Secure Transformation for Multiparty Linear Programming

Yuan Hong and Jaideep Vaidya
MSIS department and CIMIC, Rutgers University, USA
{yhong, jsvaidya}@cimic.rutgers.edu

Abstract

With the rapid increase in computing, storage and networking resources, data is not only collected and stored, but also collaboratively analyzed. This creates a serious privacy problem which often inhibits the use of the distributed data. In this paper, we focus on the problem of linear programming, which is the most important sub-class of optimization problems. Specifically, distributed linear programming problems allow different companies (even competitors) to collaboratively seek the maximum profit or minimum cost by better utilizing the combination of their limited resources (constraints). However, the constraints and the objective are typically distributed across different companies. Since the constraints generally refer to internal limitations or capacities, and the objective specifies internal costs or value, serious privacy and security problems arise if these are completely disclosed to other companies. The main contribution of this paper is to introduce privacy-preserving distributed linear programming techniques to resolve the above issue for several real-world distributed linear programming problems. We present a transformation based approach that works for arbitrarily partitioned data distributed between multiple parties, and experimentally evaluate our solution.

1 Introduction

Today, data is ubiquitously collected and stored in many different forms by different organizations. To realize the benefit of this data, it is often collaboratively analyzed in many different ways. This creates a serious privacy problem which often inhibits the use of this data. In turn, this raises the question of whether it is possible to realize value from distributed data without conflicting with security and privacy concerns? The field of Secure Multiparty Computation addresses exactly this problem. The celebrated results[1, 2, 3] in this area show that any function can be securely computed in a distributed manner efficiently (i.e., in polynomial time w.r.t. the size of the circuit required to compute the function). However, this can lead to very inefficient solutions for complex functions or for large input sizes. More efficient solutions are necessary for important sub-problems.

In this paper, we look at the specific problem of how organizations optimize allocation of global resources while preserving the privacy of local information. Optimization is a fundamental problem found in many diverse fields. Research in optimization methods has generated many successes; the ubiquitous collection of data opens even greater opportunities. Much of this data is constrained by privacy and security concerns, preventing the sharing and centralization of data needed to apply optimization techniques.

Linear programming deals with a sub-class of optimization problems where all of the constraints and the objective function are *linear*. Linear Programming models are applicable to a wide variety of problems. Well known examples arise in many industries including transportation, commodities, airlines, communication, etc. There are also a variety of military applications and other economic applications.

A specific example can be seen in the packaged goods industry, where delivery trucks are empty 25% of the time. Just four years ago, Land O'Lakes truckers spent much of their time shuttling empty trucks down slow-moving highways, wasting several million dollars annually. By using a web based collaborative logistics service (Nistevo.com), to merge loads from different companies (even competitors) bound to the same destination, huge savings were realized (freight costs were cut by 15%, for an annual savings of $2 million[4]). Though this required sending all information to a central site, there was no alternative if savings were desired. However, the proposed solution *could* make this possible without the release of proprietary information. Another example is that of Walmart, Target and CostCo, who individually, ship millions of dollars worth of goods over the seas every month. These feed in to their local ground transportation network. The cost of sending half-empty ships is prohibitive, but the individual corporations have serious problems with disclosing freight information. If it were possible simply to determine what trucks should make their way to which ports to be loaded onto certain ships i.e., solve the classic transportation problem, without knowing the individual constraints, the savings would be enormous. In all of these cases, complete sharing of data would lead to invaluable savings/benefits. However, since unrestricted data sharing is a competitive impossibility or requires great trust, better solutions must be found.

Thus, distributed linear programming problems allow distributed parties (i.e. companies or organizations) to collaboratively seek the maximum profit or minimized cost by better utilizing the limited resources. Assuming that none of the parties is trusted by all other parties, and a trusted third party also does not exist, our goal is to solve the distributed LP problems with limited disclosure wherein none of the parties can learn anything about any other party's share of the LP problem (especially, the constraints, since they refer to the specific limitations or capacities of a company). In general, distributed linear programming problems typically have different forms based on different partitions of the constraints, and may thus have different privacy concerns resulting in different ways of solving them with limited disclosure. For example, Vaidya [5] considered the case where the objective function and constraints are partitioned between two parties with one party holding the objective while the other holding the constraints. In this paper, we first extend the above work by examining the arbitrarily partitioned case – the constraints and objective function are arbitrarily partitioned between two parties where one party holds a share of the constraints and objective function while the other party holds the remaining share. We then extend this to the arbitrary partitioning of data between multiple parties (where more than two parties share a part of the constraints and objective function). To solve the above distributed linear programming problems with limited disclosure, we propose a set of secure transformation based solutions that have the significant added benefit of being independent of the specific linear programming algorithm used.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces some preliminaries required by the rest of the paper, including the secure scalar product protocol used in our transformation approach, as well the different distributed LP models of interest. Section 4 illustrates the secure matrix transformation approach to linear programming problems and briefly reviews existing work on this. Section 5 presents the secure transformation for a more general scheme between two parties with arbitrary data partitioning. Section 6 extends this to the multi-party arbitrary data partitioning case. Section 7 presents experimental results to evaluate the efficiency of our approaches. Finally, Section 8 concludes the paper and discusses future work.

2 Related Work

Secure Multiparty Computation looks at the problem of securely computing a function over distributed inputs. The setting of SMC encompasses tasks as simple as coin-tossing and broadcast, and as complex as

electronic voting, electronic auctions, electronic cash schemes, contract signing, anonymous transactions, and private information retrieval schemes. The key idea behind Secure Multi-party Computation is that a computation is secure if at the end of the computation, no party learns anything except its own input and the results (and anything that can be inferred from these two pieces). The gold standard is that of a trusted third party which performs the entire computation. Thus, the key is to achieve the same results without having a trusted third party.

Secure computation has a very rich history. Yao first postulated the two-party comparison problem (Yao's Millionaire Protocol) and developed a provably secure solution[1]. Goldreich et al.[2] generalized this to multi-party computation and proved that there exists a secure solution for any functionality. The approach used is as follows: the function F to be computed is first represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Every participant gets random shares of the input and output wires for every gate. This approach, though appealing in its generality and simplicity, means that the number of rounds of the protocol grow with the size of the circuit. This grows with the size of the input. This is highly inefficient for large inputs or complicated circuits, as in optimization. Although this proves secure solutions exist, achieving efficient secure solutions for distributed optimization is still open.

There has been significant theoretical work in this area. Both [1] and [2] assumed polynomially time bounded passive adversaries. In a line of work initiated by Ben-Or et al.[3], the computational restrictions on the adversary were removed, but users were assumed to be able to communicate in pairs in perfect secrecy. Ben-Or et al. [3] assume passive adversaries, while Chaum et al. [3] extend this to active adversaries. Ostrovsky and Yung [3] introduce the notion of mobile adversaries, where the corrupt users may change from round to round. Finally, the coercing adversary who can force users to choose their inputs in a way he favors was introduced in the context of electronic elections by Benaloh and Tunistra [6], and generalized to arbitrary multi-party computation by Canetti and Gennaro [7]. Much effort has been devoted to developing crisp definitions of security [8] [9]. However, due to efficiency reasons, it is completely infeasible to directly apply the theoretical work from SMC to form secure protocols for optimization.

There is work in distributed optimization that aims to achieve a global objective using only local information. This falls in the general area of distributed decision making with incomplete information. This line of research that has been investigated in a worst case setting (with no communication between the distributed agents) by Papadimitriou et al. [10] [11] [12]. In [12], Papadimitriou and Yannakakis first explore the problem facing a set of decision-makers who must select values for the variables of a linear program, when only parts of the matrix are available to them and prove lower bounds on the optimality of distributed algorithms having no communication. Distributed Constraint Satisfaction was formalized by Yokoo [13] to solve naturally distributed constraint satisfaction problems. These problems are divided between agents, who then have to communicate among themselves to solve them. To address distributed optimization, complete algorithms like OptAPO and ADOPT have been recently introduced. ADOPT [14] is a backtracking based bound propagation mechanism. It operates completely decentralized, and asynchronously. The downside is that it may require a very large number of messages, thus producing big communication overheads. OptAPO [15] centralizes parts of the problem; it is unknown apriori how much needs to be centralized where, and privacy is an issue. Distributed local search methods like DSA([16]) / DBA([17]) for optimization, and DBA for satisfaction ([18]) start with a random assignment, and then gradually improve it. Sometimes they produce good results with a small effort. However, they offer no guarantees on the quality of the solution, which can be arbitrarily far from the optimum. Termination is only clear for satisfaction problems, and only if a solution was found.

DPOP [19] is a dynamic programming based algorithm that generates a linear number of messages. However, in case the problems have high induced width, the messages generated in the high-width areas

of the problem become too large. There have been proposed a number of variations of this algorithm that address this problem and other issues, offering various tradeoffs (see [20] [21] [22] [23] [24]). [20] proposes an approximate version of this algorithm, which allows the desired tradeoff between solution quality and computational complexity. This makes it suitable for very large, distributed problems, where the propagations may take a long time to complete.

However, in general, the work in distributed optimization has concentrated on reducing communication costs and has paid little or no attention to security constraints. Thus, some of the summaries may reveal significant information. In particular, the rigor of security proofs has not been applied much in this area. There is some work in secure optimization. Silaghi and Rajeshirke [25] show that a secure combinatorial problem solver must necessarily pick the result randomly among optimal solutions to be really secure. Silaghi and Mitra [26] propose arithmetic circuits for solving constraint optimization problems that are exponential in the number of variables for any constraint graph. A significantly more efficient optimization protocol specialized on generalized Vickrey auctions and based on dynamic programming is proposed by Suzuki and Yokoo [27], though it is not completely secure under the framework in [25]. Yokoo et al. [28] also propose a scheme using public key encryption for secure distributed constraint satisfaction. Silaghi et al. [29] show how to construct an arithmetic circuit with the complexity properties of DFS-based variable elimination, and that finds a random optimal solution for any constraint optimization problem. Atallah et al. [30] propose protocols for secure supply chain management. However, much of this work is still based on generic solutions and not quite ready for practical use. Even so, some of this work can definitely be leveraged to advance the state of the art by building general transformations or privacy-preserving variants of well known methods. Li and Atallah[31] have proposed a more efficient solution for this problem based on adapting the simplex method to make it secure. As such, the computation and communication complexity of their solution is proportional to that of the simplex method, which could be exponential in the worst case. Also their solution uses expensive subprotocols which make it quite computationally challenging for large problems. Our solution is much more efficient and practical. Mangasarian [**?**]proposed a privacy-preserving formulation of a linear program over vertically partitioned constraint matrix while our approach is introduced to privately solve arbitrarily partitioned LP problems in this paper, and no formal security analysis is given in [**?**]. Du [32] has also studied this problem, and proposed a similar transformation based solution. Indeed, our work is inspired by his work – however, his solution does not work correctly in all cases, and little analysis of security and efficiency is provided. Our solution looks at a specific data partitioning scenario, is more robust, and is analyzed in significantly more depth.

3 Preliminaries

In this section, we first present the basic LP problem model, followed by the distributed LP model along with the ways in which data is typically distributed. We then review the secure scalar product protocol proposed by Goethals et al.[33] which is used in all of our solution protocols.

3.1 LP Model

Optimization is the study of problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. Formally, given a function $f : A \longrightarrow R$ from some set A to the real numbers, we seek an element x_0 in A such that $f(x_0) \leq f(x), \forall x \in A$ ("minimization") or such that $f(x_0) \geq f(x), \forall x \in A$ ("maximization"). Thus, an optimization problem has three basic ingredients:

- An *objective function* which we want to minimize or maximize.
- A set of *unknowns* or *variables* which affect the value of the objective function.
- A set of *constraints* that allow the unknowns to take on certain values or exclude others.

In linear programming, the objective function f is linear and the set A is specified using only linear equalities and inequalities. Thus, for linear programming, the problem can be easily restated as:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t. } Mx \quad & \geq \quad b \\ x \quad & \geq \quad 0 \end{aligned}$$

Software for linear programming (including network linear programming) consumes more computer cycles than software for all other kinds of optimization problems combined. LP is a very rich area of research with numerous algorithms known. Solution approaches from LP have also benefited the overall field of optimization.

3.2 Distributed LP Model

Optimization problems (especially issues in linear programming) have been well studied in the literature – methods have been proposed for the case when all of the data is available at a central site. Methods have also been proposed for the case with incomplete information (distributed optimization). However all of these solutions assume that all the necessary data is centralized or freely available. On the contrary, whenever the data is distributed, we have to be more concerned about the problems caused by privacy and security. Different parties might own different constraints or even different parts of the same constraint. Thus, if there are several parties solving a distributed LP problem, the first issue should be to examine the data (constraints or objective) partition.

3.2.1 Data Partition

There are many ways in which data could be distributed. Each of the ingredients of the optimization problem could be distributed. For example, the objective function might be known to only one party, or parts of it known to some subsets of the parties. The constraints that define the set A might also be distributed in some fashion. Thus, the different ways in which data is distributed give rise to the following categorization.

Horizontal Partitioning / Homogeneous Distribution: Here, each constraint would be fully owned by one party. Thus, different parties own different constraints. An example of this would be the distributed scheduling problem. Suppose that several schedulers need to schedule tasks on machines. Each task can be executed by several machines (though not all), and it can be split between several machines, but the fraction of all tasks executed by a machine must under no circumstances exceed its capacity. Each scheduler only knows the tasks that may be executed on its pertinent machines, and based on this information it must decide what fractions of its task to send to which machines. The sum of all fractions is to be maximized. Here, the objective function could be known to a single party or to all of the parties, or even be shared by the parties.

Vertical Partitioning / Heterogeneous Distribution: In this case, each constraint is shared between some subset of the parties. An example of this would be the organization theory problem. A large enterprise has a very extensive set of tasks – say, products manufactured. A fundamental question in Organization

Theory is, *how are these tasks to be partitioned among managers?* Although the profitability and resource requirements of these products may change dynamically with market conditions, the *constraint structure*, the sparsity pattern of the constraint matrix of the associated linear program, may be fixed. That is, it is known in advance, which products compete for which resources. What are the *organizational principles* that should guide this assignment of tasks to managers, so that the latter can make more informed decisions. Again, the objective function might be known to all of the parties, or just to a single party, or be shared by the parties.

Arbitrary Partitioning: Apart from the prior two partitioning methods, the data may also be arbitrarily partitioned in some way (some combination of the above). This is more general and subsumes both of the earlier cases. Completely arbitrary partitioning of data is unlikely in practice, though certain specific configurations might easily be found. In any case, solutions for this case will always work for both of the prior cases as well. Section 5 and 6 propose the extended secure solutions for arbitrarily partitioned objective function and constraints with respect to universal cases.

3.2.2 Distributed LP Problem Scenarios

Based on the above categorization of data partitioning, we now examine how data is actually partitioned in some real-world distributed LP problems, and discuss some of the practical issues.

In [5], Vaidya proposed the privacy-preserving distributed linear programming problem as a special case of arbitrary partitioning between two parties, where one party owns the objective function while the other party owns the constraints. This may happen, i.e. in Scenario 1.

Scenario 1: *A manufacturer would like to evaluate several transportation options between its factories, raw material suppliers and while different transporters would have different costs (leading to different objective functions). They need to figure out who is the lowest cost provider but without disclosing the constraints to the other party.*

In general, the constraints (or objective function) may be arbitrarily partitioned in distributed LP problems. In this paper, we propose secure solutions for two or more parties who hold arbitrarily partitioned data by ***assuming that the global constraints are the union of all partitioned data: 1. the shared constraints are the sum of all parties' partitioned data, 2. some constraints are owned by only one party***. This happens very often, i.e. in Scenario 2.

Scenario 2: *Some corporations (i.e. Walmart, CostCo) usually import goods from various external factories at different locations. Numerous local stores of Walmart, Costco .etc can collaboratively ship goods from the same factories to the same destinations for reducing their transportation costs. This is a typical distributed LP problem (arbitrarily partitioned transportation problem). Specifically, let x_{ijk} represent the delivery amount of corporation i from location j (factories) to location k (destinations), the goal is to minimize the global shipping cost while subject to the constraints in the transportation problem. The delivery of some routes (from location j to location k) might have the goods for different corporations, thus the corresponding constraints are shared by them (Vertically partitioned constraints). By contrast, some corporations may have their own constraints in transportation or production (Horizontally partitioned constraints). Therefore, this scenario is an arbitrary partition case.*

In Scenario 2, assuming three parties (P_1, P_2 and P_3), we let the share of the arbitrarily partitioned constraint matrix be M_1, M_2 and M_3: the share of vertically partitioned constraint matrix are A_1, A_2 and A_3, the share of horizontally partitioned constraints are B_1, B_2 and B_3, as shown in Equation 1.

$$M_1 = \begin{bmatrix} A_1 & 0 & 0 \\ B_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, M_2 = \begin{bmatrix} 0 & A_2 & 0 \\ 0 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, M_3 = \begin{bmatrix} 0 & 0 & A_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & B_3 \end{bmatrix}, M = \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{bmatrix} \quad (1)$$

Thus, we can consider $M = M_1 + M_2 + M_3$ as shown below:

$$\begin{array}{c} P_1, P_2, P_3: \\ P_1: \\ P_2: \\ P_3: \end{array} \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{bmatrix} = P_1: \begin{bmatrix} A_1 & 0 & 0 \\ B_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + P_2: \begin{bmatrix} 0 & A_2 & 0 \\ 0 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} + P_3: \begin{bmatrix} 0 & 0 & A_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & B_3 \end{bmatrix} \quad (2)$$

In the constraints, we do not discuss the partition on b (constraints $Mx \leq b$ or $Mx \geq b$) in this paper (by assuming that the shares of b can be known to all parties if it is partitioned among all parties as b'_1, b'_2 and b'_3).

$$\begin{array}{c} P_1, P_2, P_3: \\ P_1: \\ P_2: \\ P_3: \end{array} \begin{bmatrix} b \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}, P_1: \begin{bmatrix} b \text{ or } b'_1 \\ b_1 \\ 0 \\ 0 \end{bmatrix}, P_2: \begin{bmatrix} b \text{ or } b'_2 \\ 0 \\ b_2 \\ 0 \end{bmatrix}, P_3: \begin{bmatrix} b \text{ or } b'_3 \\ 0 \\ 0 \\ b_3 \end{bmatrix} \quad (3)$$

Moreover, the objective function is also partitioned between multiple parties and we have $C = C_1 + C_2 + C_3$ as shown in Equation 1.

$$P_1, P_2, P_3: \begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix} = P_1: \begin{bmatrix} C_1 & 0 & 0 \end{bmatrix} + P_2: \begin{bmatrix} 0 & C_2 & 0 \end{bmatrix} + P_3: \begin{bmatrix} 0 & 0 & C_3 \end{bmatrix} \quad (4)$$

In this arbitrarily partitioned LP problem, each party does not want to reveal anything to other parties. We will present privacy-preserving techniques for solving this general and untrusted collaborative LP problem in this paper.

3.3 Secure Scalar Product Protocol

Our solutions use the secure scalar product protocol as a primitive. We now briefly review how this primitive can be implemented. Goethals et al. [33] propose a simple and provably secure method to compute the scalar product using Homomorphic Encryption. The problem is defined as follows: P_1 has a n-dimensional vector $\vec{X}$ while P_2 has a n-dimensional vector $\vec{Y}$. At the end of the protocol, P_1 should get $r_a = \vec{X} \cdot \vec{Y} - r_b$ where r_b is a random number chosen from an uniform distribution and is known only to P_2. The key idea behind the protocol is to use a homomorphic encryption system such as the Goldwasser-Micali cryptosystem [34], the Benaloh cryptosystem [35], the Naccache-Stern cryptosystem[36], the Paillier cryptosystem [37], and the Okamoto-Uchiyama cryptosystem [38]. Homomorphic Encryption is a semantically-secure public-key encryption which, in addition to standard guarantees on encryption, has the additional property that given any two encryptions $E(A)$ and $E(B)$, there exists operations $*, \cdot$ such that $E(A * B) = E(A) \cdot E(B)$, where $*$ is either addition or multiplication (in some abelian group). The cryptosystems mentioned above are additively homomorphic (thus the operation $*$ denotes addition, and $\cdot$ denotes multiplication). Using

such a system, it is quite simple to create a scalar product protocol. The key is to note that $\sum_{i=1}^{n} x_i \cdot y_i = \sum_{i=1}^{n}(x_i + x_i + \cdots + x_i)$ (y_i times). If P_1 encrypts her vector and sends in encrypted form to P_2, P_2 can use the additive homomorphic property to compute the dot product. The specific details are given below:

Protocol 1 Homomorphic Encryption

Require: P_1 has input vector $\vec{X} = \{x_1, \ldots, x_n\}$
Require: P_2 has input vector $\vec{Y} = \{y_1, \ldots, y_n\}$
Require: P_1 and P_2 get outputs r_A, r_B respectively such that $r_A + r_B = \vec{X} \cdot \vec{Y}$

1: P_1 generates a private and public key pair (sk, pk).
2: P_1 sends pk to P_2.
3: **for** $i = 1 \ldots n$ **do**
4: P_1 generates a random new string r_i.
5: P_1 sends to P_2 $c_i = Enc_{pk}(x_i; r_i)$.
6: P_2 computes $w = \prod_{i=1}^{n} c_i^{y_i}$
7: P_2 generates a random plaintext r_B and a random nonce r.
8: P_2 sends to P_1 $w' = w \cdot Enc_{pk}(-r_B; r')$.
9: P_1 computes $r_A = Dec_{sk}(w') = \vec{X} \cdot \vec{Y} - r_B$.

4 A Transformation Approach for LP Problems [5]

In this section, we present a constraint matrix transformation approach for distributed LP problems and discuss the potential privacy risks in transformation.

4.1 Linear Programming Problem Transformation

The effectiveness of any solution can be measured on the basis of three critical properties – privacy/security, utility, and efficiency. Does the solution preserve the privacy/security of the data? How useful is the solution (i.e., how accurate are the analysis results)? How efficient is the solution? Any solution that completely disregards even one of these properties is not of any use, while the ideal solution would perfectly satisfy all three parameters. However, most current solutions end up meeting one or two of the objectives, while disregarding the third. We now present a transformation based solution to the problem.

The key idea in transformaion is to transform the vector space by applying a linear transformation. This idea was first proposed by Du to solve systems of linear equations[39]. Du later proposed extending this idea to solve the two party linear programming problem[32]. However their solution sketch gives incorrect results in many cases (a non-optimal solution is incorrectly reported as optimal). We now show how a correct solution can be created. As noted before, assume that you want to solve the problem:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t. } Mx \quad & \leq \quad b \\ x \quad & \geq \quad 0 \end{aligned}$$

The key to the solution is based on the fact that $MQQ^{-1}x \leq b$, and $Q^{-1}x \geq 0$ if $Mx \leq b$ and $x \geq 0$ (the elements of Q^{-1} should all be positive and Q can be selected according to Bednarz et al.'s work [40]).

Let $M' = MQ$, $y = Q^{-1}x$, and $c'^T = c^T Q$. We now have a new linear programming problem:

$$\begin{aligned} \max \quad & c'^T y \\ \text{s.t. } M'y \quad & \leq \quad b \\ y \quad & \geq \quad 0 \end{aligned}$$

Recall that Du [39][32] and Vaidya [5] proposed a transforming approach for solving two-party distributed LP problems: transforming an $m \times n$ constraint matrix M (the objective vector c^T) to another $m \times n$ matrix $M' = MQ$ ($c'^T = c^T Q$) by post-multiplying an $n \times n$ matrix Q, solving the transformed problem and deriving the original solution. This LP problem transformation approach is based on the above work.

Specifically, following the methodology of Du [32], we can prove that if $y*$ is the solution to the new problem, then $x* = Qy*$ must be the solution to the original problem that minimizes $c^T x$. The proof is based on contradiction as follows: Suppose $x* = Qy*$ is not the optimal solution for the original problem. In this case, there must be another vector x' such that $c^T x' > c^T x*$ where $Mx' \leq b$ and $x' \geq 0$. Let $y' = Q^{-1}x'$. Now, if we evaluate the objective function of the new problem for this new solution, we find $c'^T y' = c^T Q Q^{-1} x' = c^T x'$. Now,

$$\begin{aligned} c^T x' > c^T x* \quad & \Longrightarrow \quad c'^T y' > c^T x* \\ & \Longrightarrow \quad c'^T y' > c^T Q Q^{-1} x* \\ & \Longrightarrow \quad c'^T y' > c'^T y* \text{ (since } c'^T = c^T Q \text{ and x*=Qy*)} \end{aligned}$$

However, this would imply that y' is a better solution to the new problem than $y*$ which leads to a contradiction (since $y*$ is supposed to be the optimal solution). Thus, it is clear that transformation will give us the correct solution, and we can take use of it to secure the transformation and preserve each party's privacy in the collaboration. Similarly, we can use the optimal solution of the transformed problem ($y*$) to generate the optimal solution $x* = Qy*$ for the minimization problem below:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t. } Mx \quad & \geq \quad b \\ x \quad & \geq \quad 0 \end{aligned}$$

Using the matrix transformation approach presented above, Vaidya [5] presented a secure protocol (as shown in Algorithm 2) for two party secure computation by assuming that one party holds the objective function and the other party holds the constraints (Scenario 1). The key idea is to transform the constraint matrix and objective function and release it to one of the parties, which can then centrally solve the problem. The transformation is done using the secure scalar product protocol to achieve secure matrix multiplication. The transformation matrix Q is split between the two parties.

The security analysis of Protocol 2 in [5] shows that: P_1 learns only pk and P_2 learns only $M(Q_1 + Q_2)$ and $C^T(Q_1 + Q_2)$; using this it is not possible for P_1 (and P_2) to exactly compute M and Q_2 (C^T and Q_1). However, a potential privacy leakage has been identified in the work of Bednarz et al. [40] due to the exact structure of the transformation matrix. [1] We also illustrate that our approaches eliminate this attack for arbitrarily partitioned LP problems in this paper.

[1] Bednarz et al. [40] also shows that Q should be a monomial matrix to ensure transformation correctness. Thus, $Q_1 + Q_2$ cannot be guaranteed to be monomial if each party choose a monomial matrix Q_1 and Q_2 respectively. In this paper, we also resolve this issue.

Protocol 2 Secure transformation protocol in two-party distributed LP problem without partition[5]

Require: P_1 has the $n \times n$ matrix Q_1 and the coefficients of the objective function C^T

Require: P_2 has the $n \times n$ matrix Q_2 and the $m \times n$ matrix M

1: P_2 generates a private and public key pair (sk, pk).
2: P_2 computes the $m \times n$ matrix MQ_2
3: P_2 computes $(MQ_2)' = Enc_{pk}(MQ_2)$ (i.e., the encryption of each element of MQ_2 with pk – a random nonce is chosen for each encryption)
4: P_2 computes $Q_2' = Enc_{pk}(Q_2)$ (i.e., the encryption of each element of Q_2 with pk – a random nonce is chosen for each encryption)
5: P_2 computes $M' = Enc_{pk}(M)$ (i.e., the encryption of each element of M with pk – again, a random nonce is chosen for each encryption)
6: P_2 sends pk, M', Q_2' and $(MQ_2)'$ to P_1
$\{P_1$ now computes the encrypted matrix S' and vector V' as follows$\}$
7: **for** each row i of M' and each column j of Q_1 **do**
8: P_1 computes $S'_{ij} = \prod_{k=1}^{n} M'^{Q_{1kj}}_{ik} * (MQ_2)'_{ij}$
9: P_1 computes $(C^TQ_1)' = Enc_{pk}(C^TQ_1)$ (i.e., the encryption of each element of C^TQ_1 with pk – a random nonce is chosen for each encryption)
10: **for** each each column i of Q_2' **do**
11: P_1 computes $V'_i = \prod_{k=1}^{n} Q_2'^{C_k^T} * (C^TQ_1)'_i$
12: P_1 sends S' and V' to P_2
13: P_2 computes $S = Dec_{sk}(S') = M(Q_1 + Q_2)$ and $V = C^T(Q_1 + Q_2)$

4.2 Privacy in Transformed LP Problem

In a typical arbitrarily partitioned LP problem (i.e. Scenario 2), the LP problem is formulated by multiple participants. Since each party has its own portion in the collaborative constraints and objective function, and each party's portion of constraints indicate the private limitations and capacities of this company, we thus regard each share of the constraints as the main information to be protected in the distributed LP problem. Moreover, the transformation matrix of each party should be considered as private data. Specifically, in two-party arbitrarily partitioned distributed LP problem, if P_1 knows P_2's transformation matrix Q_2 and solves the problem with constraint matrix $(M_1 + M_2) \times (Q_1 + Q_2)$, P_1 can infer the constraints share of P_2: $M = M(Q_1 + Q_2) * (Q_1 + Q_2)^{-1}$ and $M_2 = M - M_1$. Finally, if the objective vector C^T is (vertically) partitioned and it should be kept private, our following protocols can provide the same secure transformation for partitioned vectors as well as partitioned matrices.

Hence, we need to ensure that for each party, the share of the constraint Matrix, the share of the objective function and the transformation matrix are all kept private.

During the transformation, we should prevent private data from being computed by adversaries at every step. We consider all the cooperative companies as semi-honest parties who strictly follow the protocol but may try to learn more information about the other parties based on whatever data they may see during the protocol. It is also possible to adopt a fully malicious model, wherein any party may act in any fashion. Typically, any protocol resilient to semi-honest adversaries can be converted into an equivalent protocol that is resilient to malicious adversaries, though at significant added computation cost. In our case, since we use the secure scalar product as the key primitive, it is easily possible to do this. Therefore, we restrict our attention to semi-honest adversaries.

5 Secure Transformation for Two-party Arbitrarily Partitioned LP

We now extend the work in [5] to present a secure approach to solve the distributed LP problem where the constraints and the objective function are arbitrarily partitioned between two parties.

5.1 Secure Transformation

Before discussing the secure transformation for arbitrarily partitioned constraint matrix and objective function, we first review the arbitrary partition between two parties. As described in Scenario 2, assuming that Walmart and CostCo need to minimize their total shipping cost, the global objective function vector and constraint matrix are $C = C_1 + C_2$ and $M = M_1 + M_2$ where C^T represents the unit shipping cost vector for every party and every pair of locations.

We thus extend the transformation based approach in Protocol 2 to privately solve this two-party arbitrarily partitioned LP problem. Protocol 3 presents the detailed secure transformation.

Protocol 3 Secure transformation protocol for two-party arbitrarily partitioned LP

Require: P_1 has the $n \times n$ matrix Q_1, the share of the $m \times n$ matrix M_1 and objective vector C_1^T;
Require: P_2 has the $n \times n$ matrix Q_2, the share of the $m \times n$ matrix M_2 and objective vector C_2^T;
{All the encryptions are based on each elements in the matrices and a random nonce is chosen for each encryption}
1: P_1 generates a private and public key pair (sk, pk)
2: P_1 computes the $m \times n$ matrices: $M_1' = Enc_{pk}(M_1)$ and the n-dimensional vector $(C_1^T)' = Enc_{pk}(C_1^T)$
3: P_1 sends pk, M_1', and $(C_1^T)'$ to P_2
4: P_2 computes $M_2' = Enc_{pk}(M_2)$, $(C_2^T)' = Enc_{pk}(C_2^T)$, $M' = M_1' * M_2' = Enc_{pk}(M_1 + M_2)$, $(C^T)' = Enc_{pk}(C_1^T) * Enc_{pk}(C_2^T)$ and $Q_2' = Enc_{pk}(Q_2)$. ($*$ represents the multiplication operation for each elements in the same position of two matrices)
5: **for** each row i of M' and each column j of Q_2 **do**
6: P_2 computes $(MQ_2)'_{ij} = \prod_{k=1}^{n} (M_{ik})'^{(Q_2)_{kj}}$
7: **for** each row i of $(C^T)'$ and each column j of Q_2 **do**
8: P_2 computes $(C^TQ_2)'_{ij} = \prod_{k=1}^{n} (C^T{}_{ik})'^{(Q_2)_{kj}}$
9: P_2 sends $(MQ_2)'$ and $(C^TQ_2)'$ back to P_1
10: **for** each row i of $(MQ_2)'$ and each column j of Q_1 **do**
11: P_2 computes $(MQ_2Q_1)'_{ij} = \prod_{k=1}^{n} (MQ_{2ik})'^{(Q_1)_{kj}}$
12: **for** each row i of $(C^TQ_2)'$ and each column j of Q_1 **do**
13: P_2 computes $(C^TQ_2Q_1)'_{ij} = \prod_{k=1}^{n} (C^TQ_{2ik})'^{(Q_1)_{kj}}$
14: P_1 decrypts $(MQ_2Q_1)'$ and $(C^TQ_2Q_1)'$ with its private key sk

Finally, we should let at least one party obtain the transformed objective function vector C^TQ and constraint matrix MQ and then solve the problem. In Protocol 3, P_1 solves the problem and it eventually obtain C^TQ and MQ with decryption. Specifically, P_1 first sends its transformed and encrypted matrix and the public key to P_2, and acquire the encrypted and transformed collaborative constraints matrix from P_2. Finally, P_1 can decrypt $(MQ_2Q_1)'$ and $(C^TQ_2Q_1)'$, and solve the LP problem with the transformed matrix and objective vector.

After solving the transformed problem (optimal solution $y*$), P_1 and P_2 jointly reconstruct $x* =$

Q_2Q_1y*. Bednarz et al. [40] proposed a possible attack on inferring Q with known C^TQ, C^T, $y*$ and $x* = Qy*$. First, since Q is known as a monomial matrix, all the permutations of Q can be enumerated and computed according to the known vectors C^TQ and C^T. Second, the adversary can determine the exact case of Q in all the permutations by verifying $x* = Qy*$ one by one. However, the attack can be eliminated in our secure transformation for arbitrarily partitioned LP problem. First, $x*$ is partitioned and each share of the solution is kept private, thus no solution can be used to verify the permutations. Second, C^T is partitioned and each share of C^T (C_1^T or C_2^T) can be permutated before transformation as well (i.e. if C^T represents the unit transportation cost of every pair of locations for every party, permutation is necessary simply because the unit cost may be identical for all parties in the same route. After individually permutating C^T and transforming C^T, the transformed objective vector is not disclosed). Since C_1^T is unknown to P_2 (so does C^T), P_2 cannot compute Q based on an unknown vector C^T and the transformed objective vector C^TQ. We discuss the security of Protocol 3 in Section 5.2.

Furthermore, without loss of generality, we let P_1 solve the LP problem. Since either P_1 or P_2 can be the party to solve the transformed problem (in fact an untrusted third party can also be used to do the same), Protocol 3 does not violate fairness in the secure computation.

5.2 Security Analysis

Theorem 1 *P_1 learns only MQ_2Q_1 and $C^TQ_2Q_1$ and P_2 learns nothing in Protocol 3.*

	P_1	P_2
Hold	Q_1, (sk, pk), M_1	Q_2, M_2
Acquire	$(MQ_2Q_1)'$ and $(C^TQ_2Q_1)'$	M_1', pk
Encrypt (or Compute)	$(MQ_2Q_1)', (C^TQ_2Q_1)', M_1', (C_1^T)'$	$(C^T)', M', (MQ_2)'$
Decrypt	MQ_2Q_1 and $C^TQ_2Q_1$	

Table 1: Matrices in Protocol 3

Proof 1 *Before showing what P_1 and P_2 can learn in each step of Protocol 3, we first collect the transferred matrices in Table 1 (note that C_1^T and C_2^T can be considered as a row of M_1 and M_2 respectively). We thus have:*

1. *P_1 holds M_1 and Q_1 while P_2 holds M_2, and Q_2. All of them are private data for themselves;*
2. *P_2 acquires M_1' from P_1, but P_2 cannot decrypt them with a public key pk;*
3. *P_2 computes(encrypts) $(C^T)', M'$, and $(MQ_2)'$ using pk, but nothing can be learnt from these steps, because all the operations are based on public key (pk) encrypted objects;*
4. *P_1 decrypts $(MQ_2Q_1)'$ and $(C^TQ_2Q_1)'$ to get MQ_2Q_1 and $C^TQ_2Q_1$.*

In Protocol 3, on one hand, P_1 can only obtain $(M_1 + M_2)Q_2Q_1$. It's impossible to calculate M_2 and Q_2 with known matrices M_1, Q_1, and the probability of guessing such private numbers is fairly low (According to the discussion in Section 5.1, the adversary has no disclosed solutions and objective vectors to verify and determine Q using the work of Bednarz et al. [40]). Specifically, even if P_1 can obtain the $m \times n$ matrix $(M_1 + M_2)Q_2$ by decrypting the cipher obtained from P_2, it is secure because P_1 doesn't know any factor of the product of matrix multiplication (neither of $(M_1 + M_2)$ and Q_2 can be inferred - in

fact, infinite such solutions are possible). Thus, P_1 cannot obtain the private data (the share of constraint matrix/objective and the transformation matrix) from the other party P_2.

On the other hand, with an public key encryption for M_1, it is also impossible for P_2 to discover M_1 or Q_1 from the given matrices.

Hence, Protocol 3 is secure.

6 Secure Transformation for Multi-party Arbitrarily Partitioned LP

While the protocol developed above is limited to two parties, in general any cooperative computation is likely to involve more than two parties (i.e. more than two corporations cooperate to ship goods). Therefore, in this section, we consider the multi-party distributed case and enable secure optimization in this context.

6.1 Securing Multi-Party Transformation Protocol

In multi-party arbitrarily partitioned LP, we can locally encrypt the matrix transformation and multiplication for all parties using their own public keys, and each party decrypts its transformed matrices which are encrypted, transformed and transferred by other parties.

Specifically, to enforce privacy protection, each party is allowed to generate and send random matrices (or vectors) for each encryption. Suppose that we are securing an ℓ-party collaborative LP problem (Scenario 2), each party P_i first distributes its public key pk_i to all other parties. Then, P_i encrypts and transforms its matrix share M_i with its privately held monomial matrix Q_i and public key P_i: $Enc_{pk_i}(M_iQ_i)$. Later, all parties jointly transform party P_i's matrix M_i to obtain $Enc_{pk_i}(M_iQ_1 \cdots Q_\ell + R)$ where R is an encrypted random number co-held by all parties. Finally, each party P_i decrypts its transformed matrix with its private key sk_i, and all parties securely sum the transformed matrix $MQ = (M_1 + \cdots + M_\ell)Q_1 \cdots Q_\ell$. The detailed steps are given in Protocol 4.

Table 2 shows each party's matrices after decrypting the summed matrices. Some essential points are worth noting as below:

- * stands for the multiplication on each element in the same position of two matrices rather than matrix multiplication.
- $R_{(i,j)}$ stands for the random matrix generated by party P_j for transformation P_i's matrix share M_i.
- If substituting M_i with C_i^T, the table will be each party's vectors (also in Table 2).

P_1	P_2	...	P_ℓ
$M_1Q_1Q_2...Q_\ell + \sum_{j=2}^{l} R_{(1,j)}$	$-R_{(1,2)}$	...	$-R_{(1,\ell)}$
$-R_{(2,1)}$	$M_2Q_1Q_2...Q_\ell + \sum_{j=1,j\neq 2}^{\ell} R_{(2,j)}$	...	$-R_{(2,\ell)}$
...	...	...	...
$-R_{(\ell,1)}$	$-R_{(\ell,2)}$	...	$M_\ell Q_1Q_2...Q_\ell + \sum_{j=1}^{\ell-1} R_{(\ell,j)}$

Table 2: Each Party's Matrices in Multi-Party Collaboration after Decryption

Protocol 4 Secure transformation protocol for multi-party arbitrarily partitioned LP problem

Require: P_i has the $n \times n$ matrix Q_i, $m \times n$ matrix M_i and n-dimensional vector C_i^T $(1 \leq i \leq \ell)$;

1: **for** ith party $(i = 1, 2, 3, \ldots, \ell)$ **do**
2: P_i generates a key pair (sk_i, pk_i), and sends pk_i to all the remaining parties.
3: P_i computes $(M_iQ)' = Enc_{pk_i}(M_iQ_i)$.
4: P_i sends $(M_iQ)'$ to the next party (one out of P_j where $j = 1, 2, 3, \ldots, \ell$ and $j \neq i$).
5: **for** jth party P_j $(j = 1, 2, 3, \ldots, \ell$ and $j \neq i)$ **do**
6: P_j generates a $m \times n$ random matrix $R_{(i,j)}$ and encrypts it with pk_i: $R'_{(i,j)} = Enc_{pk_i}(R_{(i,j)})$.
7: **for** each row a of M_i' and each column b of Q_j **do**
8: P_j computes $(M_iQ)_{(i,j)}{}'_{ab} = \prod_{h=1}^{n} M_i Q'^{Q_{jhb}}_{ah} * R_{(i,j)}{}'_{ab}$
9: P_j sends the updated $(M_iQ)'$ to the next party
10: Finally, the last party sends $(M_iQ)'$ back to party P_i
11: **for** every party P_i $(i = 1, 2, 3, \ldots, l)$ **do**
12: P_i decrypts $(M_iQ)'$ with its private key sk_i to obtain $M_iQ_1Q_2 \ldots Q_\ell + \sum_{j=1,j\neq 2}^{\ell} R_{(i,j)}$.
13: P_i vertically subtracts all the random matrices (generated by itself), and obtains a matrix $S_i = M_iQ_1 \cdot \cdot \cdot Q_\ell + \sum_{j=1,j\neq i}^{\ell} R_{(i,j)} - \sum_{j=1,j\neq i}^{\ell} R_{(j,i)}$.
14: All parties sum the matrices $S_i (i = 1 \ldots \ell)$ for all ℓ parties to obtain $MQ = (M_1 + \cdots + M_\ell)Q_1 \cdots Q_\ell$.
15: All parties securely compute $C^TQ = (C_1^T + \cdots + C_\ell^T)Q_1 \cdots Q_\ell$ following the above steps again.
{**For securely transforming** $C_1, \ldots, C_l$**, we can also transform** $C_1, \ldots, C_l$ **as any row of** $M_1, \ldots, M_l$ **and securely compute** $(C_1 + \cdots + C_l)^T Q_1 \cdots Q_l$ **using this protocol**}

6.2 Security Analysis

We have analyzed the security of Protocol 3 in Section 5. Similarly, we can analyze all the potential matrices (or vectors) in all communication steps of Protocol 4.

Theorem 2 *Protocol 4 ensures that: ($\forall i \in [1, \ell]$) P_i cannot learn ($\forall j \in [1, \ell]$ and $j \neq i$) M_j and Q_j.*

	$P_i (i = 1, 2, ..., \ell)$	$P_j (j \neq i, j = 1, 2, ..., \ell)$
Hold	Q_i, (sk_i, pk_i), M_i	Q_j, (sk_j, pk_j), M_j
Acquire	$pk_j, Enc_{pk_i}(M_iQ_1 \cdots Q_\ell) * \prod_{\forall j \neq i} * R'_{(i,j)}$, $Enc_{pk_j}(M_jQ_jQ_1 \cdot \cdot) * R'_{(j,1)} * **$	$pk_i, (Enc_{pk_j}(M_jQ_1 \cdots Q_\ell) * \prod_{\forall i \neq j} * R'_{(j,i)}$, $Enc_{pk_i}(M_iQ_iQ_1 \cdot \cdot) * R'_{(i,1)} * **$
Encrypt	$Enc_{pk_j}(M_jQ_jQ_1 \cdots Q_i) * R'_{(j,1)} * * * R_{(j,i)}$	$Enc_{pk_i}(M_iQ_iQ_1 \cdots Q_j) * R'_{(i,1)} * * * R_{(i,j)}$
Decrypt	$M_iQ_1 \cdots Q_\ell + \sum_{j=1,j\neq i}^{\ell} R_{(i,j)}$	$M_jQ_1 \cdots Q_\ell + \sum_{i=1,i\neq j}^{\ell} R_{(j,i)}$
Sum	the sum of M_iQ and random matrices	the sum of M_jQ and random matrices

Table 3: Each Party's Data in Protocol 4

Proof 2 *Table 3 depicts all the exchanged messages in the communications of Protocol 4. We consider two different categories of collaborative party: one is an arbitrary party P_i while the other one is another arbitrary party P_j. We can learn from Protocol 4, Table 2, and 3 as below:*

1. *P_i acquires $\forall j = 1, 2, 3, ..., \ell$ and $j \neq i$, pk_j, $Enc_{pk_i}(M_iQ_1 \cdots Q_\ell) * \prod_{\forall j \neq i} * R'_{(i,j)}$ and $Enc_{pk_j}(M_jQ_jQ_1 \cdot \cdot) * R'_{(j,1)} * **$ from other parties.*

2. *P_i can neither decrypt $Enc_{pk_j}(M_jQ_jQ_1\cdots)*R'_{(j,1)}***$, nor compute the random matrices and other parties's transformation matrices Q_j from its decrypted matrix $M_iQ_1\cdots Q_\ell + \sum_{j=1,j\neq i}^{\ell} R_{(i,j)}$.*

3. *If substituting the constraint matrix share with the objective vector share, the communication remains secure for all parties.*

4. *Protocol 4 does not suffer the attack from the work of Bednarz et al. [40] for the same reason as Protocol 3.*

5. *If more parties are involved in this distributed LP model, the protocol is more secure due to an increasing number ℓ which minimizes the chance of guessing the correct matrices.*

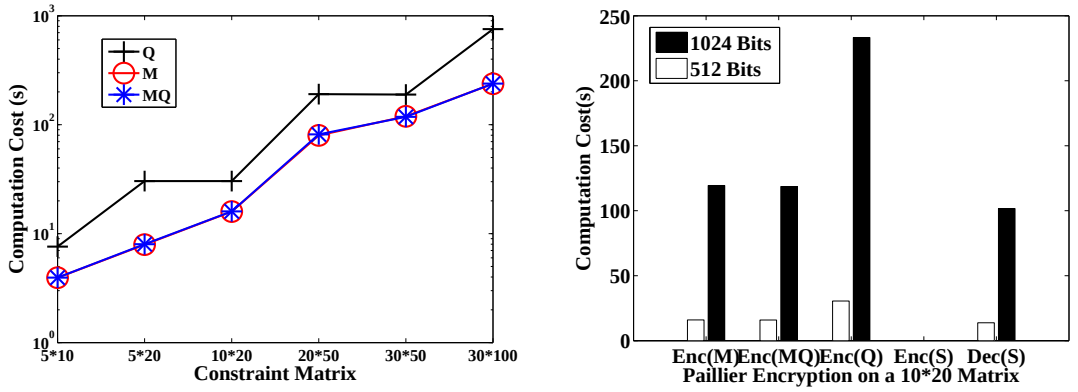
6. *$\ell > 2$ in Protocol 4.*

Finally, each party provides a local vertical sum of all the matrices and vectors in Table 2. This is secure to all parties and no one can exactly learn any other party's private constraint matrix, objective vector and transformation matrix from the sum and the data they provides to other parties.

Similar to two-party arbitrarily partitioned LP problem, we can let an arbitrary party solve the LP problem. After solving the transformed problem (optimal solution $y*$), all parties can jointly reconstruct the original optimal solution $\prod_{i=1}^{\ell} Q_i y$.

7 Experiments

We now experimentally evaluate the efficiency and utility of the proposed approaches. Since the objective vector can be included as part of the constraints matrix, the experiments are based only on the transformation of constraints matrix in Protocol 2, 3 and 4.



(a) Cost of (512 bit) Encryption for varying matrix sizes (b) Comparative cost of 512-bit versus 1024-bit Encryption for different matrices

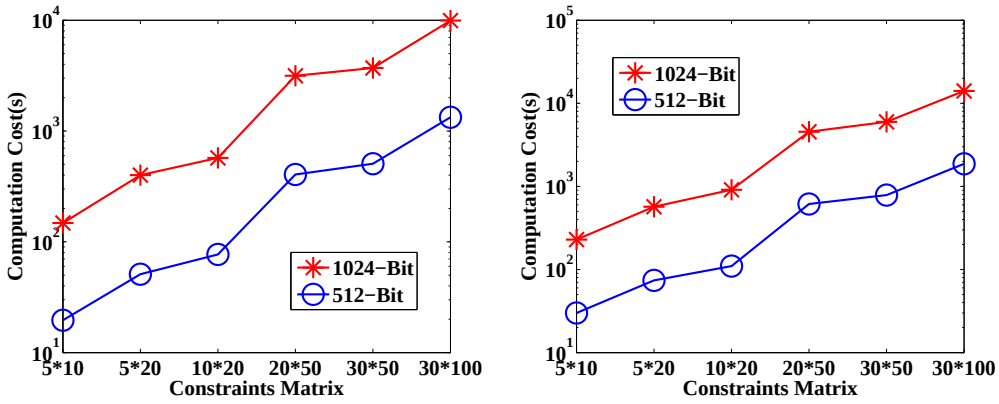
Figure 1: Computation Cost of Matrix Encryption

7.1 Experiments Setup

In our experiments, we simulate all the steps in Protocol 2, 3 and 4, respectively. To test how the constraints matrix affects computation cost, we generate matrices according to the size from set $\{5 \times 10, 5 \times 20, 10 \times 20, 20 \times 50, 30 \times 50, 30 \times 100\}$. All the experiments are performed on a HP machine with Intel Core 2 Duo CPU E8400 3GHz and 3G RAM running Windows XP Professional Operating system. We adopted the constraints matrix with a varying size in securing two-party distributed linear programming problem for Protocol 2 (without partition) and 3 (with arbitrary partition) and tested the total computation cost of encryption and decryption using two different key sizes (512 Bits and 1024 Bits).

For the multi-party protocol, we carried out two groups of experiments to test the total computation cost: first, we fix the number of parties to 5 and find the total computation cost for varying matrix sizes; second, we fix the constraints matrix to a 10×20 matrix and find the total computation cost while varying the number of parties from 3 to 10. We also implement our alternative solution on securing multi-party distributed linear programming problem for reducing running time – we assume that every three parties share the same transformation matrix and all the parties are enabled to work in parallel when encrypting and decrypting matrices. Similar to the tests on total computation cost of multi-party arbitrarily partitioned LP problem, another two groups (varying matrix size and number of parties) of experiments are also carried out under such conditions.

7.2 Computation Complexity

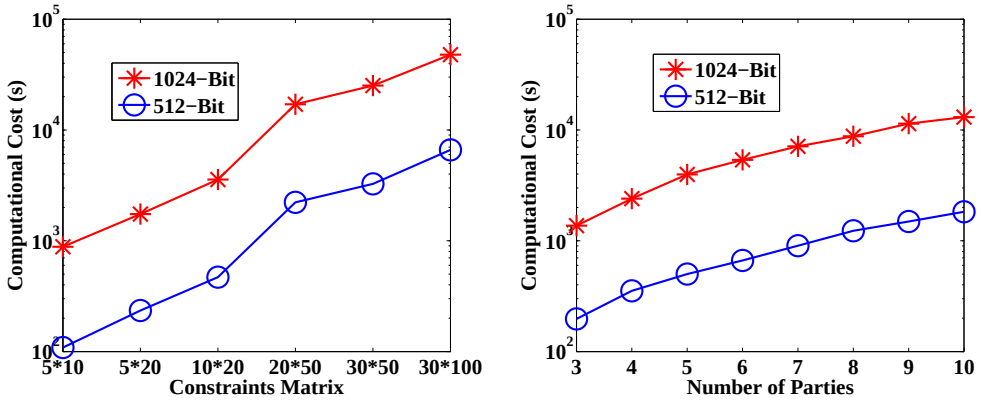


(a) Protocol 2 (b) Protocol 3

Figure 2: Securing Two-party Distributed Linear Programming

We now look at the experimental results for Protocol 2, 3 and 4. Before coming to specific protocols, we first discuss the cost of encrypting the matrices using Paillier encryption. As shown in Figure 1(a), the required runtime for encrypting M and MQ increases gradually with an increasing size of M, whereas the runtime for encrypting Q increases only if we enlarge the number of columns of the constraints matrix. This makes sense since the number of columns determines the size of transformation matrix Q. Furthermore, we plot every individual encryption and decryption step (using 512 and 1024 bits) in Figure 1(b). Since $m < n$, encrypting Q requires more computation. Encrypting S by $S'_{ij} = \prod_{k=1}^{n} M'^{Q_{1kj}}_{ik} * R'_{ij}$ requires least time because multiplication and exponentiation take significant less time than complete encryption of a matrix. Decrypting S' to obtain MQ requires less time than each single encryption.

To test the effectiveness of constraints matrices and encryption for Protocol 2 and 3, we still adopt a varying size of constraints matrix and use 512 and 1024 Bits encryption. We sum over the computation costs of all the steps in the protocol. The computation costs of Protocol 2 and 3 show an increasing trend on varying size of constraints matrix, as shown in Figure 2(a) and 2(b) respectively (Note though that the running time is in log scale – the overall time can be reduced using cryptographic accelerators, and parallelization, as we show below.).

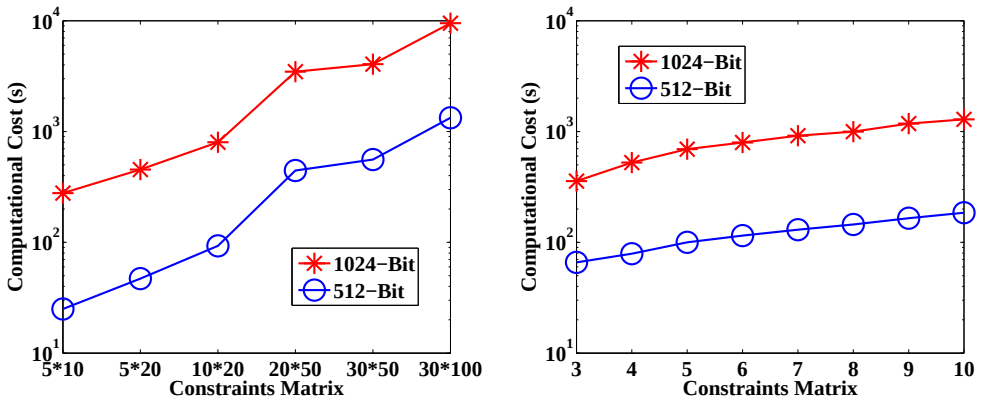


(a) Total Cost on Varying Size of Matrices (5 Parties) (b) Total Cost on Varying Number of Parties (10×20 Constraints Matrix)

Figure 3: Securing Multi-party Distributed Linear Programming (Protocol 4)

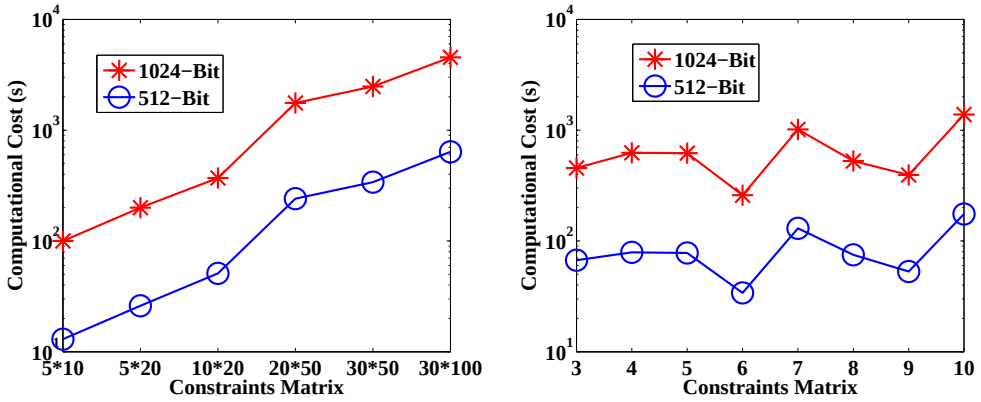
As for Protocol 4, designed for securing multi-party distributed linear programming problem, we plot the computation costs of (512 and 1024 Bits) encryption and decryption using two variables: the size of constraints matrix and number of parties. As expected, Figure 3(a) and 3(b) show an increasing trend on total computation costs of Protocol 4 as we increase the size of constraints matrix or number of parties (the other variable is fixed). Alternatively, if all parties work in parallel, the runtime of encryption can be greatly reduced in terms of the number of parties and the computing capability of each party. Figure 4(a) and 4(b) present the total computation cost by simulating Protocol 4 and assuming that all parties have the same computing capability and they can work in parallel. At this time, the total runtime should be lower as expected.

Moreover, if we adopt the solution for securing multi-party linear programming problem as discussed in Section **??**, the total computation cost for multi-party computation can be significantly reduced according to the total number of divided transformation matrices. Figure 4(c) plots the total runtime while varying the size of constraints matrix where there exist 5 collaborative parties with 3 parties sharing a common transformation matrix Q_1 and remaining 2 parties sharing another common transformation matrix Q_2. The running time behavior is similar to Figure 3(a) and 4(a). matrix and vary the number of parties, the runtime is significantly low in our simulation when the number of parties can be exactly divided by 3. All the parties can work together without idling at this time. If one of the parties holds a separate transformation matrix, the exhibited runtime stands out as shown in Figure 4(d). Thus, the computation cost of Protocol 4 (involving large number of parties) can be significantly reduced using this approach. Alternatively, to securely and efficiently solve distributed linear programming problems, we also proposed a secure revised simplex algorithm based on a basic LP solving methods – revised simplex (for two parties)[41]. As addressed in that paper, a small linear programming problem with 25 rows and 35 columns, the cost per iteration would be 23s; A moderate size problem (200 rows and 300 columns) would be 5 minutes; A large problem with



(a) Runtime on Varying Size of Matrices (5 Parties' Computing)

(b) Runtime on Varying Number of Computing Parties (10×20 Constraints Matrix)



(c) Runtime on Varying Size of Matrices (5 Parties' Computing with 3 Parties holding a same transformation matrix and remaining 2 Parties holding a same transformation matrix)

(d) Runtime on Varying Number of Parties (10×20 Constraints Matrix) with every 3 parties holding a same transformation matrix and remaining party (or two parties) holding a same matrix

Figure 4: Parallel Computing in Secured Multi-party Distributed Linear Programming (Protocol 4)

800 rows and 1300 columns would require 40 minutes. The time complexity can be reduced for two-party computation.

To sum up, even if it takes more running time to encrypt large matrices using 1024 Bits, the total encryption time is polynomial to a single encryption and our algorithms are efficient for practical distributed LP problems. Thus encryption of the transformation matrix can also be considered as off-line computation cost. Given this, the online communication and computation costs are reasonable in all distributed linear programs.

Finally, note that our approach is based on transforming the original LP problem to a new LP problem, which is then solved to give a solution which can be transformed back to get the actual solution. We therefore would like to know if it takes more time to centrally solve the transformed matrix as opposed to the original matrix – i.e., does the transformation change the characteristics of the matrix resulting in much longer runtimes? To check this, in Figure 5 we compare the runtime of solving the original LP problem with the runtime of solving the new transformed LP problem. Our experiments show that the difference

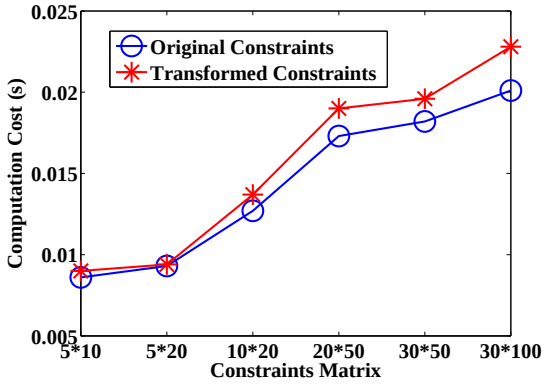


Figure 5: Solving Original LP and Transformed LP Problems

between the runtime is negligible, thus indicating that transformation does not increase the complexity of the problem.

8 Conclusion and Future Work

In this paper, we have discussed the privacy issues in distributed linear programming problems and motivated several different possible data distributions in real situations. We have extended the approach of Vaidya[5] to tackle the privacy issues for two party linear programming in a more general case wherein the constraints and objective function are arbitrarily partitioned between both parties. We have then extended this approach to enable linear programming over multi-party arbitrarily partitioned data.

After enriching the literature on privacy-preserving linear programming [42][43][44] and also related areas [45], several questions remain open for the future. First, the security implied by the solution is still somewhat heuristic (only protocol security in [45] and [41] have been formally proven). While it is clear that the original constraints, objective function and transformation matrix cannot be reconstituted or calculated from the data each party gets from others, what about other features. Can you infer anything about the hardness of the problem, or the type of constraints, or their relation to each other? All of these are significant questions that need to be looked at in more detail for practical deployment of the solution. Some of the work in secure computation shows possible approaches. If we could show that the results of any function computed from the constraints in polynomial time are indistinguishable from random numbers uniformly generated, we can prove that transformation leaks nothing. This needs to be further explored. Another observation is that it should be possible to adapt the transformation approach to work for integer programming as well as quadratic programming. For integer programming the case is exactly the case as for linear programming. For quadratic programming, the form of the constraints changes – therefore the transformation method must take this into account. However, polynomial evaluation may be used to still perform the transformation. We intend to further explore all of these problems in the future.

References

[1] A. C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, (Los Alamitos, CA, USA), pp. 162–167, IEEE, IEEE Computer

Society, 1986.

[2] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game - a completeness theorem for protocols with honest majority," in *Proceedings of the 19th ACM Symposium on the Theory of Computing*, (New York, NY), pp. 218–229, ACM, 1987.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *In Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pp. 1–10, 1998.

[4] E. Turban, R. K. Rainer, and R. E. Potter, "Introduction to information technology, chapter 2nd, information technologies: Concepts and management," *John Wiley and Sons, 3rd edition*.

[5] J. Vaidya, "Privacy-preserving linear programming," in *SAC*, pp. 2002–2007, 2009.

[6] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections (extended abstract)," in *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 544–553, ACM Press, 1994.

[7] R. Canetti and R. Gennaro, "Incoercible multiparty computation," in *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, (Washington, DC, USA), p. 504, IEEE Computer Society, 1996.

[8] S. Micali and P. Rogaway, "Secure computation," in *Advances in Cryptology: CRYPTO '91*, vol. 576 of *Lecture Notes in Computer Science*, pp. 392–404, Springer-Verlag, 1991.

[9] R. Canetti, "Asynchronous secure computation," Tech. Rep. 755, CS Department, Technion, 1992.

[10] X. Deng and C. H. Papadimitriou, "Distributed decision-making with incomplete information," in *Proceedings of the 12th IFIP Congress*, 1992.

[11] C. H. Papadimitriou and M. Yannakakis, "On the value of information in distributed decision-making (extended abstract)," in *PODC '91: Proceedings of the tenth annual ACM symposium on Principles of distributed computing*, (New York, NY, USA), pp. 61–64, ACM Press, 1991.

[12] C. H. Papadimitriou and M. Yannakakis, "Linear programming without the matrix," in *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 121–129, ACM Press, 1993.

[13] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabar, "Distributed constraint satisfaction for formalizing distributed problem solving," in *In Proceedings of International Conference on Distributed Computing Systems*, pp. 614–621, 1992.

[14] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "An asynchronous complete method for distributed constraint optimization," in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, (New York, NY, USA), pp. 161–168, ACM Press, 2003.

[15] R. Mailler and V. Lesser, "Solving distributed constraint optimization problems using cooperative mediation," *aamas*, vol. 01, pp. 438–445, 2004.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

[17] W. Zhang and L. Wittenburg, "Distributed breakout algorithm for distributed constraint optimization problems - dbarelax," in *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-03)*, 2003.

[18] M. Yokoo and K. Hirayama, "Distributed breakout algorithm for solving distributed constraint satisfaction problems," in *Proceedings of the First International Conference on Multi-Agent Systems* (V. Lesser, ed.), MIT Press, 1995.

[19] A. Petcu and B. Faltings, "A scalable method for multiagent constraint optimization," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, Aug. 2005.

[20] A. Petcu and B. Faltings, "Approximations in distributed optimization," in *CP05 - workshop on Distributed and Speculative Constraint Processing(DSCP)*, Oct. 2005.

[21] A. Petcu and B. Faltings, "Incentive compatible multiagent constraint optimization," in *WINE'05: Workshop on Internet and Network Economics*, Dec. 2005.

[22] A. Petcu and B. Faltings, "Optimal solution stability in continuous time optimization," in *IJCAI05 - Distributed Constraint Reasoning workshop(DCR05)*, Aug. 2005.

[23] A. Petcu and B. Faltings, "Superstabilizing, fault-containing multiagent combinatorial optimization," in *Proceedings of the National Conference on Artificial Intelligence(AAAI-05)*, July 2005.

[24] A. Petcu and B. Faltings, "Distributed generator maintenance scheduling," in *Proceedings of the First International ICSC Symposium on artificial intelligence in energy systems and power (AIESP'06)*, Feb. 2006.

[25] M.-C. Silaghi and V. Rajeshirke, "The effect of policies for selecting the solution of a discsp on privacy loss," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1396–1397, 2004.

[26] M. C. Silaghi and D. Mitra, "Distributed constraint satisfaction and optimization with privacy enforcement," *iat*, vol. 00, pp. 531–535, 2004.

[27] K. Suzuki and M. Yokoo, "Secure generalized vickrey auction using homomorphic encryption," *Lecture Notes in Computer Science*, vol. 2742.

[28] M. Yokoo, K. Suzuki, and K. Hirayama, "Secure distributed constraint satisfaction: Reaching agreement without revealing private information," in *In Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming (CP-2002)*, 2002.

[29] M.-C. Silaghi, B. Faltings, and A. Petcu, "Secure combinatorial optimization simulating dfs tree-based variable elimination," in *9th Symposium on Artificial Intelligence and Mathematics*, (Ft. Lauderdale, Florida, USA), Jan 2006. http://www2.cs.fit.edu/ msilaghi/papers/.

[30] M. J. Atallah, V. Deshpande, H. G. Elmongui, and L. B. Schwarz, "Secure supply-chain protocols," in *Proceedings of the 2003 IEEE International Conference on E-Commerce*, (Newport Beach, California), June24-27 2003.

[31] J. Li and M. J. Atallah, "Secure and private collaborative linear programming," in *Proceedings of the 2nd International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1 – 8, Nov.17–20 2006.

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

[33] W. D. Y. S. Han and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *In Proceedings of the 2004 SIAM International Conference on Data Mining*, 2004.

[34] M. Blum and S. Goldwasser, "An efficient probabilistic public-key encryption that hides all partial information," *In R. Blakely, editor, Advances in Cryptology – Crypto 84 Proceedings, Springer-Verlag*.

[35] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret," *In A. Odlyzko, editor, Advances in Cryptography - CRYPTO86: Proceedings, Springer-Verlag, Lecture Notes in Computer Science*, vol. 263.

[36] D. Naccache and J. Stern, "A new public key cryptosystem based on higher residues," in *In Proceedings of the 5th ACM conference on Computer and communications security*, pp. 59–66, 1998.

[37] P. Paillier, "Public key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - Eurocrypt '99 Proceedings, LNCS 1592*, pp. 223–238, 1999.

[38] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *Advances in Cryptology - Eurocrypt '98, LNCS 1403*, pp. 308–318, 1998.

[39] W. Du and M. J. Atallah, "Privacy-preserving cooperative scientific computations," in *In Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pp. 273–282, 2001.

[40] A. Bednarz, N. Bean, and M. Roughan, "Hiccups on the road to privacy-preserving linear programming," in *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, WPES '09, (New York, NY, USA), pp. 117–120, ACM, 2009.

[41] J. Vaidya, "A secure revised simplex algorithm for privacy-preserving linear programming," in *AINA '09: Proceedings of the 23rd IEEE International Conference on Advanced Information Networking and Applications*, 2009.

[42] Y. Hong, J. Vaidya, and H. Lu, "Efficient distributed linear programming with limited disclosure," in *DBSec*, pp. 170–185, 2011.

[43] Y. Hong, J. Vaidya, and H. Lu, "Secure and efficient distributed linear programming," *Journal of Computer Security*, 2012.

[44] Y. Hong and J. Vaidya, "An inference-proof approach to privacy-preserving horizontally partitioned linear programs," *Optimization Letters*, to Appear.

[45] Y. Hong, J. Vaidya, H. Lu, and B. Shafiq, "Privacy-preserving tabu search for distributed graph coloring," in *SocialCom/PASSAT*, pp. 951–958, 2011.