Privacy-Preserving Multi-Robot Task Allocation via Secure Multi-Party Computation

Murtadha Alsayegh, Peter Vanegas, Abdullah Al Redwan Newaz, Leonardo Bobadilla, Dylan A. Shell

***Abstract*— Multi-robot task allocation is a practical way to identify synergies between robots. When all the robots within a system fall under the auspices and authority of a single organization, they can simply be compelled to share their information and participate in cooperative protocols. But when, for instance, they are rivals vying in the marketplace, their own private data may be copyrighted or sensitive, so that disclosing information may erode a competitive advantage. Yet, even limited cooperation, by offering some arbitrage of common resources (such as shared infrastructure), often reduces costs for all parties; indeed, competition and cooperation are not mutually exclusive. We examine the question of how to allocate robots to tasks optimally while ensuring that no task valuations, utilities, positions, or related data are released. We do this via an auction-based assignment algorithm implemented using secure multi-party computation operations, without requiring any trusted auctioneer. The approach offers precise and effective privacy guarantees that are stronger than present methods. We demonstrate the feasibility of the approach via tests in a case study inspired by autonomous driving. First, we tested the approach in a single-computer setup, using parties with virtual network interfaces, where we studied the effects of varying the number of parties and the associated parameters of the auction. Next, we tested the approach in a decentralized, physical test-bed using single board computers running over a WiFi LAN network. Finally, we conducted a small proof-of-concept experiment using two autonomous mobile robots performing a decentralized, private auction.**

I. Introduction

Multi-robot task allocation addresses the problem of multi-robot coordination by decomposing the work that a set of robots seeks to accomplish into smaller, self-contained sub-elements called *tasks*, and then matching robots to those tasks. A variety of classical algorithms have been employed in robotics to solve the simplest case of this combinatorial problem [25], *viz.* for the single-robot tasks, single-task robots, instantaneous assignment (ST-SR-IA) problem. These include direct methods, ranging from heuristic techniques (such as picking greedy associations), to exact direct solutions (e.g., the Hungarian method, Linear Programming approaches). Such methods are usually implemented via a centralized algorithm on a single node that computes on aggregated data. When this happens, that node is privy to task evaluation information for each robot. Sharing this information may be undesirable.

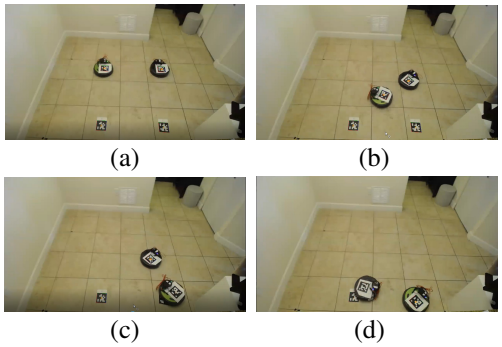


Fig. 1: A demonstration experiment: Subfigure (a) shows the initial position of the robots. At the end of the auction, each robot will be assigned a target. In subfigure (b) both robots start their motion toward their assigned targets. In (c) robot 2 detected an object and stops to avoid a collision using infrared sensors. Finally, in (d), both robots reached their target positions

In contrast, so-called market-based techniques are alternatives that are intrinsically more distributed. They employ auctions (e.g., [48], [24], [4]) or other economics-inspired means (e.g., [30]), and they may apply to ST-SR-IA instances, or more involved forms that allow additional constraints (e.g., [27], [12]). Though these approaches are more decentralized in general, they do usually involve some arbitrator (for auctions, termed an auctioneer). Less information is proffered to this party: bids are typically not placed on undesirable items/tasks, their values are only indirectly and imprecisely divulged. Even so, these are merely informal privacy properties and what *is* disclosed will vary depending on the items/tasks and robots participating. When information security is vital, new techniques with stronger privacy guarantees —such as the one we provide— are valuable.

The key innovation in this paper is that it shows how to use secure m-party computation (SMC) to enable distributed task-allocation without the robots revealing their preferences, nor the participating robots learning others' final assignments. We describe a probabilistic variant of the auction algorithm for assignment [21], [4], modified for the SMC setting. This algorithm, though implemented using cryptographic primitives, is practically feasible for real systems, illustrated via the experiments we conduct (including in a small-scale case study with basic robotic hardware).

Briefly, contributions of the paper can be summarized as: it presents a privacy preserving solution to the multi-robot task allocation problem, along with specialized privacy preserving constructions for the case of multi-vehicle routing. We provide an analysis of the algorithm (specifically: correctness and its probabilistic behavior, aspects of its

M. Alsayegh, P. Vanegas, and L. Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL 33199, USA

A. A. R. Newaz is with the Department of Electrical and Computer Engineering, North Carolina A&T State University, Greensboro, NC, USA

Dylan A. Shell is with the Department of Computer Science & Engineering at Texas A&M University, College Station, TX, USA

This work is supported in part by NSF: IIS-2034123, IIS-2024733, IIS-2034097, and by the U.S. Dept. of Homeland Security 2017-ST-062000002.

complexity, and the compositional-basis for its security). And then we describe our fully distributed implementation of the approach, report data from experiments, and a demonstration on physical hardware (see Figure 1). We also present a case study of autonomous vehicle ride-sharing for participants who require transportation between locations while preserving privacy.

II. Related Work

Multi-robot task allocation (MRTA) is a central problem in multi-robot teaming and distributed robot systems [38]; it involves the assignment of robots to tasks to maximize the collective performance of the system. Several reviews of this sub-area have been published [26], [28], [35], [25], speaking to both the applicability of the approach and the rapid evolution of the area. Part of the increment of the work deals with more complex, constrained, or specialized variants of the problem (e.g., [40], [45], [43]). One nascent line of work, highly relevant to the present study concerns robust or resilient task allocation (e.g., [32], [46], [34], [36]). Perhaps the closest work in spirit to the present paper is that of Prorok and Kumar [39], where an assignment problem is tackled where information leakage is explicitly considered as undesirable. That paper uses the differential privacy setting [20], where adding suitable noise will suffice, as distinct from the cryptographically secure model considered herein.

Our work is inspired by the use of Secure Multi-Party Computation (SMC) in auctioneering outside the robotics setting. Several types of auctions and benchmarking procedures have been proposed (see [14] for an overview and examples). Particularly related to our approach is Denmark's sugar beets auction problem [6], one of the first real-world deployments of SMC. In addition, [7] and [8] are prior works on maintaining fairness, privacy, and transparency in allocation and assignments.

Our approach employs the same techniques and tools as some of the recent applications of Machine Learning using Shamir Secret Sharing (SSS). Examples include Deep Neural Networks [1], Decision Trees [18], and Ridge Regression [5]. These approaches modify ML algorithms to use secure primitives that work on shared data; we provide similar modifications too. Secret Sharing has also been used in sensor processing where privacy is desired, such as fall detection [31]. Our work also connects with research that applies different SMC techniques to problems in estimation, control, robotics and sensor fusion [29], [2], [47].

III. Preliminaries and Problem Formulation

We have a group of m robots that are moving in a two-dimensional workspace $\mathcal{W} \subset \mathbb{R}^2$, containing an *obstacle region* $\mathcal{O} \subset \mathcal{W}$. The robots move in the *free space* of the environment, defined as $\mathcal{E} = \mathcal{W} \setminus \mathcal{O}$. The robots are modeled as points in $\mathcal{E}$, and each senses its own position via, say, GPS or other sensors (perhaps along state estimation techniques).

There will be m tasks that the robots need to accomplish, each of the robots has a *valuation* that encodes the preference for each task. We will denote by $v_{i,j}$ the preference that the i-th robot has for the j-th task. Let V be the *valuation matrix*, whose entries are $v_{i,j}$. The goal of our protocol will be to assign robots to tasks. Let A be the *assignment matrix* comprising entries $a_{i,j}$ with $a_{i,j} = 1$ if robot i is assigned task j and $a_{i,j} = 0$ otherwise.

Privacy-Preserving MRTA: *Given a group of m robots and m tasks, obtain an assignment A which maximizes the sum of the selected valuations, without revealing either the robots' valuations V or the tasks assigned to others.*

IV. Secure Multy-Party Computation Framework

We will succinctly describe only the basic elements of Secure Multi-party Computation used in this paper. For a complete treatment, see standard references [15], [22].

A. Shamir Secret Sharing

We employ MPC based on Shamir Secret Sharing (SSS) [42]. This procedure distributes $\mathcal{S}$, a secret, to a group with m parties, by creating m shares $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$ with the following properties: *i*) every subset of t parties can recover the secret and *ii*) no subset of $(t-1)$ or fewer can reconstruct the secret.

SSS relies on the fact that t points are sufficient to uniquely determine a polynomial on a finite field $GF(p)$ (with p a prime of appropriate size) of degree less or equal than $t-1$. We represent the secret $\mathcal{S}$ as an element $b_0 \in GF(p)$ and then choose randomly $t-1$ elements $b_1, b_2, \ldots, b_{t-1}$ from $GF(p)$ to build the polynomial $f(x) = b_0 + b_1 x + b_2 x^2, \ldots, b_{t-1} x^{t-1}$. After the polynomial is built, m points are obtained from this polynomial $(i, f(i))$ with $i \in \{1, \ldots, m\}$ and each given to the participant i. This procedure is shown in Protocol 1 [11].

Protocol 1 *ShamirShare*$(\mathcal{S}, m, t, p)$

Inputs: $\mathcal{S} \in GF(p)$ the secret, m the number of parties, t threshold for reconstruction, p cardinality $GF(p)$

Output: Shares $[s]_j$ for each of the m parties

1: Party P_i selects $b_1, b_2, \ldots, b_{t-1}$ from $GF(p)$
2: **foreach** $j \in \{1, 2, \ldots, m\}$
3: $\quad [s]_j \leftarrow \mathcal{S} + \sum_{k=1}^{t-1} b_k j^k$
4: $\quad$ Send $[s]_j$ to party P_j
5: **end foreach**

The secret $\mathcal{S}$ can be reconstructed if t or more parties combine their shares. Any subset of t parties can interpolate the polynomial to obtain $f(0) = \mathcal{S} = b_0$. This is achieved by the routine *OpenShare*$(A, [s])$ [11], where $[s]$ are the shares of the secret and A is a set of parties with $|A| > t$. *OpenShare* can be implemented using Lagrange interpolation [11].

B. Secure Operations on Secrets

One key aspect of SSS is that it allows each party to perform locally linear computations of secrets and public values. These linear combinations of secrets and public values include:

- **Addition of secrets** $[c] \leftarrow [a] + [b]$, in which each party P_i computes locally its share of the result $[c]_i \leftarrow [a]_i + [b]_i$.

- **Addition of a secret and a public value** $[c] \leftarrow [a]+\alpha$, $\alpha \in GF(p)$ in which each party P_i computes locally its share of the result $[c]_i \leftarrow [a]_i + \alpha$.
- **Multiplication of a secret and a public value** $[c] \leftarrow [a] \cdot \alpha$, where $\alpha \in GF(p)$ in which each party P_i computes locally its share of the result $[c]_i \leftarrow [a]_i \cdot \alpha$.

We will write the addition operation on secrets as *Sum*$([a], [b])$, where $[x]$ represents the shares of x for $x \in \{a, b\}$. Another important primitive in shares is *Mul*$([a], [b])$ that calculates the product of two shared numbers. This operation requires an interactive protocol to ensure that the resulting polynomial is uniformly random and has a degree t. (An implementation can be found in [11].)

Several other important operations can be built using these primitives. For example, *Inner*$([\mathbf{a}], [\mathbf{b}])$ which calculates the inner (dot) product between two shared vectors $\mathbf{a}$ and $\mathbf{b}$ (vectors are shared elementwise). *Inner* can be implemented naively via *Sum* and *Mul* or through a customized protocol.

Also, we will use other operations on shares built on these primitives such as *LEqualThan*$([a], [b])$, which takes two shared scalars $a, b \in GF(p)$ and returns 1 if $a > b$ and 0 otherwise, and *Min*$([\mathbf{a}])$ which returns a share of the smallest number in the shared vector $[\mathbf{a}]$. Details on these primitives can be found in [11] and implementations in the packages VIFF [23] and MPyC [41].

V. Methods

We will perform the $m \times m$ assignment through the AUCTION algorithm [4], [21], where one might consider the interpretation of each robot as a self-interested agent acting in a market. Each task j will have a *price*, denoted $price[j]$, and the robot needs to pay this price to get assigned this task. Therefore, the *utility* that each robot will get by getting assigned task j is $u_{i,j} = v_{i,j} - price[j]$. Each robot wants to obtain an assignment that maximizes its utility.

A. Privacy-Preserving Auction Algorithm

Protocol 2 gives the proposed procedure for carrying out task allocation via a secure auction, where bidding is managed via MPC primitives using shares. In the pseudocode, P_i is the party whose V_i valuation row is an input to the *Auction*, and *ShamirShare* is used to split the shares between all parties to maintain privacy. Boolean variable 'match' indicates if the given matrix $[A]$ leads to an assignment (a perfect matching), i.e., one that has none of the m parties claiming the same task. The vector variable 'prices' is shared among all robots, initialized with 0 values, and is monotonically increased over time to resolve ties. In lines 4–12, each P_i computes their preferences locally and securely enters the computation.

After each auction round, lines 13–23 determine whether there is a winner who will be assigned a task if $[A]$ is a match. Otherwise, every P_i will compute the δ, the difference between the two highest utilities $|u_i^{(1)} - u_i^{(2)}|$ and again share it securely between the parties using *ShamirShare*. This is so that all the parties will jointly compute the value [inc] which is the increment for to the 'prices' vector to break the tie existing between the parties. In the case that *LEqualThan*([inc], [1]), we add a small numerical value: $[\epsilon]$. The column totals are placed in $[C]$ for all $[A]$. In lines 23–24, we check if *GreaterThan*($[C_j]$, [1]) so that we increase the 'prices' vector by [inc] at the j-th position.

Protocol 2 *Auction*(V)

Inputs: $[V] = \{[V_1], [V_2], \ldots, [V_m]\}$ where each vector $[V_i] = \{[v_{i1}], [v_{i2}], \ldots, [v_{im}]\}$

Output: $[A] = \{[A_1], [A_2], \ldots, [A_m]\}$ where each vector $[A_i] = \{[a_{i1}], [a_{i2}], \ldots, [a_{im}]\}$

1: Initialize: prices $\leftarrow \{0, 0, \ldots, 0\}$
2: match $\leftarrow$ **False**
3: **while** $\neg$match **do**
4: **foreach** $i \in \{1, 2, \ldots, m\}$ **do in parallel**
5: P_i locally computes $u_i \leftarrow (v_i - \text{prices[i]})$
6: P_i locally computes $pos_i \leftarrow \arg\max(u_i)$
7: P_i locally sets $a_i \leftarrow \{0, 0, \ldots, 0\}$
8: P_i locally sets $a_{i,pos_i} \leftarrow 1$
9: **foreach** $j \in \{1, 2, \ldots, m\}$
10: P_i shares $a_{i,j}$ into $[a_{i,j}] \leftarrow$ *ShamirShare*$(a_{i,j})$
11: **foreach** $i \in \{1, 2, ..., m\}$ **do in parallel**
12: P_i computes match $\leftarrow$ *IsAMatch*$([A])$
13: **if** match **then**
14: return $[A]$
15: **else**
16: P_i computes $\delta_i \leftarrow |u_i^{(1)} - u_i^{(2)}|$
17: P_i shares δ_i into $[\delta_i] \leftarrow$ *ShamirShare*(δ_i)
18: P_i computes [inc] $\leftarrow$*Min*(δ)
19: **if** *LEqualThan*([inc], [1]) **then**
20: [inc] $\leftarrow [\epsilon]$
21: $[C] \leftarrow \{\sum_{i=0}^{m}[a_{i,1}], \sum_{i=0}^{m}[a_{i,2}], \ldots, \sum_{i=0}^{m}[a_{i,m}]\}$
22: **foreach** $j \in \{1, 2, ..., m\}$ **do**
23: **if** *GreaterThan*($[C_j]$, [1]) **then**
24: prices$[j] \leftarrow$ prices$[j]$ + [inc]
25: **end while**

Protocol 3 (see the next page) is designed to check if the given assignment matrix $[A]$ is a perfect match based on a probabilistic algorithm that is dependent on the calculated determinant of a matrix $[X]$. The determinant of $[A]$ being zero can be an indication that some task may still be unassigned; unfortunately, it can also result when there is redundancy (or additional choice). The solution we employ, inspired by the idea of probabilistic polynomial identity testing introduced in the Schwartz–Zippel lemma [19], is to pick random directions from the values that are assigned. On the basis of A, in Protocol 3, we construct a random matrix X: each zero element in A gives a corresponding zero in X, but each $a_{i,j} = 1$ results in a non-zero $x_{i,j}$, picked as a random integer between $\{1, \ldots, 2m\}$. (See Lines 2–7.) We then compute the determinant of the resulting X; if A has some unassigned task, so the matching is not perfect, then *Determinant*$(A) = 0$ and also *Determinant*$(X) = 0$. The *false positive* that *Determinant*$(A) = 0$ but the matching is perfect, will result in *Determinant*$(X) = 0$ only if the

Protocol 3 *IsAMatch*($[A]$)

Inputs: $[A] = \{[A_1], [A_2], \ldots [A_m]\}$ where each vector $[A_i] = \{[a_{i1}], [a_{i2}], \ldots, [a_{im}]\}$

Output: **True** if $[A]$ has a match, otherwise **False**.

```
1:  [X] ← [zero(X, len(A))]
3:      foreach t ∈ {1, 2, ..., k} do
4:          foreach i ∈ {1, 2, ..., m} do
5:              foreach j ∈ {1, 2, ..., m} do
6:                  if a_{i,j} = 1 then
7:                      [x_{i,j}] ← RandInt([1, ..., 2m])
8:                  else
9:                      [x_{i,j}] ← [0]
10:         [d] ← Determinant([X])
11:         if Equal([d], [0]) then
12:             return False
13:     return True
```

random selections also happens to be linearly dependent.

From [19], we see that the preceding gives a randomized protocol with error probability less than $\frac{1}{2}$. By running this protocol k times (Line 1), we can make the probability of error arbitrarily small. Our secure determinant was inspired by [5] where we use a secured determinant function based on secured primitives (line 8) to compute the determinant of matrix $[X]$. In lines 9–10, if $[d]$ is *Equal* to 0, then return false, and terminate the loop. Otherwise, iterate through the process again. Only after k successes, does it return true.

B. Complexity

We will proceed to calculate the complexity of the algorithm, first in the time required for each individual party, and then in communication and round complexity related to the interactive parts of the protocol.

1) Computational Complexity: The computational complexity of protocol 2 depends on the number of auction rounds r (While loop in Line 3, Protocol 2). Lines 5 and 8 are constant ($O(1)$), while lines 6 and 7 are $O(m)$, where m is the number of parties and tasks. The time required in line 15 is $O(k \cdot m^3)$ in the worst case, i.e., when Protocol 3, *IsAMatch*($[A]$), is called. Here, k is parameter of the randomized algorithm for checking the match, while $O(m^3)$ represents the computational complexity of calculating the determinant of an $m \times m$ matrix. Since k is a fixed small constant (in practice, 4 or 5 is suffices to get a high probability of match detection), we will write it as $O(m^3)$. The function call in line 15 dominates the computational complexity for the while-loop in each round of the protocol. Therefore, the worst-case complexity for Protocol 2 is $O(r \cdot m^3)$, where r is the number of auction rounds and m is the number of tasks.

2) Round Complexity: The other factors that affect the performance of the algorithm are the *round* and *communication* complexity. A round is a logical unit of the protocol when the parties must block to wait for messages from other parties to continue their computation [11]. To illustrate: unlike local addition of shares (for *Sum*($[a], [b]$)), the multiplication primitive *Mul*($[a], [b]$) requires one round of communication to be completed. The reader is referred to [11] for round complexity analysis of several SSS primitives.

The primitives in lines 10 and 19 in Protocol 2 take 1 communication round, while the primitives in lines 20, 21, and 26 take $4+\log(\ell+2)$ communication rounds [11], where ℓ is the length in bits of p in $GF(p)$ (e.g., if $p = 2^\ell - 1$). Since the while-loop in line 3 repeats per round, it would be $r \cdot (2+3(\ell+4)+\gamma)$, where γ is the number of rounds needed by line 14, which calls Protocol 3 (i.e., *IsAMatch*($[A]$)).

Protocol 3's number of rounds includes $k \cdot m^2$ calls to *RandInt* (each of which takes takes one round [11]), a call to *Determinant* (which can be implemented in a constant number of rounds d following the protocols presented in [5], [13]), and a call to *Equal* (which takes $2 + \log(\ell)$ rounds). Therefore, $\gamma = 2+\log(\ell)+k \cdot m^2+d$ and the total number of rounds for Protocol 2 is $r \cdot (2+3(\ell+4)+2+\log(\ell)+k \cdot m^2+d)$.

C. Correctness and Convergence

Our privacy-preserving algorithm is based on the AUCTION algorithm for optimal assignment presented in [4], [21], and as such it inherits its convergence and approximation properties as it will be discussed in the following subsections.

Proposition 1: Protocol 2 halts after a finite number of steps and finds a match with a probability of at least $1 - \left(\frac{1}{2}\right)^k$, where k is the parameter that randomized match checking procedure of Protocol 3.

Proof Sketch: The termination analysis is based on the ideas in [21]. Procedure of protocol 2 will stop when the vector 'prices' becomes *market-clearing*, so that each robot gets assigned a different task in the assignment matrix. It can be proven (see [21, Chapt. 10, and references therein]) that for any set of robot valuations, there exists a set of market-clearing prices. Unlike [21, chapt. 10], we do not solve the bipartite matching problem, but instead, owing to the privacy requirement of our approach, implement *IsAMatch*. It has at least $1 - (\frac{1}{2})^k$ probability of success in detecting matches.

D. Security Analysis

Proposition 2: Protocol 2 is secure under the semi-honest model when fewer than $\frac{1}{2}m$ parties are corrupted.

Proof Sketch: The analysis of the security of Protocol 3 based on Canetti's Universal Composition framework [10] that enables the modular analysis and design of complex cryptographic protocols from simple building blocks.

More specifically, Protocol 2 is built upon these 7 primitives: *ShamirShare*, *LEqualThan*, *Min*, *GreaterThan*, *RandInt*, *Equal*, and *Determinant*. Each of these primitives have been proven to be either perfectly secure or statistically secure under a passive adversary. See [11] (plus references therein) for the first 6 primitives; the comparable result for *Determinant* appears in [5], [13].

Therefore, if at some point during the execution of the protocol, the adversary has access to at most $t < \frac{m}{2}$ shares of the Shamir's (t, m)-threshold secret sharing scheme the

reconstructed secret will be a random element in the field $GF(p)$. Thus, the privacy property holds.

VI. Experimental Results

A. Case Study

Motivation for our ideas comes from the problem of ride-sharing vehicles wherein clients ought to be transported between places, but privacy between elements participating in the system is a valuable. Autonomous cars may wish to perform their tasks cooperatively whilst maintaining their privacy, working in a decentralized fashion so that multiple parties jointly perform a set of tasks without disclosing private information. One wishes that there is a fair approach to guarantee a satisfying assignment to all parties.

Our problem is formulated as follows: different autonomous robot taxis $R = \{r_0, \ldots, r_m\}$ are requested to transport a set of clients $C = \{c_0, \ldots, c_m\}$ between points $g = \{g_0, \ldots, g_m\}$ based on distance $D = \{d_0, \ldots, d_m\}$ guaranteeing that set of tasks $T = \{t_0, \ldots, t_m\}$ are conducted privately using our secure Protocol 2.

We are interested in testing our Protocol 2 for task allocation in $\mathbb{R}^2$ in a world using robots as a proof-of-concept. Each robot knows its location as well as the coordinates of the targets $C = \{c_0, \ldots, c_m\}$, through a visual fiducial system [37] (this will be discussed in more detailed in Section VI-D.

We calculate the distance between robot r_i to the targeted positions in $\mathbb{R}^2$ following the ideas in [17, Chapt. 5]. The Euclidean distance d_i between r_i and c_i is privately computed in $\mathbb{R}^2$ by finding the square root of the dot product where the distance between g_i and g_j determines r_i's preferences. We assume every r_i wishes to get the furthest distance d_i to maximize their return.

B. Single Computer Experiment

The single computer experiment was done using one computer which runs on an Ubuntu 18.04 LTS machine with an Intel i7 3.60GHz CPU and 16 GB RAM using Python 3. Each simulation experiment runs 30 times. Every time r_i receives a random coordinate from the world $\mathbb{R}^2$. Each party has a communication profile, which contains an IP and port number preserved for TCP/IP socket connections, and that is used to communicate with other parties.

Each party runs in a different virtual network interface that is associated with an MPyC asynchronous operation as discussed in detail here [16]. This means the auction runs with parties using a TCP/IP connection between each possible pair of parties, for a total of $\binom{m}{2}$ connections where m is the number of parties. The simulation results are constructed after the robots reach the optimal allocation where they run the distributed algorithm using the virtual interfaces.

Every $m - by - m$ simulation experiment is conducted 30 times with random positions of robots R and clients C. The destinations g is computed based on finding the Euclidean distance d as mentioned previously. Every simulation experiment consists of a different number of rounds where the number of rounds depends on the number of ties between parties preferences. Our results are promising and precise, but the number of parties bidding in the auction plays an important role in finding a fast match. For example, an auction between 3 parties will take less time than an auction between 10 as shown in Figures 2 and 3. We observe that most of the time spent is to break the ties between conflicting parties. For clarifications, each green triangle in the figures represents the average of round or time respectively. Some outliers shown as solid rounds appear in a few figures in particular experiments.

Our single computer experiment was conducted on a range of 3×3, 5×5, 7×7, and 10×10-sized auctions between simulated parties. Figures 2 and 3 show the number of rounds and time in ms required for each experiment depending on the $m \times m$ size of the array. Of note is that while time increases at a exponential rate (appropriate provided the nature of matrix operations), the number of comparative operations, *rounds*, increases almost linearly.

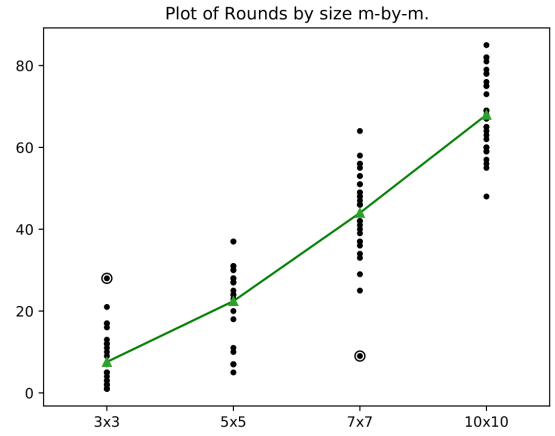


Fig. 2: Number of rounds for auctioning m different simulated target locations and m simulated parties on single computer.

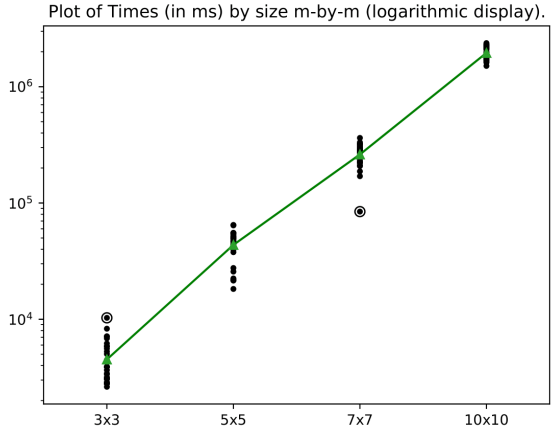


Fig. 3: Time (in ms) for auctioning m different simulated target locations and m simulated parties on single computer on a logarithmic scale.

Following this, in Figures 4 and 5, a series of tests were carried out on 5×5 arrays in which the increment value, ϵ, is increased. As a general trend, an increasing value of ϵ leads to a decreasing number of rounds and total time. However, this seems to reach a limit and converge, as values significantly over 1 (the value used as reference) saw continuously decreasing outputs.

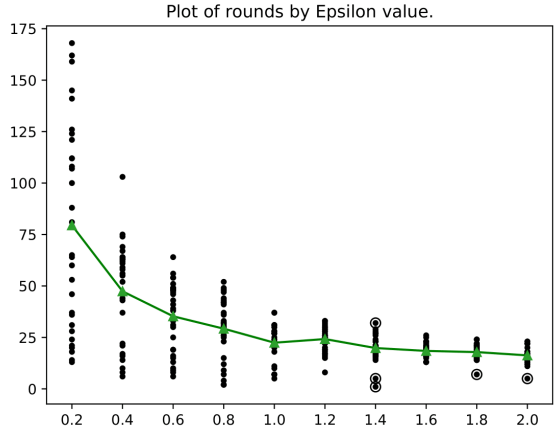


Fig. 4: Number of rounds by Epsilon value for auctioning 5 different simulated target locations and 5 simulated parties on single computer.

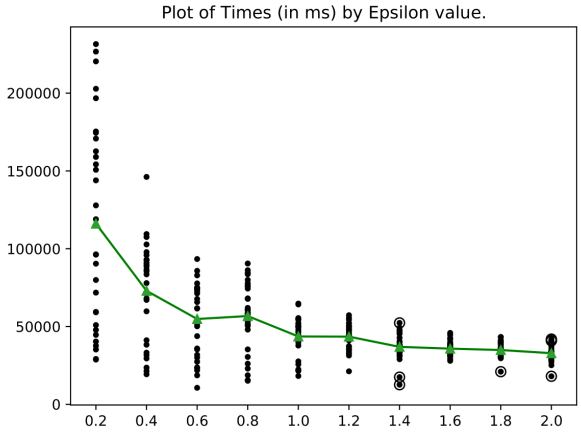


Fig. 5: Time (in ms) by Epsilon value for auctioning 5 different simulated target locations and 5 simulated parties on single computer.

We also vary the values for k to 3, 4, 5, and 6. There appears to be no direct correlation between a change in its value and a change in either time or rounds taken as shown in Figures 6 and 7.

C. Distributed Computer Experiments

We conducted a 3×3 experiment using 3 single board computers each with a 1.5GHz 64-bit quad-core CPU (4GB RAM) in a distributed fashion. Each single board computer runs a Raspbian OS, and communicates through WiFi LAN predefined with the MPyC package.

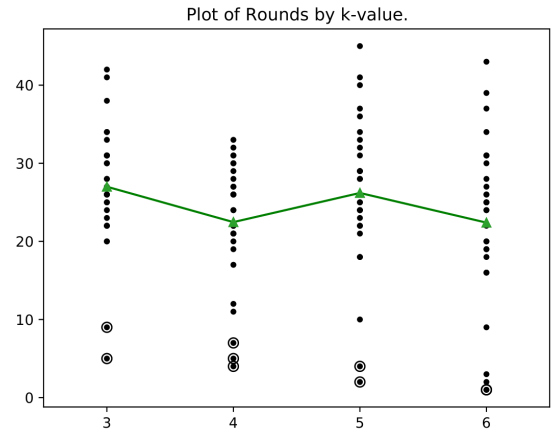


Fig. 6: Number of rounds by k value over 30 experiments on 5-by-5 assignment grids.

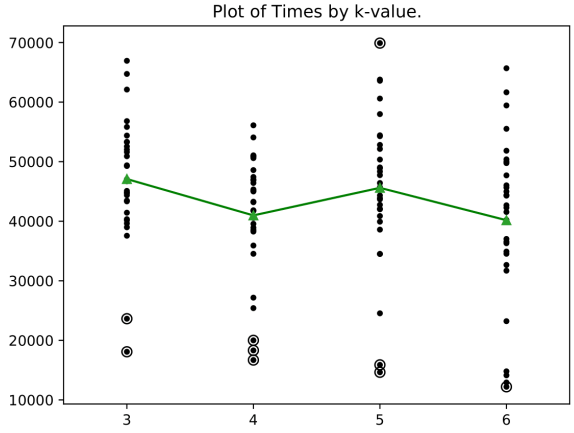


Fig. 7: Time (in ms) by k value over 30 experiments on 5-by-5 assignment grids.

Figure 8 shows the single board computers as well as three other dummy targets randomly placed on the map. Each party set their preferences by calculating the distances between their own position and positions of the targets and then, jointly participating in the auction. (See Figure 9.)

Fig. 8: Distributed Single Board Computer setup

The average time for the 30 experiments on 3 single board computers and 3 target locations in Figure 10 (e) was considerably longer than the experiment performed on the single computer because the specs of the single board com-

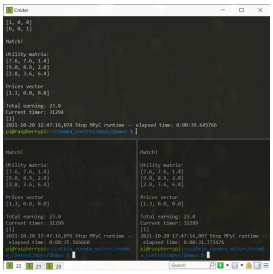
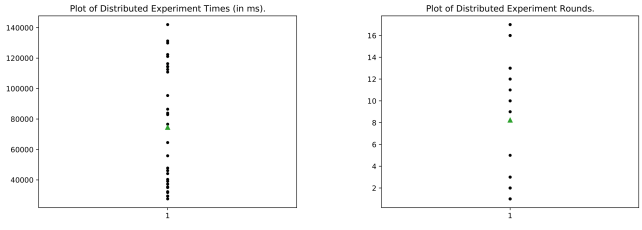


Fig. 9: Multi-Robot Task Assignment Using Privacy-Preserving Auction Algorithm.

puters are significantly different, and also the communication bandwidth between the physical devices is limited. However, the average number of rounds conducted for the experiment, Figure 10 (f) was similar to the single computer experiment.



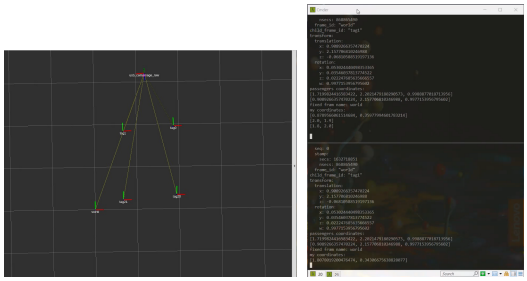
(e) (f)

Fig. 10: Subfigure (e) shows the Time (in ms) of 30 experiments for auctioning 3 different target locations from 3 distributed single-board computers. Subfigure (f) shows the number of rounds of 30 experiments for auctioning 3 different target locations from 3 distributed single board computers.

D. Mobile Robot Experiments

We have conducted physical experiments with two iRobot Create 2, each connected to a single board computer for controlling purposes. A video of this experiment can be found at https://youtu.be/a1B4jFGaZYM. For localization purposes, we use a visual fiducial system called *AprilTag* [37] to detect markers among other features in a natural scene. *AprilTag* is designed for a wide variety of tasks, including robot and camera calibration, where tags can be used as targets and could be detected using *AprilTag* software that computes the orientation and identity of the tag with respect to the camera. Our localization module, *AprilTag*, operates on a general purpose laptop. A unique tag is placed on top of each robot to identify and localize it with respect to the camera frame which later transforms to the map frame. Each robot receives its map coordinate over WiFi from the *AprilTag* server which is running on an off-board computer. All processes and communication between single board computers and the *AprilTag* server are established using the ROS packages [33], [9]. A demonstration of the world and robots outputs are shown in Figures 11 (g) and (h), respectively.

Each robot is equipped with a differential drive PID controller to navigate in the map while avoiding inter-robot collision using infrared sensors. First, robots set their task



(g) (h)

Fig. 11: (g) is the world as in RVIZ (ROS visualization) and (h) is the output screen of both robots.

preferences based on the proposed auctioning algorithm. Once task preferences are completed, then they generate point-to-point trajectories to move toward their target locations as shown in Figure 1. It required around 45 seconds to perform protocol 2 and finally assigned the two robots to their final tasks.

VII. Conclusions and Future Work

In this paper, we have proposed an approach based on Shamir Secret Sharing to allocate robots to tasks optimally while no information is released using an auction-based assignment algorithm. The algorithm was analyzed and tested in a single computer simulating network interfaces, a decentralized test-bed, and mobile robots. Several exciting research directions are open for future research.

In the applied direction, we believe that the analysis in simulated network interfaces, wireless deployments, and robot experiments supports the practical feasibility of the approach. We are currently working on two fronts. First, we want to conduct experiments with more mobile robots in a larger workspace and perhaps include other types of mobile robots. Second, we would like to explore other multi-robot task allocation study cases to extend the range of applications of our ideas. In addition, we will study adopting our algorithm to dynamic environments where prices and valuations change due to unforeseen obstacles. On the other hand, we will study when the robots have some kind of malicious behavior where they try to attack each other to reconstruct the secret and take advantage.

Given an assignment matrix, we use probabilistic polynomial identity derived from the Schwartz–Zippel lemma to test for the existence of a match. We are exploring replacing this probabilistic test with approaches in Linear Programming [44] or Maximum Flow [3].

When the number of tasks and robots differ, a standard approach is to pad the assignment matrix with dummy tasks or robots to ensure that the rows and columns match. The addition of dummy robots, however, demands additional computational elements; the addition of dummy tasks requires awareness of this fact by all the robots involved. We have not tackled the problem of mismatched task/robot numbers as this would likely leverage the mechanism used to inject tasks into the system, and those aspects have not been

the specific focus of study herein. An interesting question is to ask whether the addition of dummy elements leaks any important information.

REFERENCES

[1] M. Abspoel, N. J. Bouman, B. Schoenmakers, and N. de Vreede. Fast secure comparison for medium-sized integers and its application in binarized neural networks. Cryptology ePrint Archive, Report 2018/1236, 2018.

[2] A. B. Alexandru, A. Tsiamis, and G. J. Pappas. Towards private data-driven control. In *IEEE Conference on Decision and Control (CDC)*, pages 5449–5456, 2020.

[3] A. Aly. *Network flow problems with secure multiparty computation.* PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2015.

[4] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 20(4):133–149, 1990.

[5] F. Blom, N. J. Bouman, B. Schoenmakers, and N. d. Vreede. Efficient secure ridge regression from randomized gaussian elimination. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 301–316. Springer, 2021.

[6] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In R. Dingledine and P. Golle, editors, *Financial Cryptography and Data Security*, pages 325–343, 2009.

[7] P. Bogetoft, I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft. A practical implementation of secure auctions based on multiparty integer computation. In G. Di Crescenzo and A. Rubin, editors, *Financial Cryptography and Data Security*, pages 142–147, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] F. Brandt. Secure and private auctions without auctioneers. Technical report, Institut fur Informatik, Technische Universit at Munchen, 2002.

[9] C. Brommer, D. Malyuta, D. Hentzen, and R. Brockers. Long-duration autonomy for small rotorcraft uas including recharging. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7252–7258, 2018.

[10] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[11] P. Chen. Secure multiparty computation for privacy preserving data mining. Master's thesis, Eindhoven University of Technology, 2012.

[12] H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, 2009.

[13] R. Cramer and I. Damgård. Secure distributed linear algebra in a constant number of rounds. In *Annual International Cryptology Conference*, pages 119–136. Springer, 2001.

[14] R. Cramer, I. Damgård, and J. B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[15] R. Cramer, I. B. Damgård, et al. *Secure multiparty computation*. Cambridge University Press, 2015.

[16] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous multiparty computation: Theory and implementation. In *International workshop on public key cryptography*, pages 160–179. Springer, 2009.

[17] J. Dattorro. *Convex optimization & Euclidean distance geometry*. Meboo Publishing, USA (2011), 2010.

[18] S. de Hoogh, B. Schoenmakers, P. Chen, and H. op den Akker. Practical secure decision tree learning in a teletreatment application. In N. Christin and R. Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 179–194, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[19] M. L. Doğan, A. A. Ergür, J. D. Mundo, and E. Tsigaridas. The multivariate schwartz-zippel lemma, 2021.

[20] C. Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[21] D. Easley, J. Kleinberg, et al. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010.

[22] D. Evans, V. Kolesnikov, and M. Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2(2-3), 2017.

[23] M. Geisler. Cryptographic Protocols: Theory and Implementation. PhD thesis. `http://viff.dk/`, Feb 2010.

[24] B. Gerkey and M. Mataric. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[25] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[26] A. Khamis, A. Hussein, and A. Elmogy. Multi-robot task allocation: A review of the state-of-the-art. *Cooperative Robots and Sensor Networks 2015*, pages 31–51, 2015.

[27] S. Koenig, C. A. Tovey, X. Zheng, and I. Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *IJCAI*, pages 1359–1365, 2007.

[28] G. A. Korsah, A. Stentz, and M. B. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[29] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell. Coordinated multi-robot planning while preserving individual privacy. In *ICRA*, pages 2188–2194. IEEE, 2019.

[30] L. Liu and D. Shell. Optimal market-based multi-robot task allocation via strategic pricing. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[31] P. Mainali and C. Shepherd. Privacy-enhancing fall detection from remote sensor data using multi-party computation. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–10, 2019.

[32] M. Malencia, V. Kumar, G. Pappas, and A. Prorok. Fair robust assignment using redundancy. *IEEE Robotics and Automation Letters*, 6(2):4217–4224, 2021.

[33] D. Malyuta, C. Brommer, D. Hentzen, T. Stastny, R. Siegwart, and R. Brockers. Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition. *Journal of Field Robotics*, 37(1):137–157, 2020.

[34] S. Mayya, D. S. D'antonio, D. Saldaña, and V. Kumar. Resilient task allocation in heterogeneous multi-robot systems. *IEEE Robotics and Automation Letters*, 6(2):1327–1334, 2021.

[35] I. Mezei, V. Malbasa, and I. Stojmenovic. Task assignment in wireless sensor and robot networks. In *Telecommunications Forum (TELFOR)*, pages 596–602. IEEE, 2012.

[36] C. Nam and D. A. Shell. Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation. *IEEE Robotics and Automation Letters*, 2(1):193–200, 2016.

[37] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011.

[38] L. Parker, D. Rus, and G. Sukhatme. Multiple mobile robot systems. In *Handbook of Robotics*, chapter 53. Springer, second edition, 2016.

[39] A. Prorok and V. Kumar. Privacy-preserving vehicle assignment for mobility-on-demand systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1869–1876, 2017.

[40] H. Ravichandar, K. Shaw, and S. Chernova. Strata: unified framework for task assignments in large teams of heterogeneous agents. *Autonomous Agents and Multi-Agent Systems*, 34:1–25, 2020.

[41] B. Schoenmakers. MPyC: Secure multiparty computation in Python. `https://github.com/lschoe/mpyc`, May 2018.

[42] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[43] W. Smith and Y. Zhang. Achieving multitasking robots in multi-robot tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8948–8954, 2021.

[44] T. Toft. Solving linear programs using multiparty computation. In *International Conference on Financial Cryptography and Data Security*, pages 90–107. Springer, 2009.

[45] L. Vig and J. A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, 2006.

[46] F. Yang and N. Chakraborty. Algorithm for multi-robot chance-constrained generalized assignment problem with stochastic resource consumption. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4329–4336, 2020.

[47] Z. Zhang, J. Wu, D. Yau, P. Cheng, and J. Chen. Secure kalman filter state estimation by partially homomorphic encryption. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 345–346. IEEE Press, 2018.

[48] R. Zlot, A. Stentz, M. Dias, and S. Thayer. In *IEEE international conference on robotics and automation (ICRA)*, pages 3016–3023, 2002.