On the (Im)possibility of Privately Outsourcing Linear Programming

Peeter Laud
Cybernetica AS
peeter.laud@cyber.ee

Alisa Pankova
Cybernetica AS
Software Technologies and Applications Competence Centre
University of Tartu, Institute of Computer Science
alisa.pankova@cyber.ee

ABSTRACT

In this paper we study the security definitions and methods for transformation-based outsourcing of linear programming. The recent attacks have shown the deficiencies of existing security definitions; thus we propose a stronger, indistinguishability-based definition of security of problem transformations that is very similar to IND-CPA security of encryption systems. We will study the realizability of this definition for linear programming and find that barring radically new ideas, there cannot exist transformations that are secure information-theoretically or even computationally. We conclude that for solving linear programming problems in privacy-preserving manner, cryptographic methods for securely implementing Simplex or some other linear programming solving algorithm are the only viable approach.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Client/server*

Keywords

Cryptanalysis; Linear programming; Secure outsourcing

1. INTRODUCTION

In linear programming (LP), one seeks the optimal value of a linear function of several arguments, subject to linear constraints on these arguments. In the canonical form, a LP task is

$$\text{maximize } \vec{c}^{\mathrm{T}}\vec{x}, \text{ subject to } A\vec{x} \leq \vec{b}, \vec{x} \geq \vec{0}, \qquad (1)$$

where $A \in \mathbb{R}^{m\times n}$, $\vec{b} \in \mathbb{R}^m$, $\vec{c} \in \mathbb{R}^n$ (all vectors are column vectors), and the inequalities hold componentwise. The solution $\vec{x}_{\mathrm{opt}}$ is a vector of length n over real numbers. There exist algorithms for solving LP tasks that are efficient in theory and/or in practice. A large number of practical optimization problems can be cast as LP tasks either exactly or approximately. As the specifications of these problems may also have privacy constraints, there has been ample interest towards privacy-preserving methods for solving LP tasks.

Secure outsourcing is conceptually the simplest case of privacy-aware computation. In this case, there are two parties, the *client* and the *server*. The client has a task T which it would like to solve, but has insufficient computational resources for doing so. The server is powerful enough to solve the task T. The client wants to keep T private from the server. Thus the client and server engage in a protocol that results in the client learning the solution to T, but is computationally less demanding for the client than solving T himself. Such outsourcing protocols have been proposed for different tasks, e.g. sequence matching [2], database operations [11], cryptographic operations [12, 6], and some linear algebra tasks [1] including linear programming [8, 22]. Quite often, the outsourcing protocol consists of the client transforming the task T in some manner that hides its original description, the server solving the transformed task, and the client transforming the solution of the transformed task back to the solution of the original task. To reap the benefits of outsourcing, a good, problem-specific transformation is probably necessary [7].

In this paper, we consider the feasibility of outsourcing linear programming in privacy-preserving manner. This problem is tightly related to multiparty privacy-preserving LP, where the specification $(A, \vec{b}, \vec{c})$ of a LP task is distributed among several mutually distrusting parties. There is a decent number of proposed protocols for this task that are actually based on outsourcing as well — the clients execute a protocol that results in a transformed task, this is solved publicly, and another protocol transforms this solution back to the solution of the original LP task [9, 21, 18, 19, 4, 17, 13, 14].

This paper is motivated by some recent attacks against privacy-preserving multiparty and/or outsourcing LP solving methods [3, 15]. They show that, for linear programs given in canonical form and split horizontally between parties (each party provides some of the constraints of the system), the existing transformations can in general be undone by some of the parties, using the partial knowledge they have about the initial task. There exist other forms of LP tasks (see Sec. 2.1) and proposed transformations for them, and the existing attacks are not directly applicable to these transformations, but as all forms are equivalent, it may be possible to adapt the attacks.

CCSW'13, November 8, 2013, Berlin, Germany.

ACM 978-1-4503-2490-8/13/11 ...$15.00.
http://dx.doi.org/10.1145/2517488.2517490.

The outline of this paper is the following. In Sec. 2 we review LP-related notions and existing transformation techniques from related works. In Sec. 3 we study how to build LP transformations on top of solid cryptographic foundations. To this end, we propose a natural indistinguishability-based definition stating that any two LP tasks chosen by the adversary cannot be distinguishied by it after the transformation. In Sec. 4 we show that it is impossible to information-theoretically achieve this level of security at least with transformations from a large class that includes all transformations built on currently proposed techniques. We note that this does not yet rule out computational security, and in Sec. 5 we study a number of candidate secure transformations. It turns out that none of our proposals are secure. We generalize the reason why all these transformations fail, and then we *empirically* state another necessary property for secure transformations that excludes this way of failing. In Sec. 6 we see that this property is very strong and constrains the transformed tasks too much.We thus conclude that, barring any radically new ideas related to transforming the LP task to some different task, transformation-based techniques for privacy-preserving LP outsourcing and multi-party LP solving are unusable, and the only currently known feasible method is to use privacy-preserving building blocks in implementing existing LP solving algorithms [20, 16].

2. PRELIMINARIES

Throughout this paper, the upright upper case letters A denote matrices, and the lower case letters with vector signs $\vec{b}$ denote column vectors. Writing two matrices/vectors together without an operator $A\vec{b}$ denotes multiplication, while separating them with a whitespace and putting into parentheses $(A\ \vec{b})$ denotes column augmentation. By augmentation we mean attaching a column $\vec{b}$ to the matrix A from the right. This can be generalized to matrices: $(A\ B)$ denotes a matrix that contains all the columns of A followed by all the columns of B. Row augmentation is defined analogously. Since multiplication and augmentation may be used in the same expression at once, for clarity the augmentation is sometimes also denoted $(A|\vec{b})$, whereas the multiplication operation has higher priority.

2.1 Linear Programming

The canonical form (1) of LP is equivalent to its *standard form*

$$\text{maximize } \vec{c}^{\mathrm{T}}\vec{x},\ \text{subject to } A\vec{x} = \vec{b}, \vec{x} \geq \vec{0} \qquad (2)$$

and to its augmented form ($i \in \{1, \ldots, |\vec{x}|\}$):

$$\text{maximize } x_i,\ \text{subject to } A\vec{x} = \vec{b}, \vec{x} \geq \vec{0}\ . \qquad (3)$$

Indeed, the inequalities of the canonical form may be replaced with equalities by introducing slack variables. Each equality may be substituted with two inequalities of opposite directions. Given a linear program in its standard form, the objective function vector $\vec{c}$ may be included into the matrix, and an additional variable represent the corresponding linear combination. The augmented form is an instance of standard form where the i-th entry of $\vec{c}^{\mathrm{T}}$ is 1, and the rest are 0.

A *feasible solution* of a linear program is any vector $x_0 \in \mathbb{R}^n$ that satisfies its constraints. An *optimal solution* of a linear program is any feasible solution that maximizes the value of its objective function. The *feasible region* of a linear program is the set of all its feasible solutions. It is a polyhedron — the intersection of a finite number of hyperplanes and half-spaces. A feasible solution is *basic* if it is located in one of the vertices of that polyhedron.

2.2 Existing Types of Transformations

Let a linear programming task be given in its standard form (2). The main basic transformations from the related works are the following.

Multiplying from the left. The idea of multiplying A and $\vec{b}$ by a random invertible matrix P from the left was first introduced in [9]. Since P is invertible, all the solutions to the system, including the optimal solution, remain the same. This means that the transformation is not harmful for the correctness. However, the feasible region remains revealed.

Multiplying from the right. The idea of multiplying A and $\vec{b}$ by a random invertible matrix Q from the right was also proposed in [9]. This operation hides also the objective function vector $\vec{c}$. Unfortunately it changes the optimal solution if some external constraints of the form $B\vec{x} \geq \vec{b'}$ are present. In this case, the vector $\vec{b'}$ should also be modified according to the transformation, but that in fact reveals all the information about Q. Since in practice linear programs do require such constraints (in general of the form $\vec{x} \geq \vec{0}$), this solution is not sufficient.

Scaling and Permutation. There have been attempts to implement multiplication from the right without affecting the correctness [21], and finally it was noticed in [4] that in order to preserve the inequality $\vec{x} \geq \vec{0}$, the most general type of matrix by which we may multiply from the right is a positive monomial matrix (product of a positive diagonal matrix and a permutation matrix). This results in scaling and permuting the variables.

Shifting. In [8], the initial variable vector $\vec{x}$ is not only scaled, but also shifted. This is done by introducing special slack variables for each shifted variable.

Several of the transformations presented above are special cases of the following transformation for a LP task in the augmented form (3). Generate a random $n \times n$ positive diagonal matrix D and $n \times n$ permutation matrix Q (corresponding to the permutation $\sigma \in S_n$), where $n = |\vec{x}|$. Output $i' = \pi(i)$ and $(A'|\vec{b'}) = \mathsf{RREF}(ADQ|\vec{b})$, where $\mathsf{RREF}(M)$ is the *reduced row-echelon form (RREF)* of the matrix M. We call this transformation the *basic transformation* of a LP task.

Recall that the RREF is an invariant of matrices: we have $\mathsf{RREF}(M_1) = \mathsf{RREF}(M_2)$ for $m \times n$ matrices M_1 and M_2 iff there exists an invertible $m \times m$ matrix P, such that $M_1 = PM_2$. Also, for each $m \times n$ matrix M there exists an invertible $m \times m$ matrix P, such that $\mathsf{RREF}(M) = PM$. Thus the computation of RREF generalizes the multiplication of the system of equations with an invertible matrix, occurring in almost every transformation proposed so far.

On input $i', A', \vec{b'}$, the server finds an optimal solution $\vec{y}_{\mathrm{opt}}$ for the LP task "maximize $y_{i'}$, subject to $A'\vec{y} = \vec{b'}$, $\vec{y} \geq \vec{0}$". Upon learning $\vec{y}_{\mathrm{opt}}$, the client can recover an optimal solution to (3) by $\vec{x}_{\mathrm{opt}} = \vec{y}_{\mathrm{opt}}Q^{-1}D^{-1}$.

Note that the basic transformation does not employ shifting. We have made this choice because the shifting transformation, as used in [8], is easy to undo [15].

3. DESIRED SECURITY DEFINITION

The security definition that has been used in the previous works related to the transformation-based approach is the acceptable security. This notion was first used in [10]. A protocol achieves acceptable security if the only thing that the adversary can do is to reduce all the possible values of the secret data to some domain with the following properties:

1. The number of values in this domain is infinite, or the number of values in this domain is so large that a brute-force attack is computationally infeasible.

2. The range of the domain (the difference between the upper and lower bounds) is acceptable for the application.

Some works provide more detailed analysis [3, 8] that estimates the probability that the adversary guesses some secret value. The leakage quantification analysis [8] is a compositional method for estimating the adversary's ability to make the correct guess when assisted by certain public information. However, even this analysis is still not formal enough and is related to the same acceptable security definition. The informal definitions allow the existence of some attacks that may be yet unknown, but may turn out to be efficient. For example, some vulnerabilities against settings that were assumed to be secure have been found [15]. Additionally, to argue about the security of complex protocols that use privacy-preserving LP transformations as a subprotocol, a more standard security definition for the LP transformation is necessary.

We now give the necessary notions to formally define a problem transformation and its security. Let $\mathfrak{T} \subseteq \{0,1\}^*$ be the set of all possible tasks and $\mathfrak{S} \subseteq \{0,1\}^*$ the set of all possible solutions. For $T \in \mathfrak{T}$ and $S \in \mathfrak{S}$ let $T \vDash S$ denote that S is a solution for T. A *problem transformation* is a pair of functions $\mathcal{F} : \mathfrak{T} \times \{0,1\}^* \to \mathfrak{T} \times \{0,1\}^*$ and $\mathcal{G} : \mathfrak{S} \times \{0,1\}^* \to \mathfrak{S}$. Both $\mathcal{F}$ and $\mathcal{G}$ must work in time polynomial to the length of their first argument. The pair $(\mathcal{F}, \mathcal{G})$ is a *correct problem transformation* if

$$\forall T, r, T', S', s : \\ \big((T', s) = \mathcal{F}(T; r) \ \wedge \ T' \vDash S'\big) \ \Rightarrow \ T \vDash \mathcal{G}(S', s) \ .$$

This implication shows the intended use of $\mathcal{F}$ and $\mathcal{G}$. To transform a task T, the mapping $\mathcal{F}$ uses randomness r, producing a transformed task T' and some state s for the mapping $\mathcal{G}$ that transforms a solution S' to T' back into a solution of the original task T. We write $\mathcal{F}(T)$ for a randomized function that first samples r and then runs $\mathcal{F}(T; r)$.

Note that we have not defined the transformation in the most general manner possible. Namely, we require that the result of transforming a task from the set $\mathfrak{T}$ is again a task from $\mathfrak{T}$. This corresponds to our goal that a transformed linear program is a linear program again.

The transformation $\mathcal{F}$ is intended to hide the important details of a task T. The meaning of hiding has been recently investigated by Bellare et al. [5] in the context of garbling circuits [23]. It is possible that it is unimportant and/or too expensive to hide certain details of T. It is also possible that certain tasks are inherently unsuitable for hiding. In the context of linear programming, we most probably do not want to hide the size of the task, because we would like to avoid padding all tasks to some maximum size. Also, some tasks may be ill-specified and thus unsuitable for transformation. For example, we may require the constraint matrix to have full rank.

Both the public details and suitability for hiding are captured by the notion of *side information function* $\Phi : \mathfrak{T} \to \{0,1\}^*$ that, again, must be polynomial-time computable. When transforming a task T, we do not try to hide the information in $\Phi(T)$. If T should not be transformed at all, then we set $\Phi(T) = T$. We can now state a rather standard, indistinguishability-based definition of privacy. Recall that a function $\alpha : \mathbb{N} \to \mathbb{R}$ is *negligible* if $\forall c \exists m \forall n \geq m : \alpha(n) < 1/n^c$.

Definition 1. *A transformation $(\mathcal{F}, \mathcal{G})$ for $\mathfrak{T}$ is* Φ-private *if the advantage of any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a negligible function of η in the following experiment $\mathbf{Exp}^{\mathfrak{T},\Phi}_{\mathcal{A}}$:*

$$\left[\begin{array}{l} (T_0, T_1, s) \leftarrow \mathcal{A}_1(\eta) \\ \textbf{\textit{if}}\ |T_0| \neq \eta \vee |T_1| \neq \eta \vee \Phi(T_0) \neq \Phi(T_1) \\ \quad \textbf{\textit{return}}\ \bot \\ b \xleftarrow{\$} \{0,1\} \\ (T', \_) \leftarrow \mathcal{F}(T_b) \\ b' \leftarrow \mathcal{A}_2(T', s) \\ \textbf{\textit{return}}\ (b \stackrel{?}{=} b') \end{array}\right.$$

where the advantage of the adversary is $1/2$ less the probability of the experiment returning true.

If $\mathfrak{T}$ is the set of all linear programming tasks "maximize $\vec{c}^{\mathrm{T}}\vec{x}$, subject to $A\vec{x} = \vec{b}$, $\vec{x} \geq \vec{0}$", determined by $A \in \mathbb{R}^{m \times n}$, $\vec{b} \in \mathbb{R}^m$ and $\vec{c} \in \mathbb{R}^n$, then we will in the following take $\Phi(A, \vec{b}, \vec{c}) = (m, n, \vec{x}_{\mathrm{opt}}, \mathsf{bb}(A, \vec{b}))$, where $\vec{x}_{\mathrm{opt}}$ is the unique optimal solution for the linear programming task and $\mathsf{bb}(A, \vec{b})$ is the bounding box for the polyhedron $A\vec{x} = \vec{b}$, $\vec{x} \geq \vec{0}$. If the task is infeasible, unbounded, or has several different optimal solutions, then we consider the task unsuitable for transformation and take $\Phi(A, \vec{b}, \vec{c}) = (A, \vec{b}, \vec{c})$. This choice of Φ is at least as permissive as the privacy goals for previously proposed transformations (note that these goals have often been implicit in the papers where these transformations were presented).

Quite clearly, the transformations described in Sec. 2.2, in particular the basic transformation, do not satisfy this definition. If $\mathcal{F}$ is the basic transformation then the adversary picks two LP tasks T_0 and T_1 with n variables, m equality constraints and the same optimal solution, such that their feasible regions P_0 and P_1 both have the unit hypercube with opposite corners at $(0, \ldots, 0)$ and $(1, \ldots, 1)$ as the bounding box, but a different number of vertices with coordinates $(0, \ldots, 0, 1, 0, \ldots, 0)$. Given the transformed task T', the adversary will find its bounding box (by solving a number of LP tasks with the constraints of T' and either maximizing or minimizing a single variable), scale it to the unit hypercube, and count the vertices with coordinates $(0, \ldots, 0, 1, 0, \ldots, 0)$. This count is not changed by the basic transformation.

4. NO PERFECT SECRECY

A transformation $(\mathcal{F}, \mathcal{G})$ provides *perfect secrecy* (or *information-theoretic secrecy*) if the advantage of any adversary in the experiment described in Def. 1 is zero. It is definitely possible to define a transformation that provides perfect secrecy with respect to this definition. Such a transformation

$\mathcal{F}$ would solve the LP task and then output a randomly selected (or even a constant) LP task that has the same optimal solution. But obviously, we are not looking for such transformations because the goal of the entire outsourcing approach is to make the client not pay the price of solving the original problem.

In this section we prove that any perfectly secure transformation with the properties listed below cannot be computationally much simpler than solving the LP task. W.l.o.g. we assume that both the inputs and the outputs of $\mathcal{F}$ are linear programs in the canonical form (1). The properties are the following:

1. The optimal solution $\vec{y}_{\text{opt}}$ to the transformed linear program $\mathcal{F}(A, \vec{b}, \vec{c}; r)$ only depends on r and $\Phi(A, \vec{b}, \vec{c})$.
2. The mapping $\mathcal{F}(\cdot; r)$ is continuous with respect to the optimal solutions of the initial and transformed problems. This means that for each $\varepsilon > 0$ there exists $\delta > 0$, such that if $(A^\circ, \vec{b^\circ}, \vec{c^\circ})$ and $(A^\bullet, \vec{b^\bullet}, \vec{c^\bullet})$ are two LP tasks with n variables, m constraints and the optimal solutions satisfying $\|\vec{x^\circ}_{\text{opt}} - \vec{x^\bullet}_{\text{opt}}\| \leq \delta$, then the optimal solutions of the transformed tasks $\mathcal{F}(A^\circ, \vec{b^\circ}, \vec{c^\circ}; r)$ and $\mathcal{F}(A^\bullet, \vec{b^\bullet}, \vec{c^\bullet}; r)$ satisfy $\|\vec{y^\circ}_{\text{opt}} - \vec{y^\bullet}_{\text{opt}}\| \leq \varepsilon$.

We find the properties natural; they are satisfied by all transformations proposed so far in the literature and seem to be natural consequences of the proposed transformation techniques. For example, the result of scaling and shifting the polyhedron or permuting its variables depends only on the random matrices by which it is multiplied (the range for the random values may depend on the bounding box). For a transformation satisfying these properties, we show how to turn it into a LP solving algorithm with only little extra computational effort.

According to property 2, there exists a function $\Delta : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ (where $\mathbb{R}_{>0}$ denotes the set of positive real numbers), mapping ε to δ. W.l.o.g. we may assume that the function Δ is monotone and continuous.

For the simplicity of the exposition, we need a further technical property of $\mathcal{F}$. Let Q_n be the n-dimensional unit hypercube. For a polyhedron P bounded by a number of hyperplanes, let its *roundness* $\text{rd}(P)$ be the minimum distance between a vertex of P and a hyperplane of P that does not contain this vertex. For $\vec{v} \in Q_n$, let $\text{rd}_n^m(\vec{v}; r)$ be the roundness of the polyhedron determined by the constraints of the LP task $\mathcal{F}(A, \vec{b}, \vec{c}; r)$, where $\Phi(A, \vec{b}, \vec{c}) = (m, n, \vec{v}, Q_n)$. Note that due to property 1, the quantity $\text{rd}_n^m(\vec{v}; r)$ is well-defined. The required property of $\mathcal{F}$ is the following:

3. The function $\text{rd}_n^m(\vec{x}; r)$ is continuous (as a function of $\vec{x}$).

We show that the existence of a perfectly secure transformation with listed properties allows us to perform offline precomputations for a fixed dimension n and number of bounding hyperplanes m that afterwards allow us to solve an arbitrarily large proportion of interesting LP tasks of dimension $n-1$ with $m-2$ facets with an effort that is only marginally larger than applying $\mathcal{F}$ and $\mathcal{G}$. Let our interest in these tasks be described by a probability distribution $\mathcal{D}$; the tasks we want to solve are sampled from this distribution. We assume that the probability density function $\mathbf{p}$ of the optimal solutions of tasks sampled according to $\mathcal{D}$ is continuous. We also assume that as side information, we know the bounding boxes of the polyhedra defined by the constraints of these tasks; this assumption holds for large classes of problems occurring in practice. We will actually use an assumption equivalent to the last one due to the ease of scaling and shifting: the bounding boxes of all tasks sampled from $\mathcal{D}$ are Q_{n-1}.

For the precomputation, we first fix the randomness r. We construct a LP task T^{pre} with n variables and m bounding hyperplanes and the objective function selected in such a way that the optimal solution of T^{pre} is $\vec{x}_{\text{opt}}^{\text{pre}} = (1, \ldots, 1)^{\text{T}}$. We perform the transformation – $U^{\text{pre}} = \mathcal{F}(T^{\text{pre}}; r)$ – and solve the resulting task U^{pre}. Let the solution to U^{pre} be $\vec{y}_{\text{opt}}^{\text{pre}}$.

Let $q \in (0, 1)$ be the desired success probability of the LP solving algorithm. The online phase of LP depends on a constant $\varepsilon > 0$ that is selected according to q (but is independent of T^{pre}). Let us first explain the algorithm, and then show that there is a choice of ε that guarantees the success probability at least q. Denote $\delta = \Delta(\varepsilon)$.

Let $T \leftarrow \mathcal{D}$ be a LP task with $n-1$ variables and $m-2$ constraints. Let P be the polyhedron defined by these constraints; let the bounding box of P be Q_{n-1}. To solve T, we subject it to the following transformations.

1. Scale the polyhedron P down, with the scalar multiplier being δ. This corresponds to substituting each variable x_i in each constraint and in the objective function vector by $(1/\delta)x_i$. Let T_0 be the resulting task and P_0 the polyhedron defined by the scaled constraints. The bounding box of P_0 is the hypercube in $n-1$ dimensions with the side length δ.
2. Add the n-th dimension and build an oblique hyperprism from P_0, with the bases at hyperplanes $x_n = 0$ and $x_n = 1$, and the bounding box of the hyperprism being equal to Q_n. This modifies the system of constraints as follows:
   - Two new constraints, $x_n \geq 0$ and $x_n \leq 1$, are added corresponding to the bases of the hyperprism.
   - Each existing constraint $\sum_{i=1}^{n-1} a_i x_i \leq b$ is replaced with $\sum_{i=1}^{n-1} a_i(x_i + (1-\delta)x_n) \leq b$, corresponding to the sides of the hyperprism.

The result of these two transformations is shown in Fig. 1.

Let T' be the LP task where the resulting hyperprism P' is the set of feasible solutions, and where the objective function of T' is $\sum_{i=1}^{n-1} c_i x_i + C x_n$, where $\vec{c} = (c_1, \ldots, c_{n-1})$ is the objective function vector of T and C is a large constant. Hence the optimal solution $\vec{x'}_{\text{opt}}$ of T' is located on the base P_1 of the hyperprism, otherwise being "at the same vertex" as the optimal solution $\vec{x}_{\text{opt}}$ to T. In particular, $\vec{x'}_{\text{opt}}$ can easily be transformed back to $\vec{x}_{\text{opt}}$. Also note that $\|\vec{x'}_{\text{opt}} - \vec{x}_{\text{opt}}^{\text{pre}}\| \leq \delta$.

Let $(U', s) = \mathcal{F}(T'; r)$. If we could find the optimal solution $\vec{y'}_{\text{opt}}$ to U', then we could compute $\vec{x'}_{\text{opt}} = \mathcal{G}(\vec{y'}_{\text{opt}}, s)$ and find $\vec{x}_{\text{opt}}$ from it. Note that $\|\vec{y'}_{\text{opt}} - \vec{y}_{\text{opt}}^{\text{pre}}\| \leq \varepsilon$. Let $\bar{P}$ be the polyhedron defined by the constraints of U'. The point $\vec{y'}_{\text{opt}}$ is determined as the unique intersection point of a number of hyperplanes bounding $\bar{P}$. All these hyperplanes are at distance of at most ε from the point $\vec{y}_{\text{opt}}^{\text{pre}}$.

We now have to explain the choice of ε. We have selected it so small, that $\Pr[\text{rd}(\bar{P}) > 2\varepsilon] \geq q$, where the probability

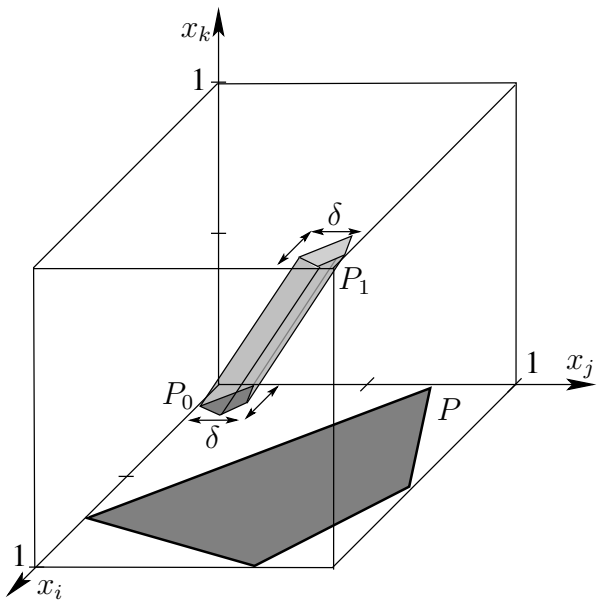

Figure 1: Results of preparatory transformations to task T

is taken over the choice of T (sampled from $\mathcal{D}$). Thanks to this choice, the following claim holds with probability at least q:

- If a hyperplane bounding $\bar{P}$ is at distance of at most ε from the point $\vec{y}^{\,\mathrm{pre}}_{\mathrm{opt}}$, then this hyperplane contains $\vec{y'}_{\mathrm{opt}}$.

Hence, to find $\vec{y'}_{\mathrm{opt}}$, we measure how far each hyperplane bounding $\bar{P}$ is from the point $\vec{y}^{\,\mathrm{pre}}_{\mathrm{opt}}$, and find the intersection point of these hyperplanes where the distance is at most ε. This amounts to solving a system of linear equations, which is much simpler than solving LP.

Existence of suitable ε.

Let $\mathsf{base}_\varepsilon \subseteq \mathbb{R}^n$ be the set of points $\vec{x}$, where $x_n = 1$ and $1 - \Delta(\varepsilon) \leq x_i \leq 1$ for all $i \in \{1, \ldots, n-1\}$. Let $\mathbf{p}'_\varepsilon$ be the probability density function of $\vec{x'}_{\mathrm{opt}} \in \mathsf{base}_\varepsilon$, where $\vec{x'}_{\mathrm{opt}}$ is the optimal solution to the LP task T', defined as before. The function $\mathbf{p}'_\varepsilon$ is continuous because it has been obtained via a continuous transformation from the continuous function $\mathbf{p}$. For a Boolean value b, let $[b]$ be 1 if b is true, and 0 if b is false. The probability of $\mathrm{rd}(\bar{P})$ being larger than 2ε is

$$PR(\varepsilon) = \int_{\mathsf{base}_\varepsilon} \mathbf{p}'_\varepsilon(\vec{x}) \cdot [\mathrm{rd}_n^m(\vec{x}; r) > 2\varepsilon]\, d\vec{x} \ .$$

We see that PR is a continuous function, because it has been constructed from continuous components in a manner that preserves continuity. As $PR(0) = 1$, there must exist some $\varepsilon > 0$, such that $PR(\varepsilon) \geq q$.

5. SOME INSECURE TRANSFORMATIONS

We have seen that perfect security is impractical. This does not yet rule out the existence of transformations with weaker security (computational), because we had to access the private randomness in order to obtain the optimal solution for the LP task.

In this section, we propose certain transformations, observe why these do not satisfy our privacy , and *empirically* derive a further necessary condition for the security of the transformation. In our opinion, this condition is a very natural one. The transformations satisfying this condition are then further explored in Sec. 6.

When aiming for computational security, the privacy has to follow from a plausible computational hardness assumption. In LP, we work with real numbers, hence the assumptions about discrete structures (e.g. the RSA assumption, various Diffie-Hellman assumptions) etc. are not directly applicable. Any assumption we base the security of the transformation on, will be a novel one. To reduce the novelty, we may take some known assumption on processes similar to transforming LP tasks producing indistinguishable distributions, and change it by stating that instead of elements of finite fields or groups, there are real numbers. We must be careful, though, because real numbers have extra structure (order) that finite fields lack.

The *Strong Secret Hiding Assumption (SSHA)* [1] may be suitable for such change. We refer to the original paper for its precise statement on the indistinguishability of certain distributions over matrices over finite fields. We only remark that if SSHA could be extended to matrices over $\mathbb{R}$, then it would allow us to hide a LP task by adding many extra columns (and rows) to the matrix A before applying the basic transformation. We must be careful, though, to not change the optimal solution of the task.

Let us now study different ways of performing this augmentation of A. Starting from the task (3), we will change it to "maximize w_j, subject to $A'\vec{w} = \vec{b'}$, $\vec{w} \geq 0$", to which we apply the basic transformation.

New variables with constraints.

We add a new variable z to the system, and fix its value with respect to existing variables: we also add a constraint $z = \vec{d}^{\mathrm{T}}\vec{x} + r$ to the system, where $\vec{d} \in \mathbb{R}^n$ and $r \in \mathbb{R}$. In this manner, we can add any number of variables $z_1, \ldots, z_k$ with constraints $z_j = \vec{d_j}^{\mathrm{T}}\vec{x} + r_j$. This corresponds to changing the task to "maximize x_i, subject to $\left(\begin{smallmatrix} A & 0 \\ D & -I \end{smallmatrix}\right) = \left(\begin{smallmatrix} \vec{b} \\ -\vec{r} \end{smallmatrix}\right)$, $\vec{x}, \vec{z} \geq \vec{0}$", where $D = (\vec{d_1} \ \cdots \ \vec{d_k})^{\mathrm{T}}$ and $\vec{r} = (r_1 \ \cdots \ r_k)^{\mathrm{T}}$.

The inequalities $\vec{z} \geq \vec{0}$ set new constraints also for the original variables $\vec{x}$. These must not change the optimal solution of the LP task. Also, we want to define the transformation in a manner that does not require it to start solving the original LP task. Hence we want that the newly introduced inequalities $z_j = \vec{d_j}^{\mathrm{T}}\vec{x} + r_j \geq 0$ are implied by the original inequalities $\vec{x} \geq \vec{0}$. This is possible only if all entries of $\vec{d_j}$ are non-negative and $r_j \geq 0$.

Why do we add the inequalities $\vec{z} \geq \vec{0}$ to the transformed LP task, instead of allowing these variables to be free? If we did so, we also would have to leave free the variables in $\vec{y}$ (after performing the basic transformation) corresponding to variables in $\vec{z}$. This means that in the transformed task, we can tell which of the variables $\vec{y}$ correspond to variables in $\vec{x}$ and which correspond to variables in $\vec{z}$. If we know which variables stem from $\vec{z}$, we can use Gaussian elimination to get rid of them. This leaves us with just the basic transformation being applied to the original task, which, as we saw before, is not sufficiently secure.

If we insist that all entries of D and $\vec{r}$ are non-negative, the variables from $\vec{x}$ and the variables from $\vec{z}$ can still be distinguished after the basic transformation. A variable z_j with the constraint $z_j = \vec{d_j}^{\mathrm{T}}\vec{x} + r_j$ can only be 0 if r_j is 0 and $x_k = 0$ for all $k \in \{1, \ldots, n\}$ where $\vec{d}_{jk} \neq 0$. If the actual transformation is such, that the values r_j are likely to be non-zero, then the adversary of Def. 1 has to pick the to-be-distinguished LP tasks so, that for each variable, there is a feasible solution where this variable equals 0. To distinguish variables from $\vec{x}$ and the variables from $\vec{z}$, the adversary minimizes each variable one by one (this amounts to solving a LP) and sees which ones have the minimal value 0. If the addends r_j are 0, but the matrix D is sufficiently dense then the variables from $\vec{z}$ have "smaller probability" of being 0 than the variables from $\vec{x}$. In this case, the adversary samples random vertices of the polyhedron $A'\vec{y} = \vec{b'}$ (by solving LP tasks with randomly chosen optimization directions, i.e randomly chosen objective functions) and records which variables are 0 with larger or smaller frequency. If the matrix D is sparse then the adversary picks the to-be-distinguished LP tasks so, that there are no affine relationships among small sets of initial variables $\vec{x}$. The transformation, however, introduces affine relationships among a variable from $\vec{z}$ and small sets of variables from $\vec{x}$. Such relationships are straightforwardly detected [15] and based on them, variables from $\vec{x}$ and variables from $\vec{z}$ can be distinguished.

Hence this way of augmenting the LP task is not sufficient to achieve privacy, at least when applied alone.

New variables without constraints.

We add k new variables $\vec{z}$ to the system (3) with m constraints and n variables, without adding new constraints. In this case, the original system of equations $A\vec{x} = \vec{b}$ must be modified to include $\vec{z}$, otherwise the variables corresponding to the ones in $\vec{z}$ can be recognized after the basic transformation and easily removed from the system.

We replace the original system of equations with $A\vec{x} + C\vec{z} = \vec{b}$, where $C = AV$, where $V \in \mathbb{R}^{n \times k}$ is a random matrix with non-negative entries. We also add the constraints $\vec{z} \geq \vec{0}$. Given an optimal solution $(\vec{x}_0, \vec{z}_0)$ of the modified task, we recover the optimal solution $\vec{x}_{\mathrm{opt}}$ of the original task as $\vec{x}_{\mathrm{opt}} = \vec{x}_0 + V\vec{z}_0$.

It is easy to see that $\vec{x'} = \vec{x}_0 + V\vec{z}_0$ is indeed the optimal solution to the original task. First, it satisfies the system of equations $A\vec{x} = \vec{b}$. Second, the value of the objective function x_i for $\vec{x'}$ is at least as good as its value in $\vec{x}_{\mathrm{opt}}$, because $(\vec{x}_{\mathrm{opt}}, \vec{0})$ is one of the solutions of the modified task and the i-th component of the vector $V\vec{z}_0$ is non-negative. Finally, all components of $\vec{x'}$ are non-negative because both $\vec{x}_0$ and $V\vec{z}_0$ are non-negative.

If we could distinguish the variables in $\vec{x}$ from the variables in $\vec{z}$ after the basic transformation, then we could remove the variables in $\vec{z}$ by setting them equal to 0, and obtain the original LP task with basic transformation. Unfortunately, we can indeed distinguish them. There are no upper bounds for variables in $\vec{z}$. If the adversary picks the to-be-distinguished LP tasks so, that their feasible solution sets are bounded, then it has to maximize each variable one by one to distinguish variables in $\vec{x}$ from those in $\vec{z}$.

The two augmentations could both be applied to the original LP task before the basic transformation. It is easy to see that the order of application does not matter here. Unfortunately, the application of both of them does not increase the security of the transformation — the variables introduced by both augmentations can still be located and removed.

Splitting the variables.

In previous transformations, each variable of the original LP task gave us one variable in the transformed task, while the transformation introduced new variables. We can also consider transformations where each variable x_j in the original task gives several variables $z_{j1}, \ldots, z_{jk}$ of the transformed task. In this case, the transformation picks non-negative $r_{j1}, \ldots, r_{jk}$ and replaces x_j with $r_{j1}z_{j1} + \cdots + r_{jk}z_{jk}$ in all equations of the original task (3). This replacement can be made for some, or all variables in the original system. After this replacement, we again apply the basic transformation to the system.

Unfortunately, the splitting of the variables can be undone, at least partially, even after the basic transformation, even if other transformations described above have also been applied. The adversary will pick the original tasks T_0, T_1 so that the set of feasible solutions is bounded. Let x_j be one of the original variables, with the upper bound $x_j^{\max}$. After the replacement, the variable z_{jt} is upper-bounded by $z_{jt}^{\max} = x_j^{\max}/r_{jt}$ (recall that the upper bounds can easily be found). A pair of variables z_{jt}, z_{ju} satisfies the inequality $r_{jt}z_{jt} + r_{ju}z_{ju} \leq x_j^{\max}$ or $z_{ju}^{\max} \cdot z_{jt} + z_{jt}^{\max} \cdot z_{ju} \leq z_{jt}^{\max} z_{ju}^{\max}$.

The basic transformation permutes the variables. To undo the splitting of variables, we have to detect whether two variables y_{j1} and y_{j2} could have resulted from the splitting of the same variable x_j. We just saw that a necessary condition for this is, that all feasible solutions of the transformed task satisfy the condition $y_{j2}^{\max} \cdot y_{j1} + y_{j1}^{\max} \cdot y_{j2} \leq y_{j1}^{\max} y_{j2}^{\max}$. In other words, this inequality must be redundant with respect to the constraints of the transformed task. The redundancy of an inequality constraint is easy to check (it corresponds to a LP task).

While this analysis may not be sufficient to completely identify which variables in $\vec{y}$ correspond to the same variable in $\vec{x}$ (we may get false positives due to certain pairs of original variables x_{j_1} and x_{j_2} also satisfying the inequality $x_{j_2}^{\max} \cdot x_{j_1} + x_{j_1}^{\max} \cdot x_{j_2} \leq x_{j_1}^{\max} x_{j_2}^{\max}$), it is sufficient to distinguish two LP tasks selected by the adversary.

A further condition.

We see that all our attempts so far have failed because in the transformed task, different variables had different roles. These roles could be determined and the variables eliminated. Thus we set an extra requirement that in the transformed task (in the polyhedron corresponding to this task), all variables "look the same". Besides the variables, we want the same condition to hold for (small) sets of variables, as the failure of our last attempt in augmenting A showed.

We need the following notions to formally define our requirement. Let $\vec{e}_i^{\,k} = (0, \ldots, 0, 1, 0, \ldots, 0)$ be the i-th *unit vector* in the k-dimensional space $\mathbb{R}^k$ (the length of $\vec{e}_i^{\,k}$ is k and the only 1 is on i-th position). For $I \subseteq \mathbb{N}$ and $i \in I$ let $\mathsf{idx}_I\, i$ be the *index* of i in the ordered set I, meaning that I has exactly $\mathsf{idx}_I\, i$ elements less than or equal to i. For $I \subseteq \{1, \ldots, k\}$, $|I| = n$ let $\pi_I^k : \mathbb{R}^k \to \mathbb{R}^n$ be the *projection* to dimensions in I. It is a linear mapping defined by $\pi_I^k(\vec{e}_i^{\,k}) = \vec{e}_{\mathsf{idx}_I\, i}^{\,n}$ if $i \in I$, and $\pi_I^k(\vec{e}_i^{\,k}) = 0$ otherwise. For a permutation $\sigma \in S_n$ let $\hat{\sigma} : \mathbb{R}^n \to \mathbb{R}^n$ be the permutation of

dimensions given by σ, i.e. $\hat{\sigma}$ is the linear mapping defined by $\hat{\sigma}(\vec{e}_i^n) = \vec{e}_{\sigma(i)}^n$.

DEFINITION 2. *Let $t \in \mathbb{N}$. A set of points $X \subseteq \mathbb{R}^k$ is* t-symmetric*, if for any $I, I' \subseteq \{1, \ldots, k\}$, $|I| = |I'| = t$, and $\sigma \in S_t$ we have $\pi_I^k(X) = \hat{\sigma}(\pi_{I'}^k(X))$.*

There is also a computational analogue to this definition.

DEFINITION 3. *Let $t, k : \mathbb{N} \to \mathbb{N}$. A family of probability distributions over sets of points $\{\mathcal{X}_n\}_{n\in\mathbb{N}}$, where each element $X_n \in \mathrm{supp}(\mathcal{X}_n) \subseteq \mathbb{R}^{k(n)}$ has a polynomial-size description, is* computationally t-symmetric*, if for any probabilistic polynomial-time (in n) algorithm $\mathcal{A}$, the following probability is negligible:*

$$\Pr[x \in \pi_I^{k(n)}(X) \backslash \hat{\sigma}(\pi_{I'}^{k(n)}(X)) \mid X \leftarrow \mathcal{X}_n, (x, I, I', \sigma) \leftarrow \mathcal{A}(X)],$$

where $x \in \mathbb{R}^{t(n)}$, $I, I' \subseteq \{1, \ldots, k(n)\}$, $|I| = |I'| = t(n)$, $\sigma \in S_{t(n)}$.

The previous definition says that computationally, t-symmetry is broken if an efficient algorithm can find two projections that are different, as well as a certificate of the non-emptiness of their difference. We want the transformation of a LP task be such, that the possible set of results of the transformation is computationally t-symmetric for small values of t, where the asymmetry could be exploited for classifying the variables in the transformed LP task. In particular, we want the transformed LP task to be computationally 1-symmetric (hence the bounding box of the transformed task must be a hypercube) and 2-symmetric.

6. 2-SYMMETRIC TRANSFORMATIONS

Let us now consider transformations that are 2-symmetric and see what properties of the feasible regions of transformed tasks this implies. From now on, let the constraints of the transformed LP task be $A\vec{x} = \vec{b}$, $\vec{x} \geq \vec{0}$.

6.1 A computable property of polyhedra

Consider the polyhedron P determined by $A\vec{x} = \vec{b}$, $\vec{x} \geq \vec{0}$, with m equality constraints and n variables. Let $i, j \in \{1, \ldots, n\}$ be two coordinates, such that $x_i = x_j = 0$ does not contradict the constraints, and consider the projection of P to the (x_i, x_j)-plane. Assume the projection does not equal the whole first quadrant of the plane. Also assume that it is not a point or a line (segment). In this case, the projection is a convex, possibly unbounded polygon. Let O_1 be the vertex of the polygon at the coordinates $(0, 0)$. Let O_2 be the next vertex of the polygon, at the coordinates $(c, 0)$. It is possible that O_2 coincides with O_1; this happens if $x_j = 0$ implies $x_i = 0$. Let O_3 be the next vertex of the polygon after O_2. Let $\alpha_{ij}^P \in \mathbb{R}$ be such, that the side O_2O_3 lies on the line $x_i + \alpha_{ij}^P x_j = c$.

LEMMA 1. *There exists a polynomial-time algorithm that on input A, $\vec{b}$, i and j, satisfying the conditions above, computes α_{ij}^P (to an arbitrary level of precision).*

PROOF. The algorithm works as follows. It will first find the coordinate c by solving the LP task "maximize x_i, subject to $\vec{x} \in P$, $x_j = 0$".

Let $D(\vec{x}, \gamma)$ denote the direction $(\cos\gamma)\cdot x_i + (\sin\gamma)\cdot x_j$. Using binary search, the algorithm will then find $\gamma \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, such that the optimal solution for the LP task "maximize $D(\vec{x}, \gamma - \varepsilon)$ subject to $\vec{x} \in P$" is in a point $\vec{x'}$ with $x'_i = c$ and $x'_j = 0$, while the optimal solution for the LP task "maximize $D(\vec{x}, \gamma + \varepsilon)$, subject to $\vec{x} \in P$" is in a point $\vec{x''}$ with $(x''_i, x''_j) \neq (c, 0)$. Here $\varepsilon > 0$ is an arbitrarily small (depending on the required precision) angle. Fig. 2 depicts all these quantities.

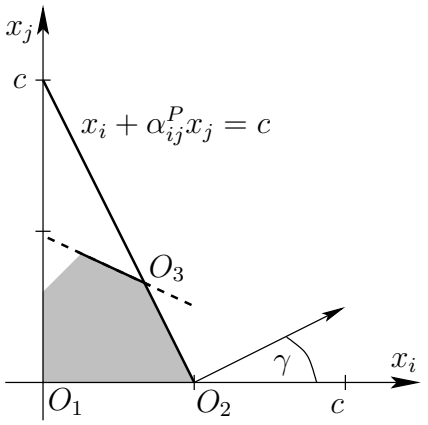

Figure 2: Finding the value of α_{ij}^P

On (x_i, x_j)-plane, the direction $D(\vec{x}, \gamma)$ is (almost) perpendicular to the line $x_i + \alpha_{ij}^P x_j = c$. Hence $\alpha_{ij}^P \approx \tan\gamma$. □

The existence of this algorithm shows that if a transformation producing the polyhedron $P \subseteq \mathbb{R}^n$ is computationally 2-symmetric, then the value α_{ij}^P must be the same for all coordinate pairs i, j. Let us denote it by α.

6.2 Scarcity

We show that any polyhedron whose α_{ij}^P is the same for all coordinate pairs i, j is a set of the form

$$\{(x_1, \ldots, x_n) \mid x_1 + \ldots + x_n \leq c\} \tag{4}$$

for some $c \geq 0$ ($c \in \mathbb{R}$) or $c = \infty$.

This result implies that any computationally 2-symmetric transformation $(\mathcal{F}, \mathcal{G})$ can be easily turned to a LP solving algorithm. Given a LP task, the solver will first apply $\mathcal{F}$ to it, resulting in the polyhedron (4) and an objective function. This polyhedron has at most $n+1$ vertices and the algorithm finds the optimal one by applying the objective function to each of them. The value c can also be straightforwardly found by adding the constraints $x_1 = \ldots = x_{n-1} = 0$ and then finding the maximum value of x_n (either by substituting 0 to $x_1, \ldots, x_{n-1}$, or by binary search).

Let the transformed polyhedron P be defined by $A\vec{x} = \vec{b}$, $\vec{x} \geq \vec{0}$, where A is a non-singular $m \times n$ matrix for $m \leq n-1$. If $m = n-1$, then P is one-dimensional, hence it has at most 2 vertices which can be easily found and $(\mathcal{F}, \mathcal{G})$ can again be turned into a LP solving algorithm. Thus we let $m \leq n-2$.

If $m \leq n-2$ and P contains at least one basic feasible solution, then at least two variables in this solution are 0. The reason is that basic solutions occur only on the intersections of constraining hyperplanes, and the system $A\vec{x} = \vec{b}$ provides at most m of them, so at least $n - m$ come from $x_j \geq 0$.

If there are two variables that can simultaneously have the value 0 in P, then *any* pair of two variables must have the

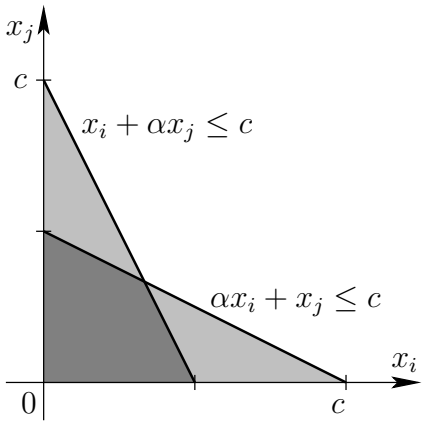

Figure 3: Symmetry property of a projection to (x_i, x_j)

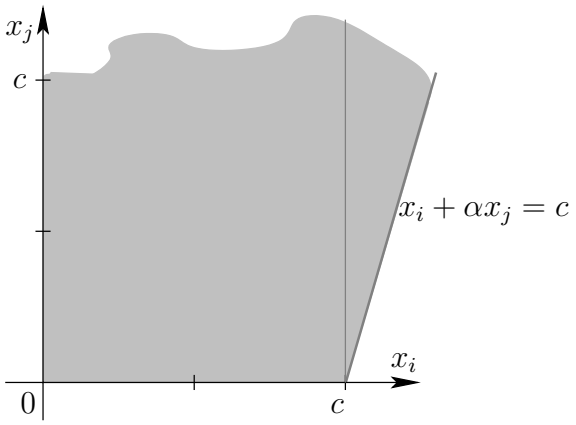

Figure 4: The case $\alpha \leq 0$

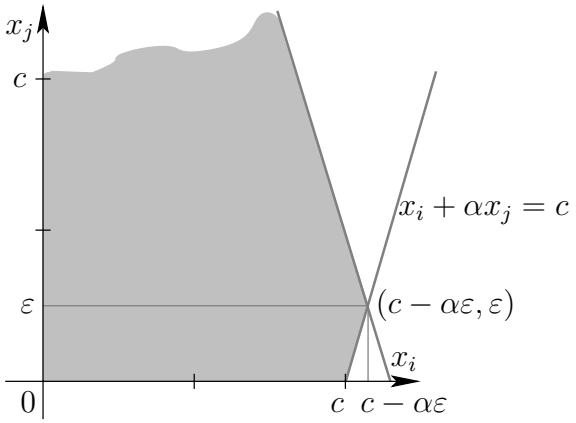

Figure 5: A point that exists on the line $x_i + \alpha x_j = c$ for $\alpha \leq 0$

same property, otherwise the (computational) 2-symmetry would be broken. For arbitrary i, j, consider the projection of P to the dimensions x_i and x_j. Since P is a convex polyhedron, this projection is a convex polygon. As discussed before, it contains the point $(0, 0)$.

Let O_2, O_3, c, $\alpha_{ij}^P = \alpha$ and $\gamma = \arctan \alpha$ be defined as in Sec. 6.1. One of the sides of the polygon is located on the line $x_i + \alpha x_j = c$. By symmetry, there also exists a side of the polygon that lies on the line $x_j + \alpha x_i = c$. Due to the convexity of the polygon, $\alpha \leq 1$, otherwise these lines pass through the interior of the convex hull of the points $(0, 0)$, $(c, 0)$ and $(0, c)$ that are contained in the polygon.

Since all projections onto any pair of variables (x_i, x_j) should have exactly the same angle γ and therefore the same $\tan \gamma = \alpha$, and exactly the same distance from the origin point c, a system of $2 \cdot \binom{n}{2}$ inequalities of the form $x_i + \alpha x_j \leq c$ is implied by the constraints defining the polyhedron P. Here $\binom{n}{2}$ is the number of possible pair choices. This number is multiplied by 2 since the inequality $x_i + \alpha x_j \leq c$ implies existence of the inequality $\alpha x_i + x_j \leq c$ due to the symmetry requirement, as shown in Fig. 3. Due to convexity of the projection, any valuation (v_i, v_j) of x_i and x_j such that $v_i + v_j \leq c$ must be possible.

Given n variables, any inequality $x_i + \alpha x_j \leq c$ comes from some equation of the form $a_1 x_1 + a_2 x_2 + \ldots + x_i + \ldots + \alpha x_j + \ldots + a_n x_n = c$ implied by the constraints defining P, where $a_k \geq 0$ for any k, and $a_\ell > 0$ for some ℓ (the variable x_ℓ acts as a slack variable for the inequality). In total, we get $2 \cdot \binom{n}{2}$ equations that all follow from these constraints:

$$\begin{cases} \alpha x_1 + x_2 + a_{123} x_3 + \ldots + a_{12(n-1)} x_{n-1} + a_{12n} x_n = c \\ x_1 + \alpha x_2 + a_{213} x_3 + \ldots + a_{21(n-1)} x_{n-1} + a_{21n} x_n = c \\ a_{231} x_1 + \alpha x_2 + x_3 + \ldots + a_{23(n-1)} x_{n-1} + a_{23n} x_n = c \\ a_{321} x_1 + x_2 + \alpha x_3 + \ldots + a_{32(n-1)} x_{n-1} + a_{32n} x_n = c \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n(n-1)1} x_1 + a_{n(n-1)2} x_2 + \ldots + \alpha x_{n-1} + x_n = c \\ a_{(n-1)n1} x_1 + a_{(n-1)n2} x_2 + \ldots + x_{n-1} + \alpha x_n = c \end{cases} \tag{5}$$

where $a_{ijk} \geq 0$ for all $i, j, k \in \{1, \ldots, n\}$, and each equation contains at least one strictly positive a_{ijk}. We will show that these equations imply $x_1 + \ldots + x_n = c$. Consider the possible values of α.

1. **The case $\alpha \leq 0$.** The side of the polygon represented by the line $x_i + \alpha x_j = c$ is either parallel to the x_j axis, or is tilted in the direction opposite from the x_j axis. This is illustrated by Fig. 4.

   Consider a point $v = (v_1, \ldots, v_n) \in P$ where $(v_i, v_j) = (c - \alpha\varepsilon, \varepsilon)$ for some $\varepsilon > 0$. Depending on ε, this can be any point that is located on the side on the line $x_i + \alpha x_j = c$. There definitely exist points on this side, otherwise that side would not be present on the projection at all. A possible location of such a point is shown in Fig. 5. Consider the equation $E \equiv a_{ij1} x_1 + \ldots + x_i + \ldots + \alpha x_j + \ldots + a_{ijn} x_n = c$ from the system (5). Since each equation represents an inequality, there exists some $a_k = a_{ijk} > 0$ that corresponds to some variable x_k, $k \neq i$, $k \neq j$ (x_k acts as a slack variable).

   (a) **The case $\alpha = 0$.** Due to the symmetry of the projections, there must exist a point $v' = (v'_1, \ldots, v'_n) \in P$ such that $(v'_i, v'_k) = (c - \alpha\varepsilon, \varepsilon) = (c, \varepsilon)$. But the point v' cannot possibly satisfy the equation E because already $v'_i + a_k v'_k = c + a_k \varepsilon > c$ and all other coefficients in E are non-negative.

   (b) **The case $\alpha < 0$.** We have $v_i + \alpha v_j = c - \alpha\varepsilon + \alpha\varepsilon = c$. The point v must satisfy the equation E. Hence $v_k = 0$, because $a_k > 0$. Now we have $(v_i, v_k) =$

$(c - \alpha\varepsilon, 0)$. The point v must also satisfy other equations of the system (5), including $a_{ik1}x_1 + \ldots + x_i + \ldots + \alpha x_k + \ldots + a_{ikn}x_n = c$. We have $v_i + \alpha v_k = c - \alpha\varepsilon + 0 > c$, hence this equation is violated.

Thus the only possible case is $\alpha > 0$.

2. **The case** $\alpha > 0$. If $c = 0$, then the only possible solution for the system (5) would be $x_1 = \ldots = x_n = 0$, since there are no negative entries at all. Consider the case $c > 0$.

   Let $v = (v_1, \ldots, v_n) \in P$ be a point where $v_i = c$. For any $j \neq i$, consider the equation $a_{ij1}x_1 + \ldots + x_i + \ldots + \alpha x_j + \ldots + a_{ijn}x_n = c$. As $v_i = c$, the left hand side of this equation, when applied to v, will be at least c. As it cannot be larger, we must have $\alpha v_j = 0$. This implies $v_j = 0$, because $\alpha > 0$. As j was arbitrary, we obtain $v = (v_1, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_n) = (0, \ldots, 0, c, 0, \ldots, 0)$.

   If the system (5) contains any equation where the coefficient of x_i is $a \neq 1$, then the point v cannot satisfy this equation, because $av_i \neq v_i = c$, and the other components of v cannot affect the left hand side of the equation since they are all 0. We get that the coefficient of x_i should be 1 in each equation.

   In the same way, we get that the coefficients of all the variables in all equations should be 1. This means that the only equation that remains is $x_1 + \ldots + x_n = c$.

Another thing that we would like to show is that if the equation system defined by P contains a constraint $x_1 + \ldots + x_n = c$ for some $c > 0$, then it is not allowed to have any other constraints. Suppose that the system contains the following two equations:

$$x_1 + \ldots + x_n = c \quad (6)$$

$$a_1x_1 + \ldots + a_nx_n = b \quad (7)$$

where $a_i, b \in \mathbb{R}$. We will show that the equation (7) can be at most a multiple of the equation (6), representing the same constraint.

Without loss of generality, let $a_1 = \min_i a_i$, $a_2 = \max_i a_i$. We may assume that $a_2 > a_1$ since if it was the case $a_2 = a_1$, then all the a_i would be equal, and the only possible way to avoid contradiction with (6) would be to assign $b = a_ic$, making the (7) a multiple of (6).

Multiplying (6) by a_1 and subtracting the result from (7), we get

$$(a_2 - a_1)x_2 + \ldots + (a_n - a_1)x_n = b - a_1c$$

Since $a_2 \neq a_1$, we may express the variable x_2 in terms of other variables:

$$x_2 = \frac{b - a_1c - \sum_{i=3}^{n}(a_i - a_1)x_i}{a_2 - a_1}$$

We know that the only allowed valuations of x_i are positive.

- Since $x_2 \geq 0$ and $a_2 > a_1$, the constraints defining the polyhedron P imply

$$b - a_1c \geq \sum_{i=3}^{n}(a_i - a_1)x_i$$

- From (6), $x_1 = c - \sum_{i=2}^{n} x_i$. We get

$$x_1 = c - \frac{b - a_1c - \sum_{i=3}^{n}(a_i - a_1)x_i}{a_2 - a_1} - \sum_{i=3}^{n} x_i = \frac{a_2c - b - \sum_{i=3}^{n}(a_2 - a_i)x_i}{a_2 - a_1}$$

- Since $x_1 \geq 0$ and $a_2 > a_1$, the constraints defining the polyhedron P imply

$$a_2c - b \geq \sum_{i=3}^{n}(a_2 - a_i)x_i$$

Recall that any variable must be allowed to take any value in the span $[0, c]$. Consider the valuation where $x_i = c$ for some i. From (6) and the requirement $\vec{x} \geq 0$, we get that $\forall j \neq i : \; x_j = 0$.

$$\begin{cases} b - a_1c \geq \sum_{i=3}^{n}(a_i - a_1)x_i \\ a_2c - b \geq \sum_{i=3}^{n}(a_2 - a_i)x_i \end{cases} \implies \begin{cases} b - a_1c \geq a_ic - a_1c \\ a_2c - b \geq a_2c - a_ic \end{cases}$$

$$\implies \begin{cases} b \geq a_ic \\ -b \geq -a_ic \end{cases} \implies a_i = \frac{b}{c} \; .$$

Similarly, we can show that for any $i \neq 1, 2$ we have $a_i = \frac{b}{c}$. What about a_1 and a_2?

From (6), we get that $x_3 + \ldots + x_n = c - x_1 - x_2$.

$$\begin{aligned} a_1x_1 + \ldots + a_nx_n &= b \\ a_1x_1 + a_2x_2 + \frac{b}{c}x_3 + \ldots + \frac{b}{c}x_n &= b \\ \left(a_1 - \frac{b}{c}\right)x_1 + \left(a_2 - \frac{b}{c}\right)x_2 &= 0 \end{aligned}$$

If either $\left(a_1 - \frac{b}{c}\right) \neq 0$ or $\left(a_2 - \frac{b}{c}\right) \neq 0$, then one variable may be expressed in terms of the other one, meaning that the projection to (x_1, x_2) can be only a line segment. This would mean that the entire polygon P is actually a line segment. We have explored this case before. But if this is not the case, then $a_1 = a_2 = \frac{b}{c}$.

We have obtained an equation $\frac{b}{c}x_1 + \ldots + \frac{b}{c}x_n = b$, and multiplying both sides by $\frac{c}{b}$ we get the same equation (6).

7. CONCLUSIONS

We have shown that the current approaches towards privacy-preserving outsourcing or multiparty linear programming are unlikely to be successful. Success in this direction requires some radically new ideas in transforming polyhedra and/or in cryptographic foundations violating the rather generous assumptions we have made in this paper. Alternatively, it may be fruitful to optimize privacy-preserving implementations of LP solving algorithms in order to have universal privacy-preserving optimization methods for large classes of tasks.

8. ACKNOWLEDGEMENTS

This work was supported by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS, and through the Software Technologies and Applications Competence Centre, STACC. It has also received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 284731.

9. REFERENCES

[1] Mikhail J. Atallah and Keith B. Frikken. Securely outsourcing linear algebra computations. In Dengguo Feng, David A. Basin, and Peng Liu, editors, *ASIACCS*, pages 48–59. ACM, 2010.

[2] Mikhail J. Atallah and Jiangtao Li. Secure outsourcing of sequence comparisons. *Int. J. Inf. Sec.*, 4(4):277–287, 2005.

[3] Alice Bednarz. *Methods for two-party privacy-preserving linear programming*. PhD thesis, University of Adelaide, 2012.

[4] Alice Bednarz, Nigel Bean, and Matthew 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, pages 117–120, New York, NY, USA, 2009. ACM.

[5] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM Conference on Computer and Communications Security*, pages 784–796. ACM, 2012.

[6] Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang, and Wenjing Lou. New algorithms for secure outsourcing of modular exponentiations. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS*, volume 7459 of *Lecture Notes in Computer Science*, pages 541–556. Springer, 2012.

[7] Yao Chen and Radu Sion. On securing untrusted clouds with cryptography. *IEEE Data Eng. Bull.*, 35(4):9–20, 2012.

[8] Jannik Dreier and Florian Kerschbaum. Practical privacy-preserving multiparty linear programming based on problem transformation. In *SocialCom/PASSAT*, pages 916–924. IEEE, 2011.

[9] Wenliang Du. *A Study Of Several Specific Secure Two-Party Computation Problems*. PhD thesis, Purdue University, 2001.

[10] Wenliang Du and Zhijun Zhan. A practical approach to solve secure multi-party computation problems. In *New Security Paradigms Workshop*, pages 127–135. ACM Press, 2002.

[11] Sergei Evdokimov and Oliver Günther. Encryption techniques for secure database outsourcing. In Joachim Biskup and Javier Lopez, editors, *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2007.

[12] Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 264–282. Springer, 2005.

[13] Yuan Hong and Jaideep Vaidya. An inference-proof approach to privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 2013. To appear. Published online 05 October 2012.

[14] Yuan Hong, Jaideep Vaidya, and Haibing Lu. Secure and efficient distributed linear programming. *Journal of Computer Security*, 20(5):583–634, 2012.

[15] Peeter Laud and Alisa Pankova. New Attacks against Transformation-Based Privacy-Preserving Linear Programming. In Rafael Accorsi and Silvio Ranise, editors, *Security and Trust Management (STM) 2013, 9th International Workshop*, volume 8203 of *Lecture Notes in Computer Science*. Springer, 2013.

[16] Jiangtao Li and Mikhail J. Atallah. Secure and private collaborative linear programming. In *International Conference on Collaborative Computing*, pages 1–8, 2006.

[17] Wei Li, Haohao Li, and Chongyang Deng. Privacy-preserving horizontally partitioned linear programs with inequality constraints. *Optimization Letters*, 7(1):137–144, 2013.

[18] Olvi L. Mangasarian. Privacy-preserving linear programming. *Optimization Letters*, 5(1):165–172, 2011.

[19] Olvi L. Mangasarian. Privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 6(3):431–436, 2012.

[20] Tomas Toft. Solving linear programs using multiparty computation. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security*, pages 90–107, Berlin, Heidelberg, 2009. Springer-Verlag.

[21] Jaideep Vaidya. Privacy-preserving linear programming. In Sung Y. Shin and Sascha Ossowski, editors, *SAC*, pages 2002–2007. ACM, 2009.

[22] Cong Wang, Kui Ren, and Jia Wang. Secure and practical outsourcing of linear programming in cloud computing. In *INFOCOM, 2011 Proceedings IEEE*, pages 820–828, 2011.

[23] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE, 1982.