Coordinated multi-robot planning while preserving individual privacy

Li Li[1], Alfredo Bayuelo[2], Leonardo Bobadilla[1], Tauhidul Alam[3], and Dylan A. Shell[4]

***Abstract*— We consider the problem of multiple robots that must cooperate within a shared environment, but which wish to limit the information they disclose during their coordination efforts. Specifically, we examine the problems of privacy-preserving rendezvous and persistent monitoring. In the former, the robots construct a joint plan to have them meet, without either knowing beforehand where or when the meeting will occur. In the latter, multiple robots dynamically cover a region of space—they plan collective motions which are collision-free but with the assurance that agents remain ignorant of the paths of others. Accordingly, the tasks are sort of inverses in that the robots must collectively determine whether their joint paths collide or not, then, using this, achieve their collective task. Other than what is learned by the outcome of the joint-collision determination, the robots possess no details of the other paths. Our approach builds on garbled circuits and homomorphic encryption to realize basic secure path intersection primitives. We present algorithms, a software implementation, and a physical experiment on mobile robots to test the practical feasibility of our approach. We believe that these ideas provide a valuable direction for adoption in small Unmanned Systems belonging to different stakeholders.**

I. Introduction

If many current predictions are to be believed, autonomous robots will be increasingly used in shared, contested, resource constrained, and adversarial scenarios. Certainly these traits underscore tasks such as automated delivery, battlefield awareness, and surveillance. In each of these cases, multiple robots operating concurrently often can achieve their ends more efficiently by cooperating to mediate their use of shared resources. But, as the information that the robots possess is sensitive or restricted, this poses the question of how to preserve individual privacy whilst coordinating. More broadly, privacy preservation is becoming a growing concern in robotics, and this paper examines several particular scenarios where we envision it being relevant to the context of coordination among robots.

The following two examples, one within the commercial/civilian context and another with a military setting, provide motivating scenarios:

– An obstacle to the widespread commercial use of small Unmanned Aerial Vehicles (UAVs) is the potential for collisions, not only against each other but also against larger manned aircraft [1]. As various airborne vehicles are owned and operated by different companies and stakeholders, it would be ideal if one could provide assurances of collision-free paths without mandating the disclosure of information that some parties would be uneasy revealing.

– Consider a covert mission against some adversary where multiple robots must coordinate, for example, to rendezvous behind enemy lines. In the event of one of the robots being abducted or compromised, we would like guarantees that any information that might be extracted from the captured agent will not make the other robots vulnerable.

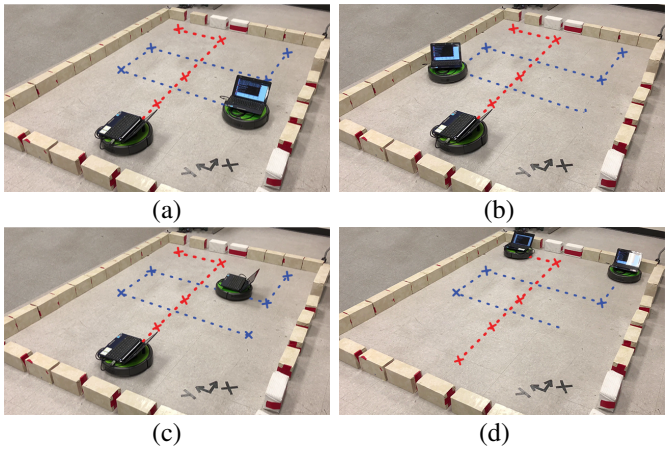

Fig. 1. An experimental evaluation of the privacy-preserving monitoring task for two parties Alice and Bob using two iRobot Create 2.0 platforms: (a) At time = 0 second, Alice and Bob start executing their paths; (b) Since an intersection is found (without sharing information), Alice moves and Bob stays still; (c) At time = 20 seconds, Alice finishes her path; (d) At time = 40 seconds, both robots have reached their goals with no collisions and without either revealing the path details to the other party.

The above mobile robotics applications can be modeled in a Secure Multi-Party Computation (SMC) [2], [3], [4] Framework. We think that SMC can model an important class of problems in mobile robotics for what we call *mutually distrusting cooperation*; in this type of problem one has multiple interacting robots that need to achieve tasks jointly within a shared environment but wish to limit the disclosure of their information.

The contribution of our paper is showing the practical feasibility of Secure Multi-Party Computation in Robotics and Autonomous Systems. We hope to attract the attention of other researchers to this area. There have been few practical implementations of SMC, and to the best of our knowledge, this is one of the first implementations of SMC in Robotics. More concretely, the contributions of our paper are:

1) A secure path intersection protocol based on the polygon intersection ideas presented in [5] and simplified

[1]Li Li and Leonardo Bobadilla are with the Florida International University, Miami, FL, USA. lli031@fiu.edu, bobadilla@cs.fiu.edu

[2]Alfredo Bayuelo is with the National University of Colombia, Bogota, Colombia. ajbayuelos@unal.edu.co

[3]Tauhidul Alam is with SUNY Old Westbury, Old Westbury, NY, USA. alamt@oldwestbury.edu

[4]Dylan A. Shell is with the Texas A&M University, College Station, TX, USA. dshell@cse.tamu.edu

to make its implementation feasible using open source software packages.
2) A protocol that ensures that two robots will not collide while executing their task, its software implementation, and hardware proof of concept experiments.
3) A new, secure 3D path collision protocol that can be used in plans involving time or 3D workspaces.

The rest of the paper is organized as follows. Section II describes related efforts in the literature. Section III introduces our model, the environment, and robot capabilities; and formulates the problem of interest. Section IV presents the proposed methods developed to address the problems of interest. Section V presents simulation and software development and physical deployment to show the feasibility of our approach. Finally, we show preliminary conclusions and directions for future work in Section VI.

II. Related work

The motivation of our work comes from the area of Secure Multi-Party Computations [2], [3] where a group of players needs to compute a joint function without disclosing their inputs. A well known particular case of this formulation is *Yao's Millionaire Problem* where two millionaires Alice and Bob want to know who has the most money without revealing their wealth to each other. The approach is general and can solve any function that can be encoded as a circuit. However, if the function is complicated, the use of *garbled circuits* will not be practical. This limitation has led to the development of specific protocols for SMC problems in particular domains [6] such as Linear Algebra, Statistics, Machine Learning, Networking, and Computational Geometry.

In particular, due to the close connection of planning algorithms to Computational Geometry, our ideas take inspiration from *Secure Computational Geometry* [7], [5], [8], [9], [10] that proposes geometric constructions to computational problems involving intersections [5] (point in polygon, polygon-polygon intersection) or based on distance [7] (distance between parametric equations and line segments). These secure geometric constructions use the primitives such as Yao's millionaires, Garbled Circuits, 1-out-N oblivious transfer, and Homomorphic Encryption. In our work, we build our protocols on top of some of these primitives, specifically from [5] and simplify them to implement using open source software and inexpensive robot platforms. We also create new secure primitives to compare paths in 3D as required by our applications. Our effort is also connected to practical system implementations of SMC [11].

Security issues in multi-robot networks have been the focus of recent research [12], [13]. In [13], defense mechanisms for Sybil attacks have been proposed and implemented in commodity Wi-Fi radios. Their approach has also been tested in mobile robotic platforms. Privacy issues related to robots have also been investigated in several recent works [14], [15], [16]. Among the investigated robotic privacy techniques, a differential privacy model is proposed in [16] for swarms of heterogeneous robots, also, combinatorial filters are designed in [14] satisfying privacy and utility constraints which have been explored for the privacy-preserving target tracking [15].

III. Preliminaries

In this section, we briefly present the model definition. Then, we formally state the problems that are addressed.

A. Model Definition

The robots move in a 2-D workspace $\mathcal{W} = \mathbb{R}^2$. The free space of the environment is defined as $E = \mathcal{W} \setminus \mathcal{O}$, where $\mathcal{O} \subset \mathbb{R}^2$. We initially have two robots Alice and Bob that operate in a shared environment. Both robots have a representation of the environment E, can plan obstacle-free paths in the environment, and can communicate with each other. Let P_A be the path of Alice and P_B be the path of Bob.

B. Problem Formulation

In our first primitive, we require a privacy-preserving mechanism to allow for Alice and Bob to determine whether their paths (P_A and P_B) collide without revealing the path information to the other party. We propose to do this in a decentralized fashion and without relying on the existence of any trusted third-party. These constraints motivate our first problem of interest.

Problem 0. Privacy-Preserving Path Intersection: *Given two robots, Alice and Bob with paths P_A and P_B, inform the robots whether the paths have at least one point of intersection without sharing the path information.*

We will use the above problem as a building block for solving more pragmatic tasks in a privacy-preserving fashion. We are interested in a continuous monitoring task where both robots are executing the task but need to guarantee collision avoidance. This motivates the following problem of interest.

Problem 1. Privacy-Preserving Persistent Monitoring: *Given two robots Alice and Bob that are executing a mission in E, ensure, without sharing any information about their paths or position, that they will not collide.*

We are also interested in problems involving time-parametrized trajectories. For this purpose, we need to construct secure primitives to calculate if two 3D line segments intersect. This primitive will allow us to solve the following problem.

Problem 2. Ascertaining Rendezvous Securely: *There are two robots, Alice and Bob, and each of them has a time-parametrized trajectory. They want to know if their paths intersect without sharing either respective paths or the intersection point and time.*

IV. Methods

In this section, we detail our method for solving the problems formulated in Section III.

Initially, we need to implement a primitive that can test if two paths intersect. To create a practical implementation, we build our protocol based on the polygon intersection protocol presented in [5]. This protocol uses a secure dot product implementation along with Yao's Millionaires, Garbled

circuits, 1-out-of-N oblivious transfer, and homomorphic encryption. We simplify this protocol, adapt it for the 2-D path intersection, and make it suitable for implementation on mobile robots.

A. Privacy-Preserving Path Intersection

A path P of a robot is represented by a sequence of contiguous segments $P = (S_1, S_2, \ldots, S_n)$. Each segment S_i is composed by its two points $\mathbf{u_i} = (x_i, y_i)$ and $\mathbf{v_i} = (x_i', y_i')$. Meanwhile, we use a line equation $f(x, y) = 0$ to represent the line that contains the segment, where $f(\mathbf{u}) = f(x, y) = ax + by + c$. We can easily calculate a, b, and c by providing $\mathbf{u_i}$ and $\mathbf{v_i}$.

Suppose Alice has a segment $S_A = (\mathbf{u_A}, \mathbf{v_A})$ and the line equation $f_A(x, y) = 0$ that contains the vertices $\mathbf{u_A}, \mathbf{v_A}$, and Bob has a segment $S_B = (\mathbf{u_B}, \mathbf{v_B})$ and the line equation $f_B(x, y) = 0$ that contains the vertices $\mathbf{u_B}, \mathbf{v_B}$. Then, S_A intersects with S_B if and only if one of the following expression is true:
$f_A(\mathbf{u_B}) \leq 0 \wedge f_A(\mathbf{v_B}) \geq 0 \wedge f_B(\mathbf{u_A}) \leq 0 \wedge f_B(\mathbf{v_A}) \geq 0$,
$f_A(\mathbf{u_B}) \leq 0 \wedge f_A(\mathbf{v_B}) \geq 0 \wedge f_B(\mathbf{u_A}) \geq 0 \wedge f_B(\mathbf{v_A}) \leq 0$,
$f_A(\mathbf{u_B}) \geq 0 \wedge f_A(\mathbf{v_B}) \leq 0 \wedge f_B(\mathbf{u_A}) \leq 0 \wedge f_B(\mathbf{v_A}) \geq 0$,
$f_A(\mathbf{u_B}) \geq 0 \wedge f_A(\mathbf{v_B}) \leq 0 \wedge f_B(\mathbf{u_A}) \geq 0 \wedge f_B(\mathbf{v_A}) \leq 0$.
To make the computation of the substitution easier, we define an operator $\sqcup$ such that $\mathbf{u} \sqcup 1 = (x, y, 1)$, where $\mathbf{u} = (x, y)$ is a vector.

Protocol 1 Secure Path Intersection Protocol

Input: Given Alice's path $P_A = (S_{A_1}, S_{A_2}, \ldots, S_{A_n})$ and Bob's path $P_B = (S_{B_1}, S_{B_2}, \ldots, S_{B_n})$.
Output: Whether there is a collision between P_A and P_B.

1) Alice generates a public/private key pair (k_A^{pub}, k_A^{pri}) using the Paillier homomorphic encryption system.
2) For each pair of segments (S_A, S_B) where $S_A = (\mathbf{u_A}, \mathbf{v_A})$ and $S_B = (\mathbf{u_B}, \mathbf{v_B})$:
   a) Alice calculates (a_1, b_1, c_1). Bob calculates (a_2, b_2, c_2).
   b) Alice generates a vector $\mathbf{m} = (E_{k_A^{pub}}(a_1), E_{k_A^{pub}}(b_1), E_{k_A^{pub}}(c_1))$. Bob assigns a vector $\mathbf{n} = (a_2, b_2, c_2)$.
   c) Alice generates two vectors $\mathbf{p_A} = (E_{k_A^{pub}}(\mathbf{u_A}.x), E_{k_A^{pub}}(\mathbf{u_A}.y), E_{k_A^{pub}}(1))$ and $\mathbf{q_A} = (E_{k_A^{pub}}(\mathbf{v_A}.x), E_{k_A^{pub}}(\mathbf{v_A}.y), E_{k_A^{pub}}(1))$. Bob generates two vectors $\mathbf{p_B} = \mathbf{u_B} \sqcup 1$ and $\mathbf{q_B} = \mathbf{v_B} \sqcup 1$.
   d) Alice sends $\mathbf{m}, \mathbf{p_A}$ and $\mathbf{q_A}$ to Bob.
   e) Bob calculates w_1, w_2, w_3, w_4, where $w_1 = \mathbf{m} \cdot \mathbf{p_B}, w_2 = \mathbf{m} \cdot \mathbf{q_B}, w_3 = \mathbf{n} \cdot \mathbf{p_A}, w_4 = \mathbf{n} \cdot \mathbf{q_A}$.
   f) Bob generates 4 random numbers r_1, r_2, r_3, r_4 and calculates h_1, h_2, h_3, h_4, where $h_i = w_i + r_i, i = 1, 2, 3, 4$.
   g) Bob sends h_1, h_2, h_3, h_4 back to Alice.
   h) Alice computes $t_i = D_{k_A^{pri}}(h_i)$ where $i = 1, 2, 3, 4$.
   i) Alice and Bob use a garbled circuit to check if the following expression is true: $(t_1 \leq r_1 \wedge t_2 \geq r_2 \wedge t_3 \leq r_3 \wedge t_4 \geq r_4) \vee (t_1 \leq r_1 \wedge t_2 \geq r_2 \wedge t_3 \geq r_3 \wedge t_4 \leq r_4) \vee (t_1 \geq r_1 \wedge t_2 \leq r_2 \wedge t_3 \leq r_3 \wedge t_4 \geq r_4) \vee (t_1 \geq r_1 \wedge t_2 \leq r_2 \wedge t_3 \geq r_3 \wedge t_4 \leq r_4)$.
   j) Return **True** if the garbled circuit returns $true$.
3) Return **False**.

Let Alice compose three vectors $\mathbf{m} = (a_1, b_1, c_1), \mathbf{p_A} = \mathbf{u_A} \sqcup 1, \mathbf{q_A} = \mathbf{v_A} \sqcup 1$, and let Bob compose three vectors $\mathbf{n} = (a_2, b_2, c_2), \mathbf{p_B} = \mathbf{u_B} \sqcup 1, \mathbf{q_B} = \mathbf{v_B} \sqcup 1$, then $i = \mathbf{m} \cdot \mathbf{p_B}, i' = \mathbf{m} \cdot \mathbf{q_B}, j = \mathbf{n} \cdot \mathbf{p_A}, j' = \mathbf{n} \cdot \mathbf{q_A}$.

So far, if Alice sends $\mathbf{m}, \mathbf{p_A}, \mathbf{q_A}$ to Bob, Bob can easily compute i, i', j, j' and do the intersection determination logic and finally send the result back to Alice, then each of them would know if there is a collision, but this reveals detailed information of Alice's path, and Bob may send a spurious result to Alice. However, this problem can be addressed by taking advantage of the Paillier homomorphic encryption system (PES). PES is an additive homomorphic encryption system [17], meaning that the sum of two encrypted numbers is the encrypted sum of the plain numbers. Paillier also supports scalar multiplication, which means that a scalar multiplied by an encrypted number yields the encryption of the scalar multiplied by the plain number. We emphasize that, though not a fully homomorphic encryption system, it suffices for the protocol we outline. To prevent the path information from being revealed, Alice can send encrypted information using PES, and Bob may perform the computation using these encrypted numbers only. However, in doing so, Bob might expose his information as stated in [5]. To prevent this, Bob adds some random numbers to the intermediate results (h_1 to h_4), then sends the outcomes to Alice. Then, Alice decrypts them and uses them as the inputs to the garbled circuit. Thus, Protocol 1 securely decides if two paths collide, avoiding leaking any path information.

Algorithm 1: ALICETRAJECTORY(P, k, IP_addr_B)

Input: $P = (S_1, S_2, \ldots, S_n)$; $k < n$ the number of segments per round; IP_addr_B Bob's IP address
Output: $C = (c_1, c_2, \ldots, c_m)$, $m = \lceil n/k \rceil$, $c_j \in \{False, True\}$ for $j = 1, 2, \ldots, m$, and $c_0 = True$ if both move for the first round.

```
CONNECT(IP_addr_B)
round ← 0      low ← 0      high ← low + k
k_A^pub, k_A^priv ← paillier()
while high ≤ n do
    R ← P[low : high]
    for i ← low to high − 1 do
        m ← E(k_A^pub, Equation(R_u, R_v))
        SEND(m)                    // encoded parameters
        RECEIVE(ACK)               // ACK awaiting
        p_A ← E(k_A^pub, R_{i,u} ⊔ 1)
        q_A ← E(k_A^pub, R_{i,v} ⊔ 1)
        SEND(p_A; q_A)
        RECEIVE(ACK)
        d_r ← RECEIVE()
        SEND(ACK)
        u ← D(k_A^pri, d_r)
        result_round ← ALICECIRCUIT(u)
        round ← round + 1
        low, high ← high, high + k
        if low < n and high > n then
            high, low ← n, n − k
return result
```

B. Privacy-Preserving Persistent Monitoring

Algorithms 1 and 2 show the implementation of the privacy-preserving persistent monitoring task for two parties Alice and Bob. A circuit that permits comparison of k segments at the time is used to compute whether a collision

among any combination of the $k \times k$ segments collides or not; for larger paths, k-length subsets are compared at a time, each referred to as a round. In this scheme, one robot moves k segments at each round. Both algorithms receive as input the list of segments, the number of segments per round, and the IP address to communicate with each other via network sockets. The algorithms can be run to plot the resulting behavior or to send commands to the robot platform.

Algorithm 2: BOBTRAJECTORY(P, k, IP_addr_A)

Input: $P = (S_1, S_2, ..., S_n)$; $k < n$ the number of segments per round; IP_addr_A Alice's IP address
Output: $C = (c_1, c_2, ..., c_m)$, $m = \lceil n/k \rceil$, $c_j \in \{False, True\}$ for $j = 1, 2, \ldots, m$, and $c_0 = True$ if both move for the first round

1 CONNECT(IP_addr_A)
2 $round \leftarrow 0$ $low \leftarrow 0$ $high \leftarrow low + k$
3 **while** $high \leq n$ **do**
4 $R \leftarrow P[low : high]$
5 **for** $i \leftarrow low$ ***to*** $high - 1$ **do**
6 $m \leftarrow$ RECEIVE() // encoded
7 SEND(ACK) // ACK receipt
8 $p_A; q_A \leftarrow receive()$
9 SEND(ACK)
10 r = ($rand()$,$rand()$,$rand()$,$rand()$)*(k*k)
11 $n \leftarrow Equation(R_u, R_v)$
12 $p_B \leftarrow R_u \sqcup 1$ $q_B \leftarrow R_j \sqcup 1$
/* $d_r = h, h_i = w_i + r_i, w_1 = (m \cdot p_B)$ */
13 $d_r \leftarrow [(m \cdot p_B), (m \cdot q_B), (n \cdot p_A), (n \cdot q_A)]$
14 $d_r \leftarrow d_r + r$
15 SEND(d_r)
16 RECEIVE(ACK)
17 $result_{round} \leftarrow$ BOBCIRCUIT(r)
18 $round \leftarrow round + 1$
19 $low, high \leftarrow high, high + k$
20 **if** $low < n$ ***and*** $high > n$ **then**
21 $high, low \leftarrow n, n - k$
22 **return** $result$

Algorithms 1 and 2 detail how the agents share the data they need. Algorithm 1 sends a variable to Bob (line 8). Bob receives this variable in Algorithm 2 (line 6). An "ACK" command is used to coordinate this data exchange since the function RECEIVE(ACK) will block the execution until "ACK" arrives. Once both algorithms have calculated and received their data, an instance of the circuit is launched. This circuit instance receives both Alice and Bob's input and sends the response to both of them. The circuit is launched via a system call, and the communication is achieved via sockets. If no collision is detected, Alice and Bob may safely move simultaneously. Otherwise, they will have to take turns.

C. Rendezvous Using Secure 3D Intersection

For the parties to find each other in the rendezvous problem, it is not enough to detect the intersection of the paths, but the meeting must occur at the same time. This is why the problem of secure rendezvous reduces to the calculation of the intersection of two time-parametrized paths with coordinates (x, y, t). In the 3D case, we need to use a fully homomorphic encryption system [18] since the homomorphic multiplication property is required between two encrypted numbers. The intersection of segments in the 3D case can be resolved using the following steps:

1) Determine whether two segments are coplanar.
2) If they are coplanar, solve the problem in a subspace.

Protocol 2 Secure Coplanar Protocol

Input: Given Alice's segment S_A and Bob's segment S_B.
Output: Whether S_A and S_B are coplanar.

1) Alice generates key pair (k_A^{pub}, k_A^{pri}) using fully homomorphic encryption system.
2) Alice computes $C_{\mathbf{u_A}}$ and $C_{\mathbf{v_A}}$, i.e., S_A encrypted components using k_A^{pub}.
3) Alice sends $C_{\mathbf{u_A}}$ and $C_{\mathbf{v_A}}$ to Bob.
4) Bob $\mathbf{p} = C_{\mathbf{v_A}} - C_{\mathbf{u_A}}, \mathbf{m} = \mathbf{u_B} - C_{\mathbf{u_A}}, \mathbf{n} = \mathbf{v_B} - C_{\mathbf{u_A}}$.
5) Bob computes $w = \mathbf{p} \cdot (\mathbf{m} \times \mathbf{n})$.
6) Bob computes $h = w + r_B$, r_B is a random number.
7) Bob sends h to Alice.
8) Alice computes t, decryption of h using k_A^{pri}.
9) Alice and Bob use a garbled circuit to check whether t is equal to r_B.
10) Return **True** if the circuit returns $true$, otherwise, return **False**.

Protocol 3 Secure 3D-Intersection Protocol

Input: Given Alice's path $P_A = (S_{A_1}, S_{A_2}, \ldots, S_{A_n})$, and Bob's path $P_B = (S_{B_1}, S_{B_2}, \ldots, S_{B_n})$.
Output: Whether P_A and P_B intersects.

1) Alice and Bob generates public / private key pairs (k_A^{pub}, k_A^{pri}) and (k_B^{pub}, k_B^{pri}) respectively using fully homomorphic encryption system.
2) For each pair of segments (S_A, S_B) where $S_A = (\mathbf{u_A}, \mathbf{v_A})$ from Alice and $S_B = (\mathbf{u_B}, \mathbf{v_B})$ from Bob.
   a) Both Alice and Bob execute Protocol 2 steps 2 to 7, Bob gets r_B and h.
   b) Bob computes $C_{\mathbf{u_B}} = (E(k_B^{pub}, \mathbf{u_B}.x), E(k_B^{pub}, \mathbf{u_B}.y), E(k_B^{pub}, \mathbf{u_B}.z))$, $C_{\mathbf{v_B}} = (E(k_B^{pub}, \mathbf{v_B}.x), E(k_B^{pub}, \mathbf{v_B}.y), E(k_B^{pub}, \mathbf{v_B}.z))$.
   c) Bob sends $C_{\mathbf{u_B}}, C_{\mathbf{v_B}}$ to Alice.
   d) Alice computes: $\mathbf{d_1} = DIR(C_{\mathbf{u_B}}, C_{\mathbf{v_B}}, \mathbf{u_A})$, $\mathbf{d_2} = DIR(C_{\mathbf{u_B}}, C_{\mathbf{v_B}}, \mathbf{v_A})$, $\mathbf{d_3} = DIR(\mathbf{u_A}, \mathbf{v_A}, C_{\mathbf{u_B}})$, $\mathbf{d_4} = DIR(\mathbf{u_A}, \mathbf{v_A}, C_{\mathbf{v_B}})$.
   e) Alice generates 2 random numbers r_{A_1}, r_{A_2} and computes $l_1 = (\mathbf{d_1} \cdot \mathbf{d_2}) + r_{A_1}, l_2 = (\mathbf{d_3} \cdot \mathbf{d_4}) + r_{A_2}$.
   f) Alice sends l_1, l_2 to Bob; Bob sends the output h from Protocol 2 to Alice.
   g) Alice computes $t = D(k_A^{pri}, h)$ and Bob computes $t_1 = D(k_B^{pri}, l_1), t_2 = D(k_B^{pri}, l_2)$
   h) Alice and Bob use a garbled circuit to check if the following expression is true: $((t == r_B) \wedge (r_{A_1} > l_1) \wedge (r_{A_2} > l_2))$ where t, r_{A_1}, r_{A_2} are inputs fed by Alice and r_B, l_1, l_2 are inputs fed by Bob.
   i) Return **True** if the garbled circuit returns $true$.
3) Return **False**.

Suppose we have four points P_1, P_2, P_3, P_4 in a 3D Cartesian space, then segment $\overline{P_1P_2}$ and $\overline{P_3P_4}$ are coplanar if and only if $\overrightarrow{P_1P_2}$ is perpendicular to $\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_4}$, that is, $\overrightarrow{P_1P_2} \cdot (\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_4}) = 0$.

Accordingly, we introduce a secure coplanar protocol in Protocol 2. A robust algorithm to determine whether two 2D segments intersect or not was given by [19]. The procedure works with 2D vectors. Since the plane (or a line if two segments are co-linear) spanned by two co-planar line segments

is a subspace of the 3D Cartesian space, the algorithm also works in this subspace. Along these lines, we extended the algorithm (in Protocol 3) to address secure computation in 3D. This protocol can be used to determine whether there is a collision between two 3D paths. Concretely, we have used it to detect rendezvous securely in two time-parameterized 2D trajectories.

The function $DIR(\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3},)$ used in Protocol 3 computes the normal vector to the plane formed by $\overrightarrow{u_3u_1}$ and $\overrightarrow{u_2u_1}$. It also must be pointed out that Protocol 3 does not consider the case when one vertex from a segment lies directly on the other segment. Additional comparisons are involved in order to handle this particular case.

V. Experimental Results

A. Software Implementation of Persistent Monitoring

The online version described in the problem formulation section was implemented in Python 3. The Fairplay software [20], [21] for Secure Multi-party Computation using garbled circuits was integrated into our implementation. Communication between the parties was achieved via sockets. We also used the Python library `python-paillier` [22] as it implements the scalar multiplicative and additive homomorphic cryptosystem Paillier [17]. Protocol 1 was implemented in a multi-round fashion, each round consisting of two k-length paths P_A and P_B. After each round, the robots decide whether to move simultaneously, if no collision is detected, or to determine who moves first otherwise. In the latter scenario, a collision was detected and the robots must move sequentially. A prior random agreement is used to settle which party has right of way. Fig. 2 shows two desired paths, the protocol rounds are highlighted accordingly for each party. It shows when a collision would only occur (though only if the paths collide in the same round). The green circles representing potential collisions never happen owing to the use of our strategy. The results are plotted in a simulation using Python.

Fig. 3 shows the execution of the paths shown in Fig. 2; the paths consist of 12 segments each. The implementation uses a subpath of length 4, i.e., $k = 4$, implying that 3 rounds must occur. In the first round, see Fig. 3(a), the paths in the round are collision-free and both robots can safely move simultaneously. Fig. 3(b) and Fig. 3(c) show round 2, wherein a collision is detected, and the robots decide to move one after the other. (In this simulation, Alice was randomly selected to go first.) Fig. 3(d) shows the last round. No collision is detected within this round so both Alice and Bob head to their destinations simultaneously.

B. Implementation of 3D Intersection

Since the secure intersection decision in 3D environments involves multiplication between encrypted numbers, a partial homomorphic cryptosystem like Paillier cannot implement Protocol 3. Hence, a fully homomorphic cryptosystem is needed instead. The Simple Encrypted Arithmetic Library

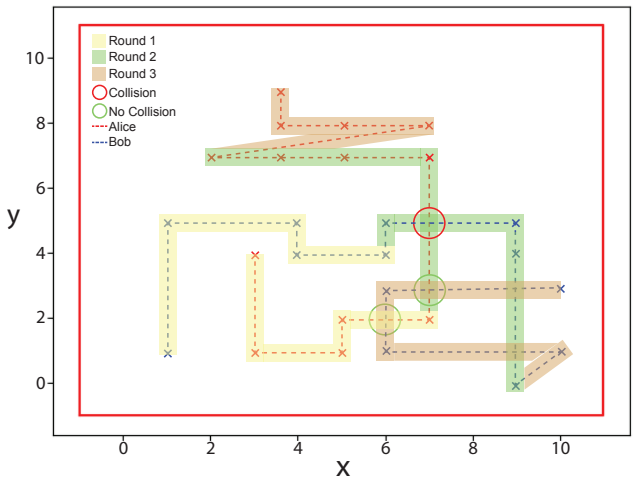

Fig. 2. Two desired paths of Alice and Bob in a multi-round simulation run. Three collisions (red and green circles) are found if two paths are compared as a whole. Only one collision (red circle) is found in our strategy as we divide the paths into different segments and compare these segments in several rounds.

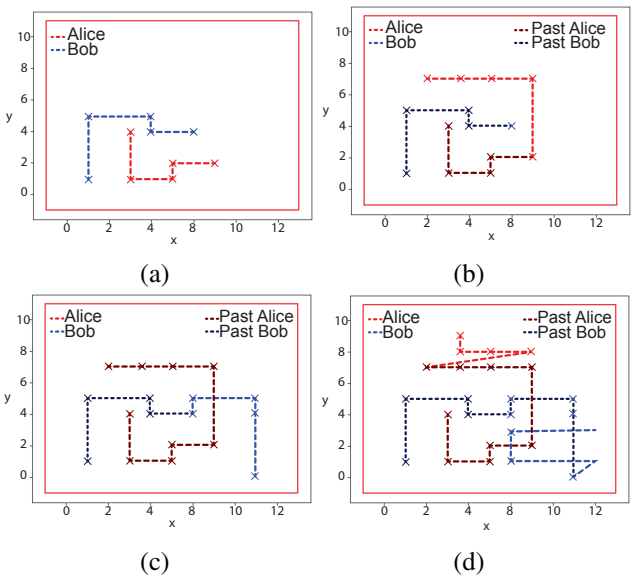

Fig. 3. Snapshots of the execution of paths shown in Fig. 2: (a) Alice and Bob move together since no collision exists in the first four segments; (b) A collision is detected in the next four segments, thus, Alice moves first; (c) Bob moves next; (d) Both Alice and Bob move simultaneously as no collision exists in the final four segments.

(SEAL) [23], [24] meets the necessary requirements. This library was developed by the Cryptography Research Group at Microsoft Research and has a Python wrapper PySEAL [25], [26] making it possible to use within Python.

Unlike the 2D case, where only Alice generates a public/private key pair, in the 3D case, both Alice and Bob generate key pairs. Thereafter, they exchange their encrypted points. We let Bob handle the vector computations related to the co-planar determination and have Alice handle the vector computations related to intersection calculation. This split is not essential for solving the problem: either Alice or Bob could perform all the calculations but doing so allows us to distribute the computational load.

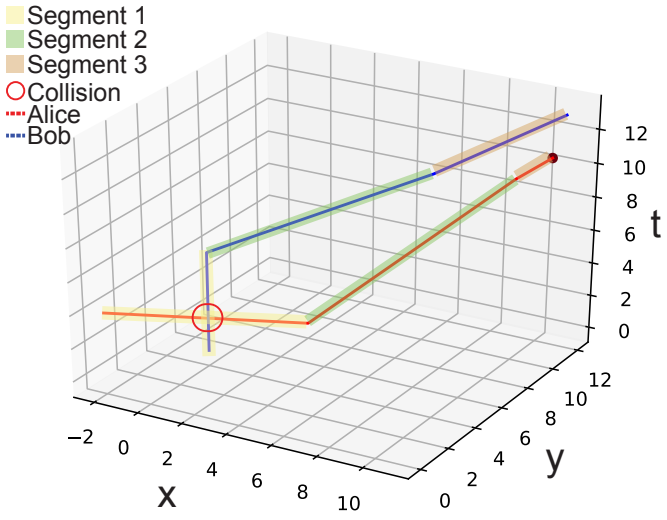

Fig. 4. Collision detection (red circle) in three segments of Alice's time-parameterized path (red) and Bob's time-parameterized path (blue).

Fig. 4 presents the simulation results for the algorithm that computes time-parameterized path collisions for objects moving in 3D space. This exemplifies our motivating example of shared areas where manned aircraft and small UAVs need to navigate without revealing their path information.

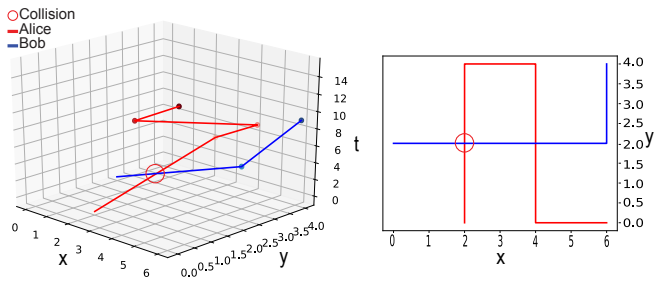

Fig. 5. Rendezvous determination using secure 3D intersection: (left) Only one point of intersection is found when the time is included as an extra dimension; (right) An additional false point of intersection is found in 2D that is resolved in time.

Fig. 5 shows a secure rendezvous experiment for Alice and Bob using 3D intersection. If only two dimensions (Fig. 5 right) are considered, then two rendezvous points might be found. However, when the time is considered, by the moment Bob arrives at the meeting point, Alice may have been there in the past, or she may only arrive in the future and so no real rendezvous will occur. A rendezvous (Fig. 5 left) only occurs when there is an intersection in $\mathbb{R}^2 \times \mathbb{T}$.

C. Hardware Experiment

A physical implementation was also conducted to test the persistent monitoring algorithms with two iRobot Create 2 platforms (see Fig. 1). The mobile robot platforms are connected to laptops running Ubuntu 12.0 SO, Intel Atom at 2.0 GHz, and 2 GB RAM. Two robots communicate with each other using the `pycreate2` Python library [27]. Fig. 1(a) shows the initial configurations and the intended path. In this experiment, all the segments are compared to each other. Since one collision is detected, Alice (blue line) moves first (Fig. 1(b) and 1(c)). Fig. 1(d) shows their final position. The interface to send commands to the robots takes care of the orientation of the robot, and the distance traveled at each segment. Finally, the robots face "EAST". More experiments and simulation videos ca benfound at: http://users.cis.fiu.edu/%7Ejabobadi/securemp/.

VI. Conclusion and Future Work

This paper demonstrated the feasibility of privacy-preserving multi-robot coordination. We believe that we have just scratched the surface and there are several practical avenues for future research.

In this work, privacy-preserving computational geometry primitives are used which *verify* properties such as the distance between points, line intersections [7], and point in polygons [5]. There also exists prior work in privacy computational geometry that *constructs* geometric objects. One such example is the privacy-preserving calculation of convex hulls [9], [28] which has an initial phase based on a data oblivious transfer algorithm that is then followed by secure protocols. We will explore this route in the future to extend the range of privacy-preserving robotic tasks that we can solve.

Hereby, a model where both agents need to cooperate in a shared environment but need to limit the disclosure of information in their coordination is considered. Fully adversarial motion planning where both Bob and Alice actively try to learn each other's information would be a further improvement of this work. We are currently exploring semi-honest models [29] where robots will follow the protocol but one robot is curious to learn the other robot's information.

A scheme to allow more than two parties is also a worthy aim, enabling multiple robots to decide how to move to prevent any collision. It presents several system challenges as to how many active connections they would manage and how to dynamically handle robots entering and leaving groups of interacting robots.

Finally, other mobile platforms, such as micro aerial vehicles, could be used for validation in future implementations. From a motion planning perspective, these ideas can be used as a final process in motion planning pipelines of autonomous vehicles [30], [31]. Although we did not consider kinematic constraints for this version, we plan to propose constructions of that work on more complex robots.

Acknowledgements

This work was supported in part by the National Science Foundation IIS-1453652. This work is also supported in part by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-062000002. This work is also supported in part by the National doctoral scholarship from COLCIENCIAS, Colombia.

REFERENCES

[1] Drone Airborne Collisions Report, 2017. Available at https://www.faa.gov/news/updates/newsId=89246&omniRss=news_updatesAoc&cid=101_N_U.

[2] A. C.-C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pp. 162–167, 1986.

[3] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, 1982.

[4] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pp. 218–229, 1987.

[5] M. J. Atallah and W. Du, "Secure multi-party computational geometry," in *Proceedings of the Workshop on Algorithms and Data Structures*, pp. 165–179, Springer, 2001.

[6] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proceedings of the Workshop on New Security Paradigms*, pp. 13–22, 2001.

[7] K. B. Frikken and M. J. Atallah, "Privacy preserving route planning," in *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pp. 8–15, 2004.

[8] K. Liang, B. Yang, D. He, and M. Zhou, "Privacy-preserving computational geometry problems on conic sections," *Journal of Computational Information Systems*, vol. 7, no. 6, pp. 1910–1923, 2011.

[9] S. Hans, S. C. Addepalli, A. Gupta, and K. Srinathan, "On privacy preserving convex hull," in *Proceedings of the International Conference on Availability, Reliability and Security*, pp. 187–192, 2009.

[10] L. Shundong, D. Yigi, W. Daoshun, and L. Ping, "A secure multi-party computation solution to intersection problems of sets and rectangles," *Progress in Natural Science*, vol. 16, no. 5, pp. 538–545, 2006.

[11] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, *et al.*, "Secure multiparty computation goes live," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 325–343, 2009.

[12] V. Renganathan and T. Summers, "Spoof resilient coordination for distributed multi-robot systems," in *Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems*, pp. 135–141, 2017.

[13] S. Gil, S. Kumar, M. Mazumder, D. Katabi, and D. Rus, "Guaranteeing spoof-resilient multi-robot networks," *Autonomous Robots*, vol. 41, no. 6, pp. 1383–1400, 2017.

[14] J. M. O'Kane and D. A. Shell, "Automatic design of discreet discrete filters," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 353–360, 2015.

[15] Y. Zhang and D. A. Shell, "Complete characterization of a class of privacy-preserving tracking problems," *International Journal of Robotics Research*, 2018.

[16] A. Prorok and V. Kumar, "A macroscopic privacy model for heterogeneous robot swarms," in *International Conference on Swarm Intelligence*, pp. 15–27, Springer, 2016.

[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238, Springer, 1999.

[18] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*, vol. 20. Stanford University Stanford, 2009.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[20] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: a system for secure multi-party computation," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, pp. 257–266, ACM, 2008.

[21] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella, *et al.*, "Fairplay-secure two-party computation system.," in *Proceedings of the USENIX Security Symposium*, vol. 4, p. 9, San Diego, CA, USA, 2004.

[22] Python Paillier Implementation. Available at https://python-paillier.readthedocs.io/en/develop/.

[23] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-seal v2. 1," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 3–18, Springer, 2017.

[24] Simple Encrypted Arithmetic Library (SEAL). Available at https://python-paillier.readthedocs.io/en/develop/.

[25] A. J. Titus, S. Kishore, T. Stavish, S. M. Rogers, and K. Ni, "PySEAL: A Python wrapper implementation of the SEAL homomorphic encryption library," *ArXiv e-prints*, Mar. 2018.

[26] PySEAL. Available at https://github.com/Lab41/PySEAL.

[27] PyCreate. Available at https://pypi.org/project/pycreate2/0.5.2/.

[28] D. Eppstein, M. T. Goodrich, and R. Tamassia, "Privacy-preserving data-oblivious geometric algorithms for geographic data," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 13–22, ACM, 2010.

[29] J. Brickell and V. Shmatikov, "Privacy-preserving graph algorithms in the semi-honest model," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 236–252, Springer, 2005.

[30] S. M. L. Valle, "Motion planning," *IEEE Robotics & Automation Magazine*, vol. 18, pp. 108–118, June 2011.

[31] S. M. LaValle, "Motion planning," *IEEE Robotics & Automation Magazine*, vol. 18, pp. 79–89, March 2011.