Section 2

Information Privacy

Chapter 5

Privacy Preserving Linear Programming*

Yuan Hong[†,¶], **Jaideep Vaidya**[‡,‖], **Nicholas Rizzo**[†,**], **and Qi Liu**[§,††]

[†]Department of Information Security and Digital Forensics
University at Albany, State University of New York

[‡]Department of Management Science and Information Systems
Rutgers University

[§]Department of Accounting and Business Law Siena College

[¶]hong@albany.edu
[‖]jsvaidya@cimic.rutgers.edu
[**]nrizzo@albany.edu
[††]qliu@siena.edu

With the rapid increase in computing, storage, and networking resources, data is not only collected and stored, but also analyzed. This creates a serious privacy problem which often inhibits the use of this data. In this chapter, we investigate and resolve the privacy issues in a fundamental optimization problem — Linear Programming (LP) which is formulated by data collected from different parties. We first consider the case where the objective function and constraints of the LP problem are not partitioned between two parties where one party privately holds the objective function while the other party privately holds the constraints. Second, we present a privacy preserving technique for the case that objective function and constraints are *arbitrarily partitioned* between two parties where each party privately holds a share of objective function and constraints. Finally, we extend the technique for securely solving two-party arbitrarily partitioned LP problems to a multiparty scenario. In summary, we propose a set of efficient and secure transformation-based techniques that create significant value-added benefits of being independent of the specific algorithms used for solving the LP problem.

*The preliminary version appeared in the *Proceedings of the 24th ACM Symposium on Applied Computing* (SAC '09)

1. Introduction

With the rapid increase in computing, storage, and networking resources, data is not only collected and stored, but also analyzed. 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 (SMC) (Yao, 1986) addresses exactly this problem and provides theoretical foundations for preserving privacy while analyzing data held by distributed sites. The celebrated results (Ben-Or *et al*., 1998; Goldreich *et al*., 1987; Yao, 1986) in this area show that any function can be securely computed in polynomial time with respect to the size of the circuit required to compute the function. However, this can lead to very inefficient solutions for complex functions or for large-scale input sizes. More efficient privacy preserving solutions are necessary for such important subproblems.

In this chapter, we look at the specific problem of how organizations optimize the allocation of global resources using Linear Programming (LP) while preserving the privacy of local information. As an important fundamental problem, LP is a subclass of optimization problems where all of the constraints and the objective function are *linear*. It is applicable to a wide variety of real world problems in many industries such as transportation, commodities, airlines, and communication.

More importantly, many practical LP models involve multiple parties to collaboratively formulate and solve the problem with global as well as local information. A specific example can be seen in the packaged goods industry, where delivery trucks are empty 25% of the time. Just 2 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 via their collaboration (freight costs were cut by 15%, for an annual savings of $2 million (Turban *et al*., 2004)). However, this required all the participating parties to send their local data to a centralized site, which charges the collaborative logistics services. Since the disclosed data may contain a considerable amount of proprietary information, different parties' privacy might be compromised in such services.

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 into 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 they can determine what trucks should make their way to which ports to be loaded onto certain ships i.e. solve the classic transportation

problem (modeled as LP problems), 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.

To tackle the above issues, the proposed technique in this chapter could make such collaboration possible without the release of proprietary information. The rest of this chapter is organized as follows. Section 2 reviews the literature relevant to this study. Section 3 introduces the fundamental LP model and the collaborative LP models, and we also present a transformation-based scheme for a two-party collaborative LP problem in this section. Section 4 presents the algorithms for securely solving two-party collaborative LP problems. Section 5 extends the algorithm to solve multi-party collaborative LP problems. Finally, Section 6 concludes this chapter and discusses the future work.

2. Related Work

2.1. SMC

Privacy preserving LP (Vaidya, 2009) is very relevant to a fundamental field — SMC (Yao, 1986). Under the definition of SMC (Yao, 1986), a computation is secure if at the end of the computation, no party learns anything except its own input and the results. The gold standard of provable security is that a trusted third party which performs the entire computation would achieve the same result as the SMC algorithm. Yao first postulated the two-party comparison problem (Yao's Millionaire Protocol) and developed a provably secure solution (Yao, 1986). Goldreich *et al*. (1987) generalized this to multiparty computation and proved that there exists a secure solution for any functionality.

There has been significant theoretical work in this area. Both Yao (1986) and Goldreich *et al.* (1987) assumed polynomially time-bounded passive adversaries. In a line of work initiated by Ben-Or *et al*. (1998), 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*. (1998) assume passive adversaries, while Chaum *et al*. (1988) extend this to active adversaries. Ostrovsky and Yung (1991) introduce the notion of mobile adversaries, where the corrupting 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 (1994), and generalized to arbitrary multiparty computation by Canetti and Gennaro (Canetti and Gennaro, 1996). Much effort has been devoted to developing crisp definitions of security (Canetti, 1992; Goldwasser and Levin, 1991; Mielikainen, 2004). However, due to efficiency reasons, it is

completely infeasible to directly apply the theoretical work from SMC to form secure protocols for optimization such as LP.

2.2. Collaborative/Distributed optimization

Optimization problems occur in all the industries. There is work in distributed optimization (*viz.* collaborative 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 has been investigated in a worst case setting (with no communication between the distributed agents) by Papadimitriou *et al*. (1991) Papadimitriou and Yannakakis (1993). Papadimitriou and Yannakakis (1993) 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. Awerbuch and Azar (1994) proposed a distributed flow control algorithm with a global objective which gives a logarithmic approximation ratio and runs in a polylogarithmic number of rounds. Bartal *et al*.'s distributed algorithm (2004) obtains a better approximation while using the same number of rounds of local communication. Furthermore, some other distributed optimization problems have been studied in literature, such as distributed Constraint Satisfaction Problem (CSP) (Mailler and Lesser, 2004; Modi *et al*., 2003) and distributed dynamic programming (Petcu and Faltings, 2005a, 2005b).

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 seen widespread applications in this area. There is some work in secure optimization. Silaghi and Rajeshirke (2004) show that a secure combinatorial problem solver must necessarily pick the result randomly among optimal solutions to be really secure. Silaghi and Mitra (2004) proposed 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 was proposed by Suzuki and Yokoo (2003), though it is not completely secure under the framework in Silaghi and Rajeshirke (2004). Yokoo *et al*. (2002) also proposed a scheme using public key encryption for secure distributed constraint satisfaction. Silaghi *et al*. (2006) showed 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*. (2003) proposed 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 be leveraged to advance the state-of-the-art, by building general transformations or privacy-preserving variants of well-known methods.

2.2.1. *Privacy-Preserving LP*

Recently, there has been significant interest in the area of privacy-preserving LP. Work in privacy-preserving LP has followed two major directions (Li and Atallah, 2006): (1) the problem transformation approach, and (2) the SMC-based approach.

- **Transformation-based approach.** Du (2001) and Vaidya (2009) transformed the LP problem by multiplying a monomial matrix to both the constraint matrix and the objective function, assuming that one party holds the objective function while the other party holds the constraints. Bednarz *et al*. (2009) pointed out a potential attack to the above transformation approach. To correct the flaw in Vaidya (2009), we revised the transformation and extended the work to the multiparty scenario in this chapter.

  In addition, Mangasarian (2011, 2012) presented two transformation approaches for horizontally partitioned linear programs and vertically partitioned linear programs, respectively. Li *et al*. (2013) extended the transformation approach (2012) for horizontally partitioned linear programs with equality constraints to inequality constraints. We have identified a potential inference attack to Mangasarian and Li's transformation-based approach, and revised the transformation with significantly enhanced security guarantee in Hong and Vaidya, 2014. Meanwhile, Hong extended the above transformation to privacy-preserving non-LP (Hong, 2013), and applied the model to securely anonymize distributed data (Hong *et al*., 2015). Furthermore, Hong *et al*. (2011, 2012) proposed approaches to securely solve the arbitrarily partitioned collaborative LP problems in both semi-honest and malicious adversarial models. More recently, Dreier and Kerschbaum proposed a secure transformation approach for a complex data partition scenario, and illustrated the effectiveness of their approach (Dreier and Kerschbaum, 2011).
- **SMC-based approach.** Li and Atallah (2006) addressed the collaborative LP problem between two parties where the objective function and constraints can be arbitrarily partitioned, and proposed a secure simplex method for such a problem using cryptographic tools. Vaidya (2009) proposed a secure revised simplex approach also using SMC techniques but improved the efficiency when compared with Li and Atallah's approach (2006). Catrina and Hoogh (2010) presented a solution to solve distributed linear program based on secret sharing. The protocols utilized a variant of the simplex algorithm and secure computation with fixed-point rational numbers, optimized for such application.

Furthermore, as for securing collaborative combinatorial optimization problems (particularly those NP-hard problems), Sakuma *et al*. (2007) proposed a genetic algorithm for securely solving two-party distributed Traveling Salesman Problem (TSP). They consider the case that one party holds the cost vector while the other party holds the tour vector (this is a special case of the multiparty collaborative combinatorial optimization). The TSP that is completely partitioned among multiple parties has been discussed but not solved in Sakuma and Kobayashi (2007). Later, Hong *et al*. (2014) consider a more complicated scenario in TSP where the problem is partitioned to multiple (more than two) parties, and securely solve it with privacy-preserving simulated annealing protocol in Hong *et al*. (2014). Meanwhile, Hong *et al*. (2011) also addressed the privacy concern in another NP-hard problem, graph coloring which is partitioned among two or more parties. Then, the graph coloring problem can be securely solved with the privacy-preserving tabu search protocol.

3. Models

In this section, we first review the fundamental LP problem, followed by introducing the LP problems in which the objective function and constraints are partitioned. Furthermore, we propose a transformation-based privacy-preserving approach to solve such LP problems and discuss the potential privacy risks in transformation.

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

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

3.2. Collaborative LP model

Optimization problems (especially issues in LP) have been well studied in the literature — methods have been proposed for the cases where all of the data is available at a central site. Methods have also been proposed for the cases with incomplete information (distributed optimization). However, all solution methods assume that all the necessary data is centralized or freely available. On the contrary, whenever the data is distributed, we have heightened concerns about problems caused by privacy/security. Different parties might own different constraints or even different parts of the same constraint. Thus, if there are several parties solving a collaborative LP problem, data (constraints or objective function) partitioning should be examined.

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.** 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.** 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 they 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 combinations 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 4.3 presents the extended secure solution for LP problems with arbitrarily partitioned objective function and constraints with respect to universal cases.

3.2.2. *Collaborative LP problem discussion*

After describing ways of partitioning data, we now discuss how to partition data for some typical collaborative LP problems. In this chapter, we start from 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, for example, in the case of a manufacturer that would like to evaluate several transportation options between its factories and raw material suppliers, with different transportation options that have different costs (leading to different objective functions).

After that, we propose privacy-preserving solutions for two or multiple parties who own arbitrarily partitioned data by assuming that the global constraints and objectives are the sum of all partitioned data. This happens very often. For example, a winemaking company that grows grapes in various locations, then sends them to different wineries for producing wine. Each vineyard can produce a limited amount of each type of grape; each winery has a requirement for how much of each type of grape it needs. These form the constraints of the LP problem. The goal of the company is to get the grapes from the vineyards to the wineries at the lowest cost (e.g. weight*distance). Obviously, the lowest cost solution would be to have the closest vineyard to each winery produce all the grapes needed by that winery. However, that violates the constraints on how much each vineyard can produce. The goal is to come up with a feasible solution that minimizes cost, where a feasible solution is one that meets all of the constraints imposed by the vineyards. If the winemaking company outsources the transportation to an external company, the collaborative LP problem will turn to the special case, where the external company owns the objective function while the winemaking company owns the constraints. They certainly do not want to reveal their data to the other party.

Now imagine that there are several winemaking companies, where the wineries have their own transportation department and several wineries in some locations. Due to the constraints on how much different vineyards can produce, they want to cooperate with each other on transporting grapes. This is a multiparty collaborative LP problem, where each party owns an objective function and several constraints (if they solve the transport problem independently) while the global objective function and constraints are the sum of all individuals (if they cooperate with each other). In case of this LP problem, each party does not want to reveal anything to any other parties, and each party eventually obtains an individual share of the solution to transport grapes. All the private data (objectives and constraints) are arbitrarily partitioned. In this chapter, we propose privacy-preserving schemes for this general and untrusted collaborative case. Besides the objective function and constraints, the global optimal solution in this case is also the sum of all the solutions of all the parties. The solution for each party means how many grapes should be delivered from each winery location to each vineyard for this party, whereas the global solution represents the total grapes distribution network for all the parties.

Note that the arbitrarily partitioned LP problems are not always formulated as the sum of the private shares (e.g. some parties may share the same constraint in an LP problem). Therefore, in this chapter, we propose privacy-preserving solutions for the LP problems in which the objective and constraints are the sum of the arbitrarily partitioned private shares (e.g. the above winemaking example). We will explore solutions for variant constraints and objectives in our future work.

3.3. A transformation approach for LP problems

A possible solution is to transform the vector space by applying a linear transformation. This idea was first proposed by Du to solve systems of linear equations (Du and Atallah, 2001). Du later extended this idea to solve the two-party LP problem (Du, 2001). Later, Vaidya (2009) proposed a secure transformation-based approach for two-party privacy preserving LP problems in which one party holds the objective function and the other party holds the constraints, detailed as below. As noted before, assume that you want to solve the problem:

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

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 $x \geq 0$, where Q should be a monomial matrix (pointed out by Bednarz *et al.*, 2009). Let $M' = MQ$, $y = Q^{-1}x$, and $c'^T = c^T Q$. We now have a

new LP problem:

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

Following the methodology of Du (2001), 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,

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

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 partitioned LP problems.

4. Secure Two-Party Transformation

In this section, we first introduce the fundamental encryption method that we are about to use, and then design the two-party secure transformation protocol in two different cases — on one hand, the constraints matrix and objective function are owned by different parties (Vaidya, 2009); on the other hand, both of them are split and shared by both parties.

4.1. Fundamental encryption

At the heart, transformations require matrix multiplication, which requires numerous scalar product computations. Thus, multiplying a $m \times n$ matrix with a $n \times n$ matrix simply requires mn scalar products. Goethals *et al.* (2004) proposed a simple and provably secure method to compute the scalar product using Homomorphic Encryption, which we now briefly describe. The problem is defined as follows: P_1 has an n-dimensional vector $\vec{X}$ while P_2 has an 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

a 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 (1984), the Benaloh cryptosystem (1986), the Naccache–Stern cryptosystem (1998), the Paillier cryptosystem (1999), and the Okamoto–Uchiyama cryptosystem (1998). Homomorphic Encryption is a semantically-secure public-key encryption which, in addition to standard guarantees, has the additional property that given any two encrypted messages $E(A)$ and $E(B)$, there exists an encryption $E(A*B)$ such that $E(A)*E(B) = E(A*B)$, where $*$ is either addition or multiplication (in some abelian group). The cryptosystems mentioned above are additively homomorphic (thus the operation $*$ denotes addition). 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 in Algorithm 1.

Algorithm 1 Secure scalar product

Require: P_1 has input vector $\vec{X} = \{x_1, \ldots, x_n\}$
Require: P_2 has input vector $\vec{Y} = \{y_1, \ldots, y_n\}$
Ensure: 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 $r_A = Dec_{sk}(w') = \vec{X} \cdot \vec{Y} - r_B$.

4.2. Two-party secure transformation without partitioning (Vaidya, 2009)

As discussed earlier, the specific problem we look at in this section consists of a special case of arbitrary partitioning, where one party owns the objective function while the other party owns the constraints. As described in Section 3.2.2, this may happen, for example, in the case of a manufacturer who would like to evaluate a transportation option for shipping its products to different locations, which is provided by an external company. Here, it would know the constraints, while the transporter would know the objective function.

It is quite easy to extend the encryption protocol to work for matrix multiplication and for shared matrices. According to the distributive law of matrix algebra, $M(Q_1 + Q_2) = MQ_1 + MQ_2$ and $C^T(Q_1 + Q_2) = C^TQ_1 + C^TQ_2$. P_2 first computes the encrypted form of MQ_2 and sends this to P_1 along with the encrypted form of M and Q_2. Using the properties of homomorphic encryption, P_1 can now compute MQ_1 and C^TQ_1 in encrypted form, and then compute the encrypted form of $MQ_1 + MQ_2$ and $C^T(Q_1 + Q_2)$. P_1 then sends them back to P_2 who decrypts to finally get MQ and C^TQ. The detailed algorithm is given in Algorithm 2.

Algorithm 2 Secure two-party transformation without partitioning

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 $V'_i = \prod_{k=1}^{n} Q_2'^{C^T_k} * (C^TQ_1)'_i$
12: P_1 sends S' and V' to P_2
13: P_2 computes $S = Dec_{sk}(S') = M(Q_1 + Q_2)$ and $V = C^T(Q_1 + Q_2)$

P_1 owns the objective function C^TX and its transformation matrix Q_1, whereas P_2 holds the constraints matrix M and its transformation matrix Q_2. We can analyze the security/privacy of Algorithm 2 by looking at the data held, received, computed, and decrypted by each party, as shown in Table 1.

(1) P_1 holds Q_1 and C^T while P_2 holds Q_2 and M. All of them are private components for themselves.

Table 1: Data in Algorithm 2.

	P_1	P_2
Hold	Q_1, C^T	(sk, pk), Q_2, M
Receive	$(MQ_2)'$, Q_2', M', pk	S', V'
Compute	S', $(C^TQ_1)'$, V'	$(MQ_2)'$, Q_2', M'
Decrypt		$M(Q_1+Q_2)$, $C^T(Q_1+Q_2)$

(2) P_1 receives $(MQ_2)'$, Q_2' and M' from P_2, but P_1 cannot decrypt them with a public key pk; P_1 computes(encrypts) S', $(C^TQ_1)'$ and V' using pk, but nothing can be learned from this step.

(3) P_2 decrypts S' and V' and obtains $M(Q_1+Q_2)$ and $C^T(Q_1+Q_2)$. Even if P_2 owns M, it cannot compute $Q = Q_1 + Q_2$ and C^T.

Finally, P_2 solves the new LP problem and shares the new solution with P_1, but P_2 still does not know the original solution of P_1. Hence, Algorithm 2 is secure.

4.3. Two-party secure transformation with arbitrary partitioning

In case that both the objective function C^T and the constraint matrix M of an LP problem are arbitrarily partitioned to two shares, where $C^T = C_1^T + C_2^T$ and $M = M_1 + M_2$ (note that C_1^T, C_2^T and C^T have the same size; M_1, M_2, and M have the same size). To securely solve such problem, we extend Algorithm 2 to Algorithm 3.

Without loss of generality, in Algorithm 3, P_1 solves the LP problem and it eventually obtains C^TQ and MQ after decryption, where $Q = Q_1Q_2$. Specifically, P_1 first sends its transformed and encrypted matrix/vector and the public key to P_2, and receives the encrypted and transformed overall constraints matrix from P_2. Then, P_1 computes the ciphertext of the transformed matrix/vector. Finally, it decrypts the transformed matrix $(MQ_2Q_1)'$ and vector $(C^TQ_2Q_1)'$ as well as solve the transformed LP problem.

After solving the transformed problem to get the optimal solution $y*$, P_1 and P_2 jointly reconstruct $x* = Q_2Q_1y*$. Bednarz *et al.* (2009) 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 problems. First, $x*$ is partitioned and each share of the solution is kept private, thus no solution can be used to verify the permutations. Second,

Algorithm 3 Secure two-party transformation with arbitrary partitioning

Require: P_1 has the $n \times n$ matrix Q_1, the share of objective function $C_1^T X$ and the $m \times n$ matrix M_1

Require: P_2 has the $n \times n$ matrix Q_2, the share of objective function $C_2^T X$ and the $m \times n$ matrix M_2

{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 $m \times n$ matrix: $M_1' = Enc_{pk}(M_1)$ and the n-dimensional vector: $C_1' = Enc_{pk}(C_1)$
3: P_1 sends pk, M_1' and C_1' to P_2
4: P_2 computes $M_2' = Enc_{pk}(M_2)$, $C_2' = Enc_{pk}(C_2)$, $M' = M_1' * M_2' = Enc_{pk}(M_1 + M_2)$, $C' = Enc_{pk}(C_1) * Enc_{pk}(C_2)$ and $Q_2' = Enc_{pk}(Q_2)$. ($*$ represents the multiplication operation for each element 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 $(MQ_2)'_{ij} = \prod_{k=1}^{n} (M_{ik})'^{Q_{2kj}}$
7: **for** each row i of C' and each column j of Q_2 **do**
8: P_2 computes $(C^T Q_2)'_{ij} = \prod_{k=1}^{n} (C_{ik})'^{Q_{2kj}}$
9: P_2 sends $(MQ_2)'$ and $(C^T Q_2)'$ back to P_1
10: **for** each row i of $(MQ_2)'$ and each column j of Q_1 **do**
11: P_1 computes $(MQ_2Q_1)'_{ij} = \prod_{k=1}^{n} ((MQ_2)_{ik})'^{Q_{1kj}}$
12: **for** each row i of $(C^T Q_2)'$ and each column j of Q_1 **do**
13: P_1 computes $(C^T Q_2 Q_1)'_{ij} = \prod_{k=1}^{n} ((C^T Q_2)_{ik})'^{Q_{1kj}}$
14: P_1 decrypts $(MQ_2Q_1)'$ and $(C^T Q_2 Q_1)'$ with its private key sk

C^T is partitioned and each share of C^T can be permuted before transformation as well. Since C^T is unknown to P_2, P_2 cannot compute Q based on an unknown vector C^T and the transformed objective vector $C^T Q$. We discuss the security of Algorithm 3 in Section 4.3.1.

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), Algorithm 3 does not violate the fairness in the secure computation.

4.3.1. *Security analysis*

Theorem 1. *P_1 learns only MQ_2Q_1 and $C^T Q_2 Q_1$, and P_2 learns nothing in Algorithm 3.*

Table 2: Each party's messages in Algorithm 3.

	P_1	P_2
Hold	Q_1, (sk, pk), M_1, C_1	Q_2, M_2, C_2
Receive	$(MQ_2Q_1)'$ and $(C^TQ_2Q_1)'$	M_1', pk
Encrypt or Compute	$(MQ_2Q_1)'$, $(C^TQ_2Q_1)'$, M_1', C_1'	$C', M', (MQ_2)'$
Decrypt	MQ_2Q_1 and $C^TQ_2Q_1$	

Proof. Before showing what P_1 and P_2 can learn in each step of Algorithm 3, we first look at different messages (e.g. matrices, vectors) each party can obtain in Table 2. 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', M, and (MQ_2) using pk, but nothing can be learned from these steps, since 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 Algorithm 3, on one hand, P_1 can only obtain $(M_1 + M_2)Q_2Q_1$ and $(C_1 + C_2)^TQ_2Q_1$. It's impossible to calculate M_2, C_2, and Q_2 with known matrices M_1, Q_1, and vector C_1. Also, the probability of guessing such private numbers is fairly low (note that in the earlier discussion, the adversary does not have any disclosed solutions and objective vectors to verify and determine Q using Bednarz *et al*. (2009)'s attack scenario). Specifically, even if P_1 can obtain the $m \times n$ matrix $(M_1 + M_2) \times Q_2$ and n-dimensional vector $(C_1 + C_2)^TQ_2$ from P_2 by decrypting the ciphertexts obtained from P_2, it is secure because P_1 does not know any factor of the product of matrix/vector multiplication (none of M, C, and Q_2 can be inferred). 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 a public key encryption for M_1, it is also impossible for P_2 to discover M_1 or Q_1 from the given matrices. Therefore, Algorithm 3 is secure. □

5. Secure Multi-Party Transformation

While Algorithm 3 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 extend the algorithm to multiple parties (more than two) with arbitrary partitioning.

Table 3: Each party's matrices in a multiparty LP (arbitrary partitioning) after decryption.

P_1	P_2	$\cdots$	P_l
$M_1Q_1Q_2\cdots Q_l + \sum_{j=2}^{l} R_{(1,j)}$	$-R_{(1,2)}$	$\cdots$	$-R_{(1,l)}$
$-R_{(2,1)}$	$M_2Q_1Q_2\cdots Q_l + \sum_{j=1,j\neq 2}^{l} R_{(2,j)}$	$\cdots$	$-R_{(2,l)}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$-R_{(l,1)}$	$-R_{(l,2)}$	$\cdots$	$M_1Q_1Q_2\cdots Q_l + \sum_{j=1}^{l-1} R_{(l,j)}$

5.1. Securing multiparty transformation

In multiparty 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/vectors for each encryption. Suppose that we are securing an l-party LP problem, each party P_i first distributes its public key pk_i to all the other parties. Then, P_i encrypts and transforms its matrix share M_i with its privately held monomial matrix Q_i and public key pk_i: $Enc_{pki}(M_iQ_i)$. Later, all parties jointly transform party P_i's matrix M_i to obtain $Enc_{pki}(M_iQ_1\cdots Q_l + R)$ where R is an encrypted random matrix co-held by all parties. Finally, each party P_i decrypts its transformed matrix with its private key sk_i, and all the parties securely sum the transformed matrix $MQ = (M_1 + +M_l)Q_1\cdots Q_l$. Table 3 shows each partys 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 partys objector vectors.

Then, Algorithm 4 presents the details of the secure multiparty transformation.

5.2. Security analysis

We have analyzed the security of Algorithm 3 in Section 5. Similarly, we can analyze all the messages in Algorithm 4.

Algorithm 4 Secure multi-party transformation with arbitrary partitioning

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

1: **for** ith party $(i = 1, 2, 3, \ldots, l)$ **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 $(M_i Q)' = Enc_{pk_i}(M_i Q_i)$ and $(C_i^T Q)' = Enc_{pk_i}(C_i^T Q_i)$
4: P_i sends $(M_i Q)'$ and $(C_i^T Q)'$ to the next party
5: **for** jth party $P_j (j = 1, 2, \ldots, l$ and $j \neq i)$ **do**
6: P_j generates an $m \times n$ random matrix $R_{(i,j)}$ and encrypts it with $pk_i : R'_{(i,j)} = Enc_{pk_i}(R_{(i,j)})$
7: P_j generates an n-dimensional random matrix $S_{(i,j)}$ and encrypts it with $pk_i : S'_{(i,j)} = Enc_{pk_i}(S_{(i,j)})$
8: **for** each row a of $(M_i Q)'$ and each column b of Q_j **do**
9: P_j computes $(M_i Q)'_{(i,j)_{ab}} = \prod_{h=1}^{n} M_i Q_{ah}^{\prime(Q_j)_{hb}} * R_{(i,j)'_{ab}}$
10: **for** each entry a of $(C_i^T Q)'$ and each column b of Q_j **do**
11: P_j computes $(C_i^T Q)'_{(i,j)_{ab}} = \prod_{h=1}^{n} C_i^T Q_{ah}^{\prime(Q_j)_{hb}} * S_{(i,j)'_{ab}}$
12: P_j sends the updated $(M_i Q)'$ and $(C^T Q)'$ to the next party
13: Finally, the last party sends $(M_i Q)'$ and $(C^T Q)'$ back to party P_i
14: **for** ith party $(i = 1, 2, 3, \ldots, l)$ **do**
15: P_i decrypts $(M_i Q)'$ and $(C^T Q)'$ with its private key sk_i to obtain $M_i Q_1 Q_2 \cdots Q_l + \sum_{i=1, i\neq j}^{l} R_{(i,j)}$ and $C_i^T Q_1 Q_2 \cdots Q_l + \sum_{i=1, i\neq j}^{l} S_{(i,j)}$
16: P_i vertically subtracts all the random matrices/vectors (generated by itself), and obtains $M_i Q_1 Q_2 \cdots Q_l + \sum_{i=1, i\neq j}^{l} R_{(i,j)} - \sum_{j=1, j\neq i}^{l} R_{(j,i)}$, and $C_i^T Q_1 Q_2 \cdots Q_l + \sum_{i=1, i\neq j}^{l} S_{(i,j)} - \sum_{j=1, j\neq i}^{l} S_{(j,i)}$
17: All the parties sum the matrices to obtain: $MQ = (M_1 + \cdots + M_l) Q_1 \cdots Q_l$ and $C^T Q = (C_1 + \cdots + C_l)^T Q_1 \cdots Q_l$

Theorem 2. *Algorithm* 4 *ensures:* $\forall i \in [1, l]$, P_i *cannot learn* $\forall j \in [1, l]$ *and* $j \neq i$, M_j *and* Q_j.

Proof. Table 4 presents all the exchanged messages in Algorithm 4:

We consider two different categories of collaborative parties: one is an arbitrary party P_i while the other one is another arbitrary party P_j. We can learn the following from Algorithm 4, Tables 2, and 4:

1. P_i receives $\forall j = 1, 2, 3, \ldots, l$ and $j \neq i$, pk_j, $Enc_{pk_i}(M_i Q_1 \cdots Q_l) * \prod * R'_{(i,j)}$ and $Enc_{pk_j}(M_j Q_j Q_1 \cdots) * R'_{(j,1)} * **$ from other parties;
2. P_i can neither decrypt $Enc_{pk_j}(M_j Q_j Q_1 \cdots) * R'_{(j,1)} * **$ nor compute the random matrices and other parties' transformation matrices Q_j from its decrypted matrix $M_i Q_1 \cdots Q_l + \sum_{j=1, j\neq i}^{l} R_{(i,j)}$;

Table 4: Each party's messages in Algorithm 4.

	$P_i(i = 1, 2, \ldots, l)$	$P_j(j \neq i, j = 1, 2, \ldots, l)$
Hold	$Q_i, (pk_i, sk_i), M_i, C_i^T$	$Q_j, (pk_j, sk_j), M_j, C_j^T$
Receive	$pk_j, Enc_{pk_i}(M_i Q_1 \cdots Q_l) * \prod_{\forall j \neq i} * R'_{(i,j)}$, $Enc_{pk_j}(M_j Q_j Q_1 \cdots Q_i) * R'_{(j,1)} * **$	$pk_i, Enc_{pk_j}(M_j Q_1 \cdots Q_l) * \prod_{\forall i \neq j} * R'_{(j,i)}$, $Enc_{pk_i}(M_i Q_i Q_1 \cdots Q_i) * R'_{(i,1)} * **$
Encrypt	$Enc_{pk_j}(M_j Q_j Q_1 \cdots Q_i) * R'_{(j,1)} * * * R'_{(j,i)}$	$Enc_{pk_i}(M_i Q_i Q_1 \cdots Q_i) * R'_{(i,1)} * * * R'_{(i,j)}$
Decrypt	$M_i Q_1 \cdots Q_l + \sum_{j=1, j \neq i}^{l} R_{(i,j)}$	$M_j Q_1 \cdots Q_l + \sum_{i=1, i \neq j}^{l} R_{(j,i)}$
Sum	the sum of $M_i Q$ and random matrices	the sum of $M_j Q$ and random matrices

3. If substituting the constraint matrix share with the objective vector share, the communication remains secure for all the parties;
4. Algorithm 4 does not suffer from Bednarz *et al*. (2009)'s attack for the same reason as Algorithm 3;
5. If more parties are involved in this multiparty LP problem, the protocol is more secure due to an increasing number l which further minimizes the probabilities of guessing the correct matrices and vectors;
6. $l > 2$ in Algorithm 4.

Finally, each party provides a vertical sum of all the matrices and vectors in Table 3. This is secure to all the parties and no one can learn any other party's private share of the constraints, vectors or transformation matrix from the sum and the data they provide to other parties. □

Similar to the 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 the parties can jointly reconstruct the original optimal solution $\prod_{i=1}^{l} Q_i y*$.

6. Concluding Remarks

This chapter presents the privacy-preserving LP problem, and motivates several different data distributions possible in real life LP models. We present a practical transformation-based solution that can solve the special two-party case where one party owns the objective while the other owns the constraints. We also extend the above approach to tackle such issues in a more general case that arbitrary

partitioned constraints and objective function are shared by both parties. In addition to a two-party secure computation, we propose an approach to secure multi-party transformation on their matrices and vectors in distributed LP problems. In all our secure algorithms, each party owns a part of the collaborative components as its privacy, eventually acquires the collaborative LP solution, calculates the individual solution using a reverse transformation from the common solution and implements it during the cooperation. We consider the individual transformation matrix, share of the objective vectors and constraints matrices as the private information, and take advantage of our algorithm to preserve them in the transformation and communication.

However, several questions remain for the future. First, the security implied by the solution is still somewhat heuristic. 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 the others, what about other features. Can anything be inferred 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 the same as the case as for LP. However, for quadratic programming, the form of the constraints changes — therefore the transformation method must take this into account. Nevertheless, polynomial evaluation may still be used to perform the transformation. We intend to further explore all of these problems in the future. Finally, as the intersection of two fundamental fields: secure computation and optimization, privacy preserving LP can be applied to tackle the privacy concerns in many real world applications or systems, such as web search (Hong *et al*., 2009, 2011, 2012), data mining (Lu *et al.*, 2009, 2012a, 2012b, 2013, 2015; Vaidya *et al*., 2013), transportation (Rizzo *et al*., 2015), smart grid (Hong *et al*., 2016), and healthcare systems (Hong *et al*., 2007, 2008; Liu *et al*., 2008; Lu *et al*., 2007). We will explore such real world applications in our future work.

Acknowledgments

This work is partially supported by the National Science Foundation under Grants No. CNS-0746943 and CNS-1618221.

References

Atallah, M. J., Deshpande, V., Elmongui, H. G. and Schwarz, L. B. (2003). Secure supply-chain protocols. In *Proceedings of the 2003 IEEE International Conference on E-Commerce*, Newport Beach, California, June 24–27.

Awerbuch, B. and Azar, Y. (1994). Local optimization of global objectives: Competitive distributed deadlock resolution and resource allocation. In *IEEE Symposium on Foundations of Computer Science*, pp. 240–249.

Bartal, Y., Byers, J. W. and Raz, D. (2004). Fast, distributed approximation algorithms for positive linear programming with applications to flow control. *SIAM Journal on Computing*, 33(6), 1261–1279.

Bednarz, A., Bean, N. and Roughan, M. (2009). Hiccups on the road to privacy-preserving linear programming. In *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*, WPES '09, pp. 117–120, New York, NY, USA, 2009. ACM.

Ben-Or, M., Goldwasser, S. and Wigderson, A. (1998). 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.

Benaloh, J. and Tuinstra, D. (1994). 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. ACM Press, pp. 544–553.

Benaloh, J. C. (1986). Secret sharing homomorphisms: Keeping shares of a secret secret. In Odlyzko, A. ed., *Advances in Cryptography — CRYPTO86: Proceedings, Springer-Verlag, Lecture Notes in Computer Science*, p. 263.

Blum, M. and Goldwasser, S. (1984). An efficient probabilistic public-key encryption that hides all partial information. In *Blakely, R. editor, Advances in Cryptology — Crypto 84 Proceedings*, Springer-Verlag.

Canetti, R. (1992). Asynchronous secure computation. Technical Report 755, CS Department, Technion, 1992.

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

Catrina, O. and de Hoogh, S. (2010). Secure multiparty linear programming using fixed-point arithmetic. In *ESORICS*, pp. 134–150.

Chaum, D., Crépeau, C. and Damgard, I. (1988). Multiparty unconditionally secure protocols. In *STOC '88: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pp. 11–19, New York, NY, USA.

Dreier, J. and Kerschbaum, F. (2011). Practical privacy-preserving multiparty linear programming based on problem transformation. In *SocialCom/PASSAT*, pp. 916–924.

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

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

Goldreich, O., Micali, S. and Wigderson, A. (1987). 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*, pp. 218–229, New York. NY.

Goldwasser, S. and Levin, L. A. (1991). Fair computation of general functions in presence of immoral majority. In *CRYPTO '90*: *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pp. 77–93, London, UK.

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

Hong, Y. (2013). *Privacy-Preserving Collaborative Optimization*. PhD thesis, Rutgers University, Newark, NJ.

Hong, Y., Goel, S. and Liu, W. M. (2016). An efficient and privacy-preserving scheme for p2p energy exchange among smart microgrids. *International Journal of Energy Research*, 40(3), 313–331.

Hong, Y., He, X., Vaidya, J., Adam, N. R. and Atluri, V. (2009). Effective anonymization of query logs. In *CIKM*, pp. 1465–1468.

Hong, Y., Lu, S., Liu, Q., Wang, L. and Dssouli, R. (2007). A hierarchical approach to the specification of privacy preferences. In *Innovations in Information Technology, 4th International Conference on*, pp. 660–664. IEEE.

Hong, Y., Lu, S., Liu, Q., Wang, L. and Dssouli, R. (2008). Preserving privacy in e-health systems using hippocratic databases. In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pp. 692–697. IEEE.

Hong, Y. and Vaidya, J. (2014). An inference-proof approach to privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 8(1), 267–277.

Hong, Y., Vaidya, J. and Lu, H. (2011). Efficient distributed linear programming with limited disclosure. In *DBSec*, pp. 170–185.

Hong, Y., Vaidya, J. and Lu, H. (2011). Search engine query clustering using top-k search results. In *Web Intelligence*, pp. 112–119.

Hong, Y., Vaidya, J. and Lu, H. (2012). Secure and efficient distributed linear programming. *Journal of Computer Security*, 20(5), 583–634.

Hong, Y., Vaidya, J., Lu, H., Karras, P. and Goel, S. (2015). Collaborative search log sanitization: Toward differential privacy and boosted utility. *IEEE Transactions on Dependable and Secure Computing*, 12(5), 504–518.

Hong, Y., Vaidya, J., Lu, H. and Shafiq, B. (2011). Privacy-preserving tabu search for distributed graph coloring. In *SocialCom/PASSAT*, pp. 951–958.

Hong, Y., Vaidya, J., Lu, H. and Wang, L. (2014). Collaboratively solving the traveling salesman problem with limited disclosure. In *Data and Applications Security and Privacy XXVIII — 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014*, Vienna, Austria, pp. 179–194.

Hong, Y., Vaidya, J., Lu, H. and Wu, M. (2012). Differentially private search log sanitization with optimal output utility. In *EDBT*, pp. 50–61.

Hong, Y., Vaidya, J. and Wang, S. (2014). A survey of privacy-aware supply chain collaboration: From theory to applications. *Journal of Information Systems*, 28(1), 243–268.

Li, J. and Atallah, M. J. (2006). Secure and private collaborative linear programming. In *Proceedings of the 2nd International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1–8.

Li, W., Li, H. and Deng, C. (2013). Privacy-preserving horizontally partitioned linear programs with inequality constraints. *Optimization Letters*, 7(1), 137–144.

Liu, Q., Lu, S., Hong, Y., Wang, L. and Dssouli, R. (2008). Securing telehealth applications in a web-based e-health portal. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pp. 3–9. IEEE.

Lu, H., Hong, Y., Street, W. N., Wang, F. and Tong, H. (2012a). Overlapping clustering with sparseness constraints. In *ICDM Workshops*, pp. 486–494.

Lu, H., Hong, Y., Yang, Y., Duan, L. and Badar, N. (2015). Towards user-oriented RBAC model. *Journal of Computer Security*, 23(1), 107–129.

Lu, H., Hong, Y., Yang, Y., Fang, Y. and Duan, L. (2014). Dynamic workflow adjustment with security constraints. In *Data and Applications Security and Privacy XXVIII — 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14–16, 2014. Proceedings*, pp. 211–226.

Lu, H., Vaidya, J., Atluri, V. and Hong, Y. (2009). Extended boolean matrix decomposition. In *ICDM*, pp. 317–326.

Lu, H., Vaidya, J., Atluri, V. and Hong, Y. (2012b). Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Transactions on Dependable and Secure Computing*, 9(5), 655–669.

Lu, S., Hong, Y., Liu, Q., Wang, L. and Dssouli, R. (2007). Access control in e-health portal systems. In *Innovations in Information Technology, 2007. IIT'07. 4th International Conference on*, pp. 88–92. IEEE.

Mailler, R. and Lesser, V. (2004). Solving distributed constraint optimization problems using cooperative mediation. *AAMAS*, 01, 438–445.

Mangasarian, O. L. (2011).Privacy-preserving linear programming. *Optimization Letters*, 5(1), 165–172.

Mangasarian, O. L. (2012). Privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 6(3), 431–436.

Mielikainen, T. (2004). Privacy problems with anonymized transaction databases. In *Discovery Science: 7th International Conference Proceedings*, volume 3245 of *Lecture Notes in Computer Science*, pp. 219–229. Springer-Verlag.

Modi, P. J., Shen, W.-M., Tambe, M. and Yokoo, M. (2003). An asynchronous complete method for distributed constraint optimization. In *AAMAS '03: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent systems*, pp. 161–168, New York, NY, USA.

Naccache, D. and Stern, J. (1998). A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pp. 59–66.

Okamoto, T. and Uchiyama, S. (1998). A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology — Eurocrypt '98, LNCS 1403*, pp. 308–318.

Ostrovsky, R. and Yung, M. (1991). How to withstand mobile virus attacks (extended abstract). In *PODC '91: Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pp. 51–59, New York, NY, USA.

Paillier, P. (1999). Public key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — Eurocrypt '99 Proceedings, LNCS 1592*, pp. 223–238.

Papadimitriou, C. H. and Yannakakis, M. (1991). 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*, pp. 61–64, New York, NY, USA.

Papadimitriou, C. H. and Yannakakis, M. (1993). Linear programming without the matrix. In *STOC '93: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 121–129, New York, NY, USA.

Petcu, A. and Faltings, B. (2005a). An efficient constraint optimization method for large multiagent systems. In *AAMAS05 - LSMAS workshop*.

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

Rizzo, N., Sprissler, E., Hong, Y. and Goel, S. (2015). Privacy preserving driving style recognition. In *2015 IEEE International Conference on Connected Vehicles and Expo*.

Sakuma, J. and Kobayashi, S. (2007). A genetic algorithm for privacy preserving combinatorial optimization. *GECCO*, pp. 1372–1379.

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

Silaghi, M. C. and Mitra, D. (2004). Distributed constraint satisfaction and optimization with privacy enforcement. *IAT*, pp. 531–535.

Silaghi, M.-C. and Rajeshirke, V. (2004). 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.

Suzuki, K. and Yokoo, M. (2003). Secure generalized vickrey auction using homomorphic encryption. *Financial Cryptography*, p. 2742.

Turban, E., Rainer, R. K. and Potter, R. E. (2004). Introduction to information technology Chapter 2, 3rd edn., Hoboken: John Wiley and Sons.

Vaidya, J. (2009). Privacy-preserving linear programming. *SAC*, pp. 2002–2007.

Vaidya, J. (2009). 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*.

Vaidya, J., Shafiq, B., Basu, A. and Hong, Y. (2013). Differentially private naive bayes classification. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, volume 1, pp. 571–576.

Yao, A. C. (1986). How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pp. 162–167, Los Alamitos, CA, USA, 1986. IEEE, IEEE Computer Society.

Yokoo, M., Suzuki, K. and Hirayama, K. (2002). 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).