extended to have the threshold associated with the segments instead of the points. In Section 4.1.1, we discuss the general idea of the solution, in Section 4.1.2 the details of the protocols are given, and in Section 4.1.3 the protocols are extended to three dimensions.

4.1.1 General Idea

It is enough to develop a protocol that determines if a specific point and segment are within the threshold of each other in a split manner, because this can be run on all pairs of points and segments and the results can be combined using a straightforward circuit. However, this does reveal the number of points and segments, but this will not be a concern in many cases. If it is a concern, then the actual number can be hidden by inserting several instances of each point or segment (which will not change the end result, but would hide the actual number of points). Thus we consider

only a single point with a threshold and a single segment. A point is within a distance threshold T of a line segment $\overline{AB}$ if and only if at least one of the following holds: i) the point is within T units of point A, ii) the point is within T units of point B, or iii) the point is contained in a rectangle of size $2T$ by $\parallel \overline{AB} \parallel$ and that has $\overline{AB}$ as axis of symmetry (see Figure 1).

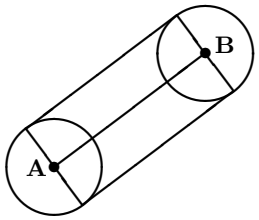

FIGURE 1. *Area of all points within distance T from segment $\overline{AB}$; note that the circles have radius T.*

4.1.2 Protocols

In this section we outline the details of the protocols for determining if a point or line segment are within a certain distance of each other. Protocol 4 is the actual protocol for this, and Protocols 1-3 are utilized by Protocol 4.

PROTOCOL 1. *Distance between Two Points*

Input: Alice has point $P_1(x_1, y_1)$ and threshold T. Bob has a point $P_2(x_2, y_2)$.

Output: Computes (in XOR split manner) if the distance between P_1 and P_2 is $\leq T$.

Description: The distance between P_1 and P_2 is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. This distance being smaller than the threshold T is equivalent to $(x_1 - x_2)^2 + (y_1 - y_2)^2 \leq T^2$. This protocol is also presented in [7, 1]. The left hand side can be computed with a single dot product, and then the comparison can be done.

Steps:

1. Alice computes $\vec{a} =< {x_1}^2, -2x_1, 1, {y_1}^2, -2y_1, 1 >$. Likewise, Bob computes
   $\vec{b} =< 1, x_2, {x_2}^2, 1, y_2, {y_2}^2 >$.
2. Alice and Bob engage in protocols to compute $d = DOTPRODUCT(\vec{a}, \vec{b})$ in a modular additively split manner.
3. Alice and Bob compute the predicate $(d \leq T^2)$ in an XOR split manner.

PROTOCOL 2. *Distance between a Point and a Line*

Input: Alice has point $P(x, y)$ and threshold T. Bob has line $L(A, B, C)$.

Output: Computes if the distance between P and L is $\leq T$.

Description: The distance between P and L is $|\frac{Ax+By+C}{\sqrt{A^2+B^2}}|$. This distance being smaller than the threshold T is equivalent to $(Ax + By + C)^2 \leq T^2(A^2 + B^2)$. The left hand side and right hand side can be computed with a single dot product, and then the comparison can be performed.

Steps:

1. Alice computes $\vec{a_1} =< x^2, 2xy, 2x, y^2, 2y, 1 >$ and $\vec{a_2} =< T^2 >$. Likewise, Bob computes
   $\vec{b_1} =< A^2, AB, AC, B^2, BC, C^2 >$ and $\vec{b_2} =< A^2 + B^2 >$.
2. Alice and Bob engage in protocols to compute the predicate:
   $DOTPRODUCT(\vec{a_1}, \vec{b_1}) \leq DOTPRODUCT(\vec{a_2}, \vec{b_2})$.

PROTOCOL 3. *Point between Two Lines*

Input: Alice has point $P(x, y)$. Bob has two parallel lines $L_1(A, B, C_1)$ and $L_2(A, B, C_2)$ (we assume that L_2 is above L_1 (i.e., $C_2 > C_1$).

Output: Computes if P lies between L_1 and L_2.

Description: In order for P to be between L_1 and L_2 it must be such that $C_1 \leq Ax + By \leq C_2$. If this is the case, then P will be above L_1 and below L_2.

Steps:

1. Alice computes $\vec{a} =< x, y >$. Likewise, Bob computes $\vec{b} =< A, B >$
2. Alice and Bob engage in protocols to compute $d = DOTPRODUCT(\vec{a}, \vec{b})$ in a modular additively split fashion.
3. Alice and Bob compute the predicate $(d \geq C_1) \wedge (d \leq C_2)$.

PROTOCOL 4. *Distance between a Point and a segment*

Input: Alice has point $P(x_p, y_p)$ and a threshold T. Bob has a line segment $S((x_1, y_1), (x_2, y_2))$.

Output: Computes if the distance between P and S is $\leq T$.

Description: As discussed in Section 4.1.1 all that needs to be checked is if the endpoints are within T from P (for which Protocol 1 can be used) or are within a rectangle with dimensions of size $2T$ by $\parallel S \parallel$ that has S as axis of symmetry (see Figure 1). P is within this rectangle if it is no more than T units away from the line through S (for which Protocol 2 can be used) and if P is within the lines perpendicular to S at its endpoints (for which Protocol 3 can be used).

Steps:

1. Alice and Bob engage in Protocol 1 on (x_p, y_p) and (x_1, y_1) and threshold T, with result γ_1 obtained in an XOR-split fashion.
2. Alice and Bob engage in Protocol 1 on (x_p, y_p) and (x_2, y_2) and threshold T, with result γ_2 known in an XOR-split fashion.
3. Bob computes the line through his segment, which is $L(A, B, C)$ where $A = y_2 - y_1$, $B = x_1 - x_2$, and $C = x_1(y_1 - y_2) + y_1(x_2 - x_1)$. Alice and Bob engage in Protocol 2 on P and L and threshold T, with result γ_3 known in an XOR-split fashion.
4. Bob computes perpendicular lines through the segment endpoints, denoted by $L_1 = (x_2 - x_1)x + (y_2 - y_1)y + (x_1 - x_2)x_1 + (y_1 - y_2)y_1$ and $L_2 = (x_2 - x_1)x + (y_2 - y_1)y + (x_1 - x_2)x_2 + (y_1 - y_2)y_2$. Alice and Bob engage in Protocol 3 with inputs P, L_1, and L_2 with result γ_4 known in an XOR-split fashion.

5. Alice and Bob output $\gamma_1 \vee \gamma_2 \vee (\gamma_3 \wedge \gamma_4)$ using secure circuit evaluation.

Complexity Analysis:
Note that steps 1-4 in Protocol 4 can be done in parallel. Furthermore, step 5 can be done in $O(1)$ rounds with $O(1)$ communication. Thus Protocol 4 can be done with $O(1)$ communication and $O(1)$ rounds.

4.1.3 Extending to higher dimensions

Extending these protocols to three dimensions can be achieved by extending each of the base protocols (1-3) to three dimensions.

1. Protocol 1 can be extended to three dimensions easily by changing the value $(x_1 - x_2)^2 + (y_1 - y_2)^2$ to $(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$ which can easily be computed with dot product.

2. Protocol 2 is not as direct to translate to 3 dimensions, but there is a formula for determining the distance between a point and a line segment in 3 dimensions. The idea behind the formula is to take a vector on the line segment and a vector with the same origin to the point, and then take the magnitude of the cross product, i.e., the area of the parallelogram created by these vectors (which can be computed with dot product) and then divide by the length of the vector on the line, which reveals the height of the parallelogram and is thus the distance between the point and the line.

3. Protocol 3 can easily be extended to three dimensions, where it becomes whether or not a point is located between two planes. The equation for a plane is of the form $Ax + By + Cz + D = 0$, and thus the previous technique will work.

4.2 Parameterized Lines

Suppose Alice has a route which consists of several line segments and Bob has a route which consists of a set of line segments. Furthermore, suppose that there is an object that is traveling on each of Alice and Bob's routes whose position can be described by a point with constant velocity (with parametric equations) and that Alice has a threshold distance for which she does not want her object to get close to Bob's object.

4.2.1 General Idea

For this protocol we assume that Alice and Bob's time intervals are not secret (i.e., only the location, direction, and velocity during the intervals are private, but it is okay to reveal the time when the direction or velocity changes), and thus Alice and Bob can easily compute all time intervals where there is a change (in direction or velocity) by one of the objects. In this section, we give a protocol to determine if the distance between Alice and Bob's objects (during the time interval) is under some threshold. Suppose Alice's point can be described by $x_1(t) = v_{x,1}t + x_{0,1}$ and $y_1(t) = v_{y,1}t + y_{0,1}$, and likewise Bob's position can be described as $x_2(t) = v_{x,2}t + x_{0,2}$ and $y_2(t) = v_{y,2}t + y_{0,2}$. The distance between the points at any time t can be described by $\sqrt{(x_1(t) - x_2(t))^2 + (y_1(t) - y_1(t))^2}$. This being less than T is equivalent to $f(t) < T^2$ where $f(t)$ is a quadratic polynomial of t (denote it by $At^2 + Bt + C$). Now, if $A \leq 0$ then the minimum value during the interval will be at one of the endpoints of the time interval. However, if $A > 0$, then it is possible for a minimum to be inside the interval, in this case Protocol 5 computes whether or not the minimum is inside the interval and if so outputs whether or not the minimum is smaller than T. Protocol 6 combines the results from the endpoints and from Protocol 5 to determine the final answer.

4.2.2 Protocols

PROTOCOL 5. *Quadratic polynomial minimization*

Input: Alice and Bob have a polynomial $f(t) = At^2 + Bt + C$ where $t \in [s, e]$ and where the values A, B, C, s, and e are additively split between them. Furthermore, Alice has a threshold T^2.

Output: The predicate "the minimum of $f(t)$ is smaller than T^2 and this minimum occurs in the interval $[s, e]$".

Description: The polynomial $f(t)$ will have a minimum iff $A > 0$. If $A < 0$ it will be a downward parabola (i.e., it will have a maximum) and if $A = 0$ it will be a line (i.e., it will not have a maximum or minimum). The point the minimizes $f(t)$ can be easily found to be $t_{min} = \frac{-B}{2A}$. Assuming $A > 0$, then the minimum is in the interval $[s, e]$ iff $2As \leq -B \leq 2Ae$. Finally, the minimum value of $f(t)$ (assuming that there is such a minimum) is $f(t_{min}) = \frac{B^2}{4A} - \frac{B^2}{2A} + C$. Thus $(f(t_{min}) < T^2) \equiv (-B^2 + 4AC < 4AT^2)$.

Steps:

1. *Determine if a minimum exists to $f(t)$.* Alice and Bob engage in a protocol $\gamma_1 = (A > 0)$ in an XOR-split fashion.

2. *Assuming that there is a minimum determine if the minimum exists in $[s, e]$.* Alice and Bob engage in a protocol to determine: $\gamma_2 = (2As \leq -B) \wedge (-B \leq 2Ae)$ in an XOR-split fashion. Note that the values inside these comparisons can be computed with secure dot product.

3. *Assuming that a minimum exists, is it smaller than T^2.* Alice and Bob engage in a protocol to determine $\gamma_3 = (-B^2 + 4AC \leq 4AT^2)$ in an XOR-split fashion. Note that these values can be computed with secure dot product.

4. Alice and Bob compute $\gamma_1 \wedge \gamma_2 \wedge \gamma_3$.

PROTOCOL 6. *Two Parameterized Line Segments*

Input: Alice has segment $x_1(t) = v_{x,1}t + x_{0,1}$, $y_1(t) = v_{y,1}t + y_{0,1}$, and a threshold T. Bob has a line segment $x_2(t) = v_{2,B}t + x_{0,2}$ and $y_2(t) = v_{y,2}t + y_{0,2}$. The time interval is known to Alice and Bob and is represented by s to e. Furthermore, the parameterized segments are additively split between Alice and Bob.

Output: Computes if the distance between the two segments is $< T$ at any point during the time interval.

Description: The distance between the two points during an interval is $\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$. Since all terms are linear in t, this means that there is a polynomial $f(t) = At^2 + Bt + C$ such that the distance is $\sqrt{f(t)}$. This is smaller than threshold T iff $f(t) \leq T^2$. Either $f(t)$ will have a global

minimum in the interval $[s, e]$ or the minimum value in that interval is $\min\{f(s), f(e)\}$. Protocol 5 can be used to determine if a global minimum is smaller than T^2 and a dot product protocol can be used to determine if $f(s) \leq T^2$ or $f(e) \leq T^2$.

Steps:

1. Alice and Bob engage in dot product protocols to determine the distance function $f(t) = At^2 + Bt + C$. The values A, B, and C are additively split between Alice and Bob. To avoid cluttering the paper, we omit the inclusion of the exact details of these dot product computations.
2. Alice and Bob determine if at either endpoint of the interval, the distance between their points is smaller than T:
   (a) Alice and Bob compute $d_s = As^2 + Bs + C$ and $d_e = Ae^2 + Be + C$ in an additively split fashion. Note that these values can easily be computed without communication.
   (b) Alice and Bob engage in protocols to determine: $\gamma_1 = (d_s \leq T^2) \vee (d_e \leq T^2)$ in an XOR-split fashion.
3. Alice and Bob use protocol 5 with parameters A, B, C, s, e, and T^2 to determine if there is a point in the interval that is a minimum less than T^2. The result is obtained in γ_2 in an XOR-split fashion.
4. Alice and Bob output $\gamma_1 \vee \gamma_2$.

Complexity Analysis:

Much of protocol 6 can be done in parallel. The protocol requires $O(1)$ communication and $O(1)$ rounds.

4.2.3 Extending to Higher Dimensions

This protocol is easily extended to arbitrary dimensions. All that needs to be changed is the computation of the the polynomial $f(t)$. Note that the distance between the two points at any given time t will still be $\sqrt{f(t)}$ for some quadratic polynomial $f(t)$.

4.3 Lines and Lines

Suppose Alice has a route which consists of several line segments each associated with a distance and Bob has a route which consists of a series of segments. Furthermore, Alice and Bob want to know if Bob's route gets within Alice's specified distance for a specific segment of Alice's route. Note that the protocols can easily be extended to have the threshold associated with either set of segments. In Section 4.3.1, we discuss the general idea of the solution. In Section 4.3.2, the details of the protocols are given.

4.3.1 General Idea

It is enough to develop a protocol that determines if two segments are within the threshold of each other in a split manner, because this can be run on all pairs of points and segments and the results can be combined using a straight forward circuit. Although this reveals the number of segments, this will not be a concern in many cases. If it is a concern, then the actual number can be hidden by inserting several instances of each segment (which will not change the end result, but would hide the actual number of points). Thus we consider only two segments and a threshold.

In two dimensions, either the segments will intersect or the closest point to the segment will involve one of the endpoints. Thus Protocol 4 can be used 4 times (once with each endpoint and the other segment) to handle the second case, and all that is needed is a protocol for determining if two segments intersect, for which a protocol is given in [7, 1].

4.3.2 Protocols

PROTOCOL 7. *Line and Line Segment Intersection*

Input: Alice has line $L(A, B, C)$ and Bob has a line segment $S((x_1, y_1), (x_2, y_2))$.

Output: Computes if L and S intersect.

Description: L and S will intersect iff there is the endpoints of S lie on different sides of L or one or more of the endpoints is on L. The side of a point $P(x, y)$ on L is the sign of $Ax + By + C$. Essentially if the value of this is negative for one endpoint and is positive for the other, then the segment intersects the line. This protocol is also given in [7, 1].

Steps:

1. Alice computes $\vec{a} =< A, B, C >$. Likewise, Bob computes $\vec{b_1} =< x_1, y_1, 1 >$ and $\vec{b_2} =< x_2, y_2, 1 >$.
2. Alice and Bob engage in protocols to compute $d_1 = DOTPRODUCT(\vec{a}, \vec{b_1})$ in a modular additively split fashion.
3. Alice and Bob engage in a protocol to compute $d_2 = DOTPRODUCT(\vec{a}, \vec{b_2})$ in a modular additively split fashion.
4. In parallel Alice and Bob engage in secure protocols to compute in an XOR-split fashion $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ with respective values $(d_1 \leq 0), (d_2 \leq 0), (d_1 \geq 0), (d_2 \geq 0)$
5. Alice and Bob output $(\gamma_1 \wedge \gamma_4) \vee (\gamma_2 \wedge \gamma_3)$.

PROTOCOL 8. *Two Line Segments*

Input: Alice has segment $S_1(P_1(x_1, y_1), P_2(x_2, y_2))$ and a threshold T. Bob has a line segment $S_2(P_3(x_3, y_3), P_4(x_4, y_4))$.

Output: Computes if the distance between S_1 and S_2 is $< T$.

Description: As mentioned in Section 4.3.1 all that needs to be done is verify if any endpoint is within T of the other segment (Step 1) or to see if the segments intersect. The segments will intersect iff both segments intersect the line through the other segment (for which Protocol 7 can be used, Steps 2-3).

Steps:

1. In parallel Alice and Bob engage in secure protocols to compute in an XOR-split fashion $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ where the values are the result of execution of Protocol 4 (distance between a point and a line segment) with respective inputs (P_1 and S_2), (P_2 and S_2), (P_3 and S_1), and (P_4 and S_1).
2. Bob computes the line through his segment, which is $L_2(A, B, C)$ where $A = y_4 - y_3$, $B = x_3 - x_4$, and $C = x_3(y_3 - y_4) + y_3(x_4 - x_3)$. Alice and Bob engage in Protocol 7 on S_1 and L_2 obtaining the results in γ_5 in an XOR-split fashion.

3. Alice computes the line through her segment, which is $L_1(A, B, C)$ where $A = y_2 - y_1$, $B = x_1 - x_2$, and $C = x_1(y_1 - y_2) + y_1(x_2 - x_1)$. Alice and Bob engage in Protocol 7 on S_2 and L_1 storing the results in γ_6 in an XOR-split fashion.

4. Alice and Bob engage in secure protocols to determine: $\gamma_1 \vee \gamma_2 \vee \gamma_3 \vee \gamma_4 \vee (\gamma_5 \wedge \gamma_6)$.

Complexity Analysis:
Much of Protocol 8 can be done in parallel. The protocol requires $O(1)$ communication and $O(1)$ rounds.

4.3.3 Extending to Higher Dimensions

This protocol can be extended to three dimensions. This is because the minimum distance between two 3-d line segments is either the distance from one of the endpoints to the other segment or is the minimum distance between the lines defined by the segments (if this minimal point is on the segments). This can be computed with similar techniques as the two dimensional protocol.

5. SUMMARY

In this paper we have introduced privacy-preserving computational geometry protocols with applications to route planning. These applications include: planning of routes that do (or do not) get close to certain objects, planning for routes with constant velocities that do not get within a certain distance of each other, and planning routes of objects that do not get within certain distances of each other without knowing the exact velocity of the objects beforehand. Future directions of this work include:

1. Distance between parametric lines with acceleration.
2. An implementation of the protocols.
3. A system that computes routes that do not get within a certain range of each other. For example, given a set of points with threshold distances and a start and end point, compute a route from the start to the end point without violating the distance thresholds.
4. One difficulty with route planning protocols is the requirement that the device know where it is at, which would seem to require a some form of query to a GPS system, but this would reveal the location of the device. Is there a way to privately determine your position without revealing information about your location?

Acknowledgment.

The authors are grateful to the reviewers for their useful suggestions, including the "future research" suggestion 4.

6. REFERENCES

[1] Mikhail J. Atallah and Wenliang Du. Secure multi-party computational geometry. *Lecture Notes in Computer Science*, 2125:165–179, 2000.

[2] Michael Ben-Or and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM Press, 1988.

[3] Christian Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 120–127. ACM Press, 1999.

[4] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[5] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM Press, 1988.

[6] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001*, LNCS 1992, pages 119–136, 2001.

[7] W. Du. *A Study of Several Specific Secure Two-party Computation Problems*. PhD thesis, Purdue University, West Lafayette, Indiana, 2001.

[8] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *RSA Security 2001 Cryptographer's Track*, LNCS 2020, pages 457–471, 2001.

[9] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229. ACM Press, 1987.

[10] Oded Goldreich. Secure multi-party computation. Working Draft, 2000.

[11] Oded Goldreich. Cryptography and cryptographic protocols. *Distrib. Comput.*, 16(2-3):177–199, 2003.

[12] Shafi Goldwasser. Multi party computations: past and present. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6. ACM Press, 1997.

[13] T. Okamoto, S. Uchiyama, and E. Fujisaki. Epoc: Efficient probabilistic public-key encryption, 1998.

[14] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Application of Cryptographic Techniques, EUROCRYPT 99*, LNCS 1592, pages 223–238, 1999.

[15] P. Rogaway. *The Round Complexity of Secure Protocols*. Ph.d. thesis, MIT, 1991.

[16] A.C Yao. Protocols for secure computation. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[17] A.C Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.