Privacy-Aware Quadratic Optimization Using Partially Homomorphic Encryption

Yasser Shoukry[(1,2)], Konstantinos Gatsis[(3)], Amr Alanwar[(2)]
George J. Pappas[(3)], Sanjit A. Seshia[(1)], Mani Srivastava[(2)], and Paulo Tabuada[(2)]

***Abstract*— We consider a problem where multiple agents participate in solving a quadratic optimization problem subject to linear inequality constraints in a privacy-preserving manner. Several variables of the objective function as well as the constraints are privacy-sensitive and are known to different agents. We propose a privacy-preserving protocol based on partially homomorphic encryption where each agent encrypts its own information before sending it to an untrusted cloud computing infrastructure. To find the optimal solution the cloud applies a gradient descent algorithm on the encrypted data without the ability to decrypt it. The privacy of the proposed protocol against coalitions of colluding agents is analyzed using the cryptography notion of zero knowledge proofs.**

I. INTRODUCTION

In this paper we consider the problem of solving an important class of convex optimization problems frequently appearing in estimation and control applications, namely quadratic programs with linear inequality constraints, defined over privacy-sensitive data. There has been a vast amount of work in the literature targeting the problem of performing computations over sensitive data. We classify the existing work based on the techniques used to ensure privacy into three classes: 1) differential privacy, 2) obfuscation, and 3) homomorphic encryption.

The notion of differential privacy was initially introduced in the database literature and ensures that if the user decides to participate in a certain database, then the risk of breaching privacy of the user's data by means of data queries is low. To ensure privacy of the user data against data queries, a trusted data aggregator corrupts the output of a query by adding appropriately structured noise [1]. If alternatively the aggregator is not trusted, the notion of local differential privacy specifies that every user corrupts its own data before sending it to the aggregator. Several recent works adopted differential privacy to solve convex optimization problems while preserving privacy of user data [2], [3], [4], [5]. These works cover different setups based on the presence or absence of a trusted data aggregator. Besides these advances, a major drawback of differential privacy-based protocols for optimization is the trade-off between accuracy and privacy. That is, to provide higher privacy guarantees a protocol needs to add more noise, resulting in the final output differing significantly from the desired optimal solution [3], [5], and thus affecting the usability of the output.

The second class of work relies on obfuscation-based techniques. A randomized transformation, known as the obfuscation transformation, is applied to the actual optimization problem before it is sent to the cloud to be solved. Upon finding the optimal solution of the obfuscated problem, the optimal solution of the original problem can be retrieved by inverting the obfuscation transformation. Representative of this class is [6], [7] which considers linear and quadratic programming problems. Unfortunately, existing obfuscation-based techniques are only applicable in the case where all the data is owned by a single agent.

The third class of work, which is the one followed in our paper, is based on homomorphic encryption techniques. Homomorphic cryptosystems have the distinguishing ability to perform operations on encrypted data without being able to decrypt it [8]. A potential approach to implement convex optimization algorithms over encrypted data is to use the new generation of fully homomorphic encryption (FHE) algorithms [9], [10]. Unfortunately, FHE is overwhelmingly impractical in terms of execution time [8]. On the other hand, partially homomorphic encryption (PHE) has been shown to be computationally practical but comes with the cost of supporting only limited mathematical operations over encrypted data. Yet, in recent work PHE has been used to perform machine learning classification [11] and statistical estimation of time series [12]. This opens the door for implementing practical convex optimization algorithms over encrypted data.

Our work describes how partially homomorphic encryption can be used to solve quadratic programs with linear inequality constraints, with formal privacy guarantees. A major advantage of our technique based on PHE in comparison to the differential privacy approaches is that it provides strong formal privacy guarantees *without compromising the optimality of the solution.* In comparison to the aforementioned PHE-based works which are tailored towards specific applications, we show how to solve general classes of quadratic optimization problems. For brevity, the proofs of the main technical results are omitted here and are reported in an extended version of the paper [13].

This work was partially sponsored by the NSF award CNS-1505799 and the Intel-NSF Partnership for Cyber-Physical Systems Security and Privacy, by the NSF award 1136174, by DARPA under agreement number FA8750-12-2-0247, by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA or the U.S. Government.

The authors are affiliated with [1] the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, [2] the Department of Electrical Engineering, University of California, Los Angeles, CA, [3] the Department of Electrical and Systems Engineering, University of Pennsylvania, PA.

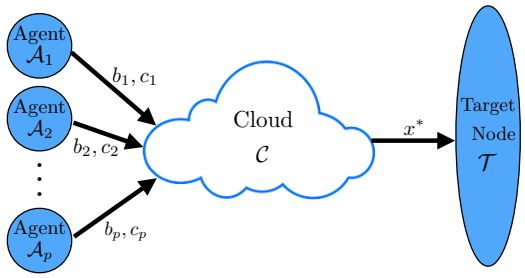

Fig. 1: Diagram showing the problem setup. Agents $\mathcal{A}_1, \ldots, \mathcal{A}_p$ sends their private information to the cloud $\mathcal{C}$ which solves the optimization problem and sends the optimal solution to the target node $\mathcal{T}$.

II. Problem Setup

A. Notation

The symbols $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{R}_{\geq 0}$ denote the set of natural, integer, real, and non-negative numbers, respectively. The integer ring of size N is denoted by $\mathbb{Z}_N$. The set of positive definite matrices of size $n \times n$ is denoted by $\mathbb{S}^n_{++}$. We denote by $[\![a]\!]_{\text{pk}}$ the encryption of $a \in \mathbb{Z}$ using some encryption algorithm and the public encryption key pk. Similarly, we denote by $\text{DECRYPT}_{\text{sk}}([\![a]\!]_{\text{pk}})$ the decryption of $[\![a]\!]_{\text{pk}}$ using the same algorithm and the secret decryption key sk.

B. Problem Setup

We start by defining the parties involved in our problem. As shown in Fig. 1, we have three types of parties:

- **Agents** $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_p)$: is a set of *untrusted parties* participating in solving the optimization problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} x^T Q x + c_{\mathcal{A}}^T x \tag{1}$$

$$\text{subject to} \quad A\, x \leq b_{\mathcal{A}} \tag{2}$$

  where $b_{\mathcal{A}} \in \mathbb{R}^m, c_{\mathcal{A}} \in \mathbb{R}^n$ is privacy-sensitive information owned by the agents (we use the subscript $\mathcal{A}$ to stress the fact that this is the agents' sensitive information). More specifically we can decompose $b_{\mathcal{A}} = (b_1, \ldots, b_p), c_{\mathcal{A}} = (c_1, \ldots, c_p)$ so that each agent i owns the parts $b_i \in \mathbb{R}^{m_i}, c_i \in \mathbb{R}^{n_i}$ with $\sum_{i=1}^p m_i = m, \sum_{i=1}^p n_i = n$. Here n, m are the dimensions of the optimization vector x and the number of inequality constraints, respectively. The matrices $Q \in \mathbb{S}^n_{++}$ and $A \in \mathbb{R}^{m \times n}$ are assumed to be public information and therefore known to all parties. The optimal solution x^* of this optimization problem is also privacy-sensitive and needs to be kept private.
- **Cloud** $\mathcal{C}$**:** An *untrusted party* which has a known public key $\text{pk}_{\mathcal{C}}$ and a hidden private key $\text{sk}_{\mathcal{C}}$.
- **Target Node** $\mathcal{T}$**:** An *untrusted party* which has a known public key $\text{pk}_{\mathcal{T}}$ and a hidden private key $\text{sk}_{\mathcal{T}}$. The target node is the only node that is entitled to know the final output of the optimization problem x^*. This target node can be physically the same as any of the agents, or it can be an entirely different party (other than the cloud).

As shown in Figure 1 we are interested in a protocol where the cloud collects the agents information and provides the final solution x^* of the optimization problem to the target node with strong privacy guarantees, formally described next.

The quadratic optimization problem (1)-(2) arises in many applications. For example, consider the problem of estimating the state x of a dynamical system from sensitive sensor measurements $y_{\mathcal{A}}$. Under the assumption that the system dynamics are public, the state estimation problem can be cast as a least squares problem of the form: $\min_{x \in \mathbb{R}^n} \frac{1}{2} \|\mathcal{O}x - y_{\mathcal{A}}\|_2^2$ where $\mathcal{O}$ is the well known observability matrix computed from the publicly known system dynamics. In such a scenario, the matrix Q in (1) and the vector $c_{\mathcal{A}}$ in (2) become $Q = \mathcal{O}^T\mathcal{O}$ and $c_{\mathcal{A}} = \mathcal{O}^T y_{\mathcal{A}}$ respectively. The inequality constraints in (2) capture the scenario where, for example, the state is constrained to be in a polyhedron whose shape is publicly known (captured by the matrix A) but its position and size (captured by the vector $b_{\mathcal{A}}$) depends again on sensitive information.

C. Adversarial Model and Privacy Goals

We consider a *semi-honest* adversarial model, also known as *honest but curious*. A *semi-honest* adversary is defined as one who follows the exact computations of a protocol with the exception that it keeps record of all intermediate computations and communication messages exchanged. It may then use all recorded information along with any other publicly available information to leak information about the sensitive data. In addition, we assume that all messages are exchanged using a broadcasting channel. Therefore, all the entities can listen to all the exchanged messages

In our setup of Figure 1 different sets of parties might be adversarial, hence we *informally* state our privacy notions. No party involved in the optimization protocol should be able to determine the values of the sensitive information owned by other parties. This privacy guarantee should hold even if some of the parties collude and exchange their private information, cryptographic keys, and/or intermediate computation results. More specifically we consider the following privacy goals:

- **Privacy against agent coalitions:** By the end of the protocol execution, if any agent colludes with up to k other agents by exchanging their private values, the coalition should *gain zero knowledge* about the non-colluding agents' private information.
- **Privacy against cloud coalitions:** By the end of the protocol execution, if the cloud $\mathcal{C}$ colludes with up to k agents by exchanging their k private values, cryptographic private keys, and intermediate results computed at $\mathcal{C}$, the coalition should *gain zero knowledge* about the non-colluding agents' private information.
- **Privacy against target node coalitions:** By the end of the protocol execution, if the target node $\mathcal{T}$ colludes with up to k agents by exchanging their k private values, cryptographic private keys, and the final decrypted outcome of the optimization algorithm, the coalition should *gain zero knowledge* about the non-colluding agents' information.

In Section IV-A these privacy goals are *formally* defined using the notion of *zero-knowledge proofs*.

D. Paillier Additive Partial Homomorphic Encryption

Our protocols utilize a particular homomorphic cryptosystem named "Paillier cryptosystem" [14]. The Paillier cryptosystem is a public-key cryptosystem that enjoys the following features:

- **Probabilistic encryption:** For a given integer $a \in \mathbb{Z}_N$ (where $N = pq$ is a product of two prime numbers p and q), there exist N encryptions of a, i.e., $[\![a]\!]_{\text{pk}} \in \mathcal{E}nc(a) \subset \mathbb{Z}_{N^2}$ where $\mathcal{E}nc(a)$ is the set of all encryptions of the plaintext a with $|\mathcal{E}nc(a)| = N$. At encryption time, the Paillier cryptosystem randomly picks one possible encryption from the set $\mathcal{E}nc(a)$.
- **Additive Homomorphism:** Paillier cryptosystem allows to add two encrypted values, i.e., there exists an addition operator $\oplus$ such that $[\![a]\!]_{\text{pk}} \oplus [\![b]\!]_{\text{pk}} \in \mathcal{E}nc(a+b)$ for any $a, b \in \mathbb{Z}_N$.
- **Multiplication by plain text:** Paillier cryptosystem allows the multiplication of an encrypted value with a plaintext one, i.e., there exists a multiplication operator $\otimes$ such that $a \otimes [\![b]\!]_{\text{pk}} \in \mathcal{E}nc(ab)$ for any $a, b \in \mathbb{Z}_N$.

Note that the last two properties can be generalized to perform multiplication between a plaintext matrix and a vector of encrypted variables. That is, given a vector $x \in \mathbb{Z}_N^n$ and a matrix $M \in \mathbb{Z}_N^{n \times n}$, we can compute $M[\![x]\!]_{\text{pk}}$ where, for simplicity of notation, we used the juxtaposition in $M[\![x]\!]_{\text{pk}}$ to denote the multiplication of a plaintext matrix M with an encrypted vector $[\![x]\!]_{\text{pk}}$.

Although the Paillier cryptosystem operates over non-signed integers, the convex optimization problem (1)-(2) is defined over real variables. For that reason, we rely on a variant of fixed-point arithmetic encoding [15] which allows for both signed and fixed-precision non-integer values. Fixed-precision arithmetic leads to a solution with finite precision and hence introduces numerical errors to the final solution of the optimization problem. Such errors can be decreased by increasing the scaling factor (number of digits after the radix point) used to represent fractions. In this paper we do not analyze the errors due to fixed-precision arithmetic.

III. Privacy Preserving Quadratic Programming

Consider the quadratic optimization problem subject to linear inequality constraints presented in (1)-(2). The difficulty in using PHE to solve this optimization problem in a privacy-preserving manner is that PHE only supports a limited number of operations on encrypted data (cf. Section II-D). Hence common interior-point methods for constrained optimization [16, Ch. 11] require operations which might be challenging to implement over encrypted data. We identify instead an alternative method which is amenable to PHE-based implementation, in particular a projected dual gradient method.

Towards this end we associate dual variables $\mu \in \mathbb{R}^m_{\geq 0}$ for the constraints in (2) and define the Lagrange dual problem [16, Ch. 5] as follows

$$\underset{\mu \in \mathbb{R}^m_{\geq 0}}{\text{maximize}} \quad g(\mu) = -\frac{1}{2}(A^T\mu + c_{\mathcal{A}})^T Q^{-1}(A^T\mu + c_{\mathcal{A}}) - \mu^T b_{\mathcal{A}} \tag{3}$$

where $g(\mu)$ is called the dual function. We assume that the polyhedral set (2) is strictly feasible (Slater's condition [16, Ch. 5.2]), i.e., there exists some $x \in \mathbb{R}^n$ such that $A\,x < b_{\mathcal{A}}$. Then the original problem (1)-(2) and its dual above have the same optimal objective value. Moreover, the desired optimal solution x^* of the original problem is expressed with respect to the optimal solution μ^* of the dual problem as:

$$x^* = -Q^{-1}(A^T\mu^* + c_{\mathcal{A}}), \tag{4}$$

where the inverse exists as Q is assumed positive definite. This relationship follows from the KKT optimality conditions [16, Ch. 5.5.3] as the gradient of the Lagrangian function vanishes at the primal optimal solution.

We propose hence to solve (3) and find μ^* using the following projected gradient ascent iteration:

$$\mu_{k+1} = \max\big\{0,\, \mu_k + \eta\, \nabla g(\mu_k)\big\}, \tag{5}$$

where:

$$\nabla g(\mu) = -AQ^{-1}(A^T\mu + c_{\mathcal{A}}) - b_{\mathcal{A}}. \tag{6}$$

is the gradient of the dual objective function and $\eta > 0$ is a constant step size. The step size η can be chosen so as to guarantee convergence to the optimal dual point μ^*.

Next we develop a privacy-preserving implementation of the gradient algorithm (5) using the additive homomorphic encryption cryptosystem described in Section II-D. This is facilitated by the fact that the sensitive information $c_{\mathcal{A}}$ and $b_{\mathcal{A}}$ appears additively in the gradient direction (6) and multiplied only by publicly known matrices.

A. Protocol Description

The proposed protocol consists of the following steps:
Step1-Encrypt: Each agent $\mathcal{A}_i$, $i \in \{1, \ldots, p\}$ encrypts its sensitive information b_i and c_i using the public key $\text{pk}_{\mathcal{T}}$ of the target node $\mathcal{T}$, followed by a second encryption using the public key $\text{pk}_{\mathcal{C}}$ of the cloud $\mathcal{C}$, and sends the result as a message to the cloud $\mathcal{C}$:

$$msg_i := ([\![[\![b_i]\!]_{\text{pk}_{\mathcal{T}}}]\!]_{\text{pk}_{\mathcal{C}}}, [\![[\![c_i]\!]_{\text{pk}_{\mathcal{T}}}]\!]_{\text{pk}_{\mathcal{C}}}) \tag{7}$$

The first encryption guarantees that the cloud $\mathcal{C}$ cannot directly access the agents' data, while the second encryption prevents the target node $\mathcal{T}$ from listening to and decrypting the messages $[\![b_i]\!]_{\text{pk}_{\mathcal{T}}}, [\![c_i]\!]_{\text{pk}_{\mathcal{T}}}$ using its own private key $\text{sk}_{\mathcal{T}}$.
Step2-Compute: Upon receiving all messages, the cloud $\mathcal{C}$ decrypts them using its own private key and constructs:

$$[\![b_{\mathcal{A}}]\!]_{\text{pk}_{\mathcal{T}}} = \begin{bmatrix} [\![b_1]\!]_{\text{pk}_{\mathcal{T}}} \\ \vdots \\ [\![b_p]\!]_{\text{pk}_{\mathcal{T}}} \end{bmatrix}, \qquad [\![c_{\mathcal{A}}]\!]_{\text{pk}_{\mathcal{T}}} = \begin{bmatrix} [\![c_1]\!]_{\text{pk}_{\mathcal{T}}} \\ \vdots \\ [\![c_p]\!]_{\text{pk}_{\mathcal{T}}} \end{bmatrix}. \tag{8}$$

Next, the cloud initializes the dual variable $\mu_0 = 0$ and encrypts it as $[\![\mu_0]\!]_{\text{pk}_{\mathcal{T}}}$. Then following (5)-(6) the cloud computes at every iteration:

$$\begin{aligned} [\![\nabla g(\mu_k)]\!]_{\text{pk}_{\mathcal{T}}} &= -AQ^{-1}(A^T[\![\mu_k]\!]_{\text{pk}_{\mathcal{T}}} \oplus [\![c_{\mathcal{A}}]\!]_{\text{pk}_{\mathcal{T}}}) \ominus [\![b_{\mathcal{A}}]\!]_{\text{pk}_{\mathcal{T}}}, \\ [\![\overline{\mu}_{k+1}]\!]_{\text{pk}_{\mathcal{T}}} &= [\![\mu_k]\!]_{\text{pk}_{\mathcal{T}}} \oplus \eta\, [\![\nabla g(\mu_k)]\!]_{\text{pk}_{\mathcal{T}}}. \end{aligned} \tag{9}$$

Both computations over encrypted data are enabled by additive homomorphic encryption.

Agent Nodes $\mathcal{A}_i$	Cloud $\mathcal{C}$	Target Node $\mathcal{T}$
Input: b_i, c_i	**Input**: -	**Input**: -
1. Encrypt the private information: $msg_i \leftarrow$ (8); 2. Send encrypted messages to $\mathcal{C}$; →	0. Uniformly generate a positive random number r; 3. Decrypt all messages sent from $\mathcal{A}_i$; 4. Construct the vectors: $[\![b_{\mathcal{A}}]\!]_{\text{pk}_\mathcal{T}}, [\![c_{\mathcal{A}}]\!]_{\text{pk}_\mathcal{T}} \leftarrow$ (7); 5. Encrypt the initial iterate $[\![\mu_0]\!]_{\text{pk}_\mathcal{T}} = [\![0]\!]_{\text{pk}_\mathcal{T}}$; 6. Iteratively update the dual variables: $[\![\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}} \leftarrow$ (9); 7. Calculate the obfuscated vector: $[\![r\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}} \leftarrow$ (10); 8. Send $[\![r\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}}$ to $\mathcal{T}$; →	
		9. Decrypt $[\![r\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}}$; 10. Send back the projected dual variables $[\![r\mu_{k+1}(i)]\!]_{\text{pk}_\mathcal{T}} \leftarrow$ (11) ←

	11. Calculate the dual variables: $[\![\mu_{k+1}]\!]_{\text{pk}_\mathcal{T}} \leftarrow$ (12);
	12. Check termination condition:
	if $k < \overline{k}$
	Repeat steps 6-12;
	else
	Compute the candidate optimal solution: $[\![x_K]\!]_{\text{pk}_\mathcal{T}} \leftarrow$ (13);
	Send $[\![x_K]\!]_{\text{pk}_\mathcal{T}}$ to $\mathcal{T}$; →
	end if	

		14. Decrypt $[\![x_K]\!]_{\text{pk}_\mathcal{T}}$ and obtain x_K;
Output: -	**Output:** -	**Output:** $x_K, r\overline{\mu}_{K-1}, r\overline{\mu}_K$

TABLE I: Protocol for privacy-aware quadratic programs with inequality constraints (thick arrows indicate the messages exchanged between different parties).

The remaining step in order to compute the dual variable $[\![\mu_{k+1}]\!]_{\text{pk}_\mathcal{T}}$ for the next iteration would be to take the maximum between $[\![0]\!]_{\text{pk}_\mathcal{T}}$ and $[\![\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}}$ as in (5). Unfortunately, partial homomorphic encryption does not preserve ordering and hence comparisons over encrypted data are not supported. That is, for two integers $a \leq b$ in general $[\![a]\!]_{\text{pk}_\mathcal{T}} \not\leq [\![b]\!]_{\text{pk}_\mathcal{T}}$. To overcome this, we propose to offload the comparison to the target node $\mathcal{T}$. To avoid the latter from having direct access to the value of $\overline{\mu}_{k+1}$, the cloud obfuscates the elements of $[\![\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}}$ by multiplying them with a uniformly drawn positive random number $r \in \mathbb{R}_{>0}$. That is, the cloud computes

$$[\![r\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}} = rI \otimes [\![\overline{\mu}_{k+1}]\!]_{\text{pk}_\mathcal{T}} \tag{10}$$

where I is the identity matrix of size $m \times m$, and sends it to the target node $\mathcal{T}$.

Upon receiving the message, the target node $\mathcal{T}$ decrypts it using its private key and projects it to the non-negatives to obtain the obfuscated projected dual variables as:

$$r\mu_{k+1} = \max\{0,\, r\overline{\mu}_{k+1}\}. \tag{11}$$

These values are then encrypted again using the public key of the target node and sent back to the cloud $\mathcal{C}$. Thanks to the properties of the partial homomorphic encryption, the cloud $\mathcal{C}$ can retrieve the value of $[\![\mu_{k+1}]\!]_{\text{pk}_\mathcal{T}}$ by performing:

$$[\![\mu_{k+1}]\!]_{\text{pk}_\mathcal{T}} = \frac{1}{r} I \otimes [\![r\mu_{k+1}]\!]_{\text{pk}_\mathcal{T}}. \tag{12}$$

We note that the employed obfuscation technique is known as *multiplicative blinding* and is similar to the well known *additive blinding* "one-time pad". In the next section, we show that multiplicative blinding preserves the privacy of $[\![\mu_{k+1}]\!]_{\text{pk}_\mathcal{T}}$.

Step3-Termination: The iterations in the previous step terminate after a fixed number of iterations K. Then the cloud $\mathcal{C}$ computes the encrypted candidate optimal solution:

$$[\![x_K]\!]_{\text{pk}_\mathcal{T}} = -Q^{-1}(A^T [\![\mu_K]\!]_{\text{pk}_\mathcal{T}} + c_{\mathcal{A}}), \tag{13}$$

similar to (4) and sends it to the target node $\mathcal{T}$. At the end of the protocol, the target node $\mathcal{T}$ declares its possession of the final output x_K. For technical reasons we artificially include the decrypted values of the last two messages the target received, i.e., $r\overline{\mu}_{K-1}, r\overline{\mu}_K$, as protocol outputs - see Remark IV.7.

IV. Privacy Analysis of Quadratic Programming Protocol

Our goal in this section is to formally analyze the privacy guarantees of the proposed optimization protocol. In particular we show that semi-honest colluding parties do not gain any information by participating in the protocol, or in other words, they cannot use any polynomial time algorithm to tell apart their actual observations from the observations they would obtain from a different set of private data. To make this precise we employ the notion of zero-knowledge proofs which in turn depends on the notion of computational indistinguishability.

A. Computational Indistinguishability and Zero Knowledge Proofs

In this subsection, we review the standard notion of computational indistinguishability which will be the basis for analyzing the privacy guarantees of Protocol I.

Definition IV.1 (Computational Indistinguishability (Definition 3.2.2 [17]))**.** *Let X_i and Y_i be binary random variables. The ensembles $\mathcal{X}_w = (X_1, \ldots, X_w)$ and $\mathcal{Y}_w = (Y_1, \ldots, Y_w)$*

are computationally indistinguishable in polynomial time, and we write $\mathcal{X}_w \equiv_c \mathcal{Y}_w$, if for every probabilistic polynomial time algorithm D, every positive polynomial $p : \mathbb{N} \to \mathbb{R}_{>0}$, and all sufficiently large lengths $w \in \mathbb{N}$, the following is satisfied:

$$\Big|Pr[D(x) = 1] - Pr[D(y) = 1]\Big| < \frac{1}{p(w)}$$

where x and y are random samples drawn from the ensembles $\mathcal{X}_w$ and $\mathcal{Y}_w$, respectively.

It follows from the previous definition that even if an adversary has access to multiple samples, as long as their number is polynomial, the probability of distinguishing between the two ensembles is still negligible (Theorem 3.2.6 [17]).

Consider now a protocol where messages are exchanged between different parties to compute some function f (where in our setup the function f is the optimal solution of the quadratic optimization problem). First, we formally define the information obtained by each participating party during the execution of the protocol using the notion of the *execution view* of a party.

Definition IV.2 (Execution View)**.** *Let f be a deterministic polynomial-time function and Π a multi-party protocol computing f. Let $\mathcal{A}_1, \ldots, \mathcal{A}_p$ be a set of agents that compute $f(a_1, \ldots, a_p)$ using Π, where a_i is $\mathcal{A}_i$'s input. The view of a coalition of agents $\mathcal{A}_j$, with $j \in J \subseteq \{1, \ldots, p\}$, during the execution of Π is the tuple:*

$$V_{\mathcal{A}_{j\in J}}(a_1, \ldots, a_p) = (a_{j\in J}; r^{\mathcal{A}_{j\in J}}; m_1^{\mathcal{A}_{j\in J}}, \ldots, m_t^{\mathcal{A}_{j\in J}}),$$

where $r^{\mathcal{A}_j}$ is $\mathcal{A}_j$'s random variables and $m_1^{\mathcal{A}_j}, \ldots, m_t^{\mathcal{A}_j}$ are the messages received by $\mathcal{A}_j$.

The protocol privately computes f if all the information gained by a semi-honest party during the execution of the protocol could have also been derived just from the input and output available to that party, i.e., without even participating in the protocol. This happens if an algorithm S, referred to as *simulator*, using only the party's input and the final protocol output can generate an execution view that is computationally indistinguishable from the party's actual execution view obtained during the execution of the protocol. If such a simulator S exists, then the adversarial party *gains zero knowledge* by participating in the protocol. This notion of *zero-knowledge proof* is captured by the following definition.

Definition IV.3 (Zero-Knowledge Proof (Definition 7.2.1 [18]))**.** *Consider the multi-party protocol Π that computes the deterministic polynomial-time function $f = (f_{\mathcal{A}_1}, \ldots, f_{\mathcal{A}_p})$ where $f_{\mathcal{A}_i}$ is the output of f to agent $\mathcal{A}_i$. A coalition $\mathcal{A}_{j\in J}$ of semi-honest adversaries is said to gain zero knowledge from the execution of protocol Π if there exists a probabilistic polynomial time algorithm $S_{\mathcal{A}_{j\in J}}$ such that for every possible input $a_1, \ldots, a_p$ of f:*

$$S_{j\in J}(a_{j\in J}, f_{\mathcal{A}_{j\in J}}(a_1, \ldots, a_p)) \equiv_c V_{\mathcal{A}_{j\in J}}(a_1, \ldots, a_p),$$

where $f_{\mathcal{A}_{j\in J}}$ is all the outputs owned by the coalition $\mathcal{A}_{j\in J}$ and $V_{\mathcal{A}_{j\in J}}(a_1, \ldots, a_p)$ is the execution view of the coalition.

It is crucial to note that the notion of *zero-knowledge proof* ensures that the distributed computation of f across different agents leaks no more information than a centralized computation, but it does not examine whether the final output of the function leaks information about the function input. The latter requires a separate analysis, which we include for our specific optimization problem in in Section IV-C.

B. Privacy Leakage due to Communication and Internal Computations

We start by analyzing the leakage due to the exchange of the messages in the proposed protocol. Recall that in Section II-C, we assume that all messages are exchanged using a broadcast channel and hence, all the entities can listen to all the exchanged messages. First we analyze the case where agents collude by exchanging their private inputs as follows.

Theorem IV.4. *Consider the privacy-preserving quadratic programming protocol (Protocol I). Assuming a semi-honest adversarial model and broadcast communication channels, under standard cryptographic assumptions, namely the Decisional Composite Residuosity (DCR) assumption[1], any coalition of up to $p - 1$ agents gains zero knowledge from the execution of the protocol.*

Next we analyze the case where the cloud $\mathcal{C}$ colludes with agents by exchanging their private inputs and private decryption keys. This is captured by the following result.

Theorem IV.5. *Consider the privacy-preserving quadratic programming protocol (Protocol I). Assuming a semi-honest adversarial model and broadcast communication channels, under standard cryptographic assumptions, namely the Decisional Composite Residuosity (DCR) assumption, any coalition consisting of the cloud $\mathcal{C}$ and up to $p - 1$ agents gains zero knowledge from the execution of the protocol.*

Finally, we analyze the case where the target node $\mathcal{T}$ colludes with agents by exchanging their private inputs, the final output of the protocol x^* as well as private decryption keys. This is captured by the following result.

Theorem IV.6. *Consider the privacy-preserving quadratic programming protocol (Protocol I). Assuming a semi-honest adversarial model and broadcast communication channels, under standard cryptographic assumptions, namely the Decisional Composite Residuosity (DCR) assumption, any coalition consisting of the target node $\mathcal{T}$ and up to $p - 1$ agents gains zero knowledge from the execution of the protocol.*

Remark IV.7. *As can be seen from Definition IV.3 zero knowledge proofs argue about leakage from messages a party receives given its outputs. Our Protocol I includes messages the target node receives as outputs because in*

[1]Let $N = p \times q$, $|N| = \lambda$ be the product of two distinct odd primes p and q. A number z is said to be an Nth residue modulo N^2 if there exists a number $y \in \mathbb{Z}_{N^2}$ such that $z = y^N \bmod N^2$. The DCR assumption states that the Nth residues are computationally indistinguishable from non Nth residues with respect to probabilistic polynomial time algorithms. The DCR assumption is one of the standard assumptions in the cryptography literature and is used to ensure the semantic security of the Paillier cryptosystem [14].

general they leak private information that is not covered under the notion of zero knowledge proofs. This privacy concern is examined in the following section. The reason only the last two messages suffice to be included in the output is that due to the recursive form of the algorithm all previous messages can be uniquely reconstructed from these two.

C. Privacy Leakage due to Protocol Output

The next step is to analyze the privacy leaked from the final output of the protocol. Since at the end of the protocol neither the agents nor the cloud obtain any output, we focus only on analyzing the case for the target node. In particular after running the protocol for infinitely long time so that it has converged, the target node $\mathcal{T}$ obtains outputs: i) the final solution which has converged to the optimal x^*, and ii) the last two obfuscated messages which have converged to the same limit value denoted here by $r\bar{\mu}^*$. To infer private input information, the target needs to argue about what values of the private input variables lead to this observed output.

Proposition IV.8. *The private value $b_{\mathcal{A}}$ cannot be uniquely determined from the outputs x^* and $r\bar{\mu}^*$ of the target node $\mathcal{T}$ if and only if $r\bar{\mu}_i^* < 0$ for some $i \in \{1, \dots, m\}$.*

Here $r\bar{\mu}_i^* < 0$ implies that the corresponding optimal dual value is zero, $\mu_i^* = 0$. This means that at least one constraint is inactive at the optimal solution x^* [16, Ch. 5.5.2] and the target node cannot uniquely determine the corresponding bound $b_{\mathcal{A},i}$. On the other hand, if all constraints (2) are active at the optimal solution ($\mu^* \geq 0$), which can be seen by the target as $r\bar{\mu}_i^* \geq 0$, then the private value $b_{\mathcal{A}}$ is uniquely determined from the output as $b_{\mathcal{A}} = Ax^*$.

Additionally, we have the following result about the privacy with respect to cost value $c_{\mathcal{A}}$.

Proposition IV.9. *The private value $c_{\mathcal{A}}$ cannot be uniquely determined from the outputs x^* and $r\bar{\mu}^*$ of the target node $\mathcal{T}$ if and only if $r\bar{\mu}_i^* > 0$ for some $i \in \{1, \dots, m\}$.*

The case $r\bar{\mu}_i^* > 0$ corresponds now to the case where the corresponding constraint is active at the optimal solution x^*. Intuitively when this fails, all constraints are inactive at the optimal solution x^* and they do not play a role. Hence we have an unconstrained quadratic problem in (1)-(2), i.e., the optimal solution satisfies the first order condition $Qx^* + c_{\mathcal{A}} = 0$, which reveals the value of $c_{\mathcal{A}}$.

To guarantee privacy with respect to both private values $b_{\mathcal{A}}, c_{\mathcal{A}}$ we take the conjunction of the two propositions.

Theorem IV.10. *The private values $b_{\mathcal{A}}$, $c_{\mathcal{A}}$ cannot be uniquely determined from the outputs x^* and $r\bar{\mu}^*$ of the target node $\mathcal{T}$ if $r\bar{\mu}_i^* < 0$ and $r\bar{\mu}_j^* > 0$ for some $i, j \in \{1, \dots, m\}$.*

Intuitively this happens when there are both active and inactive constraints at the optimal solution, confusing the target node. We also note that if the target node colludes with a subset of agents, learning their private input data, there is more privacy leakage than what is provided above for the target alone. The privacy guarantees against such coalitions become more complex and will be analyzed in future work. We note that the above privacy guarantees do not hold for arbitrary private data but depend on the actual quadratic problem instance. Future work involves the determination of algorithms with stronger privacy guarantees.

V. Conclusions

We have presented a privacy-preserving protocol for solving quadratic programs defined over private data of multiple agents. The proposed protocol employs the Pallier additive homomorphic cryptosystem, and its privacy guarantees are analyzed using the notion of zero-knowledge proof. We also examined conditions under which the output of the protocol, i.e., the optimal solution, does not completely reveal the private problem information to an adversarial party. Future work includes the development of privacy-aware protocols for general convex optimization problems.

References

[1] C. Dwork, "Differential privacy," in *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, ser. Lecture Notes in Computer Science, vol. 4052. Venice, Italy: Springer Verlag, July 2006, pp. 1–12.

[2] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Transactions on Automatic Control*, 2016, (To appear).

[3] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially Private Distributed Convex Optimization via Functional Perturbation," *ArXiv e-prints*, Dec. 2015.

[4] M. T. Hale and M. Egerstedt, "Differentially private cloud-based multi-agent optimization with constraints," in *American Control Conference (ACC), 2015*, July 2015, pp. 1235–1240.

[5] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, 2015, pp. 4:1–4:10.

[6] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *INFOCOM, 2011 Proceedings IEEE*, April 2011, pp. 820–828.

[7] Z. Xu and Q. Zhu, "Secure and resilient control design for cloud enabled networked control systems," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. ACM, 2015, pp. 31–42.

[8] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand, "A guide to fully homomorphic encryption," IACR Cryptology ePrint Archive (2015/1192), Tech. Rep., 2015.

[9] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 6836–6843.

[10] T. Fujita, K. Kogiso, K. Sawada, and S. Shin, "Security enhancements of networked control systems using rsa public-key cryptosystem," in *2015 10th Asian Control Conference (ASCC)*, May 2015, pp. 1–6.

[11] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *22nd Annual Network and Distributed System Security Symposium, NDSS*, 2015.

[12] E. Shi, R. Chow, T. h. Hubert Chan, D. Song, and E. Rieffel, "Privacy-preserving aggregation of time-series data," in *In NDSS*, 2011.

[13] Y. Shoukry, K. Gatsis, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada, "Cloud-based quadratic optimization using partial homomorphic encryption," available on arXiv.

[14] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, ser. EUROCRYPT'99, 1999, pp. 223–238.

[15] "Python-Paillier lib," http://python-paillier.readthedocs.org/en/latest/phe.html, accessed: 2016-3-12.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009.

[17] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2007.

[18] ——, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.