Cloud-Based Quadratic Optimization With Partially Homomorphic Encryption

Andreea B. Alexandru, *Student Member, IEEE*, Konstantinos Gatsis, *Member, IEEE*, Yasser Shoukry, *Member, IEEE*, Sanjit A. Seshia, *Fellow, IEEE*, Paulo Tabuada, *Fellow, IEEE*, and George J. Pappas, *Fellow, IEEE*

***Abstract*—This article develops a cloud-based protocol for a constrained quadratic optimization problem involving multiple parties, each holding private data. The protocol is based on the projected gradient ascent on the Lagrange dual problem and exploits partially homomorphic encryption and secure communication techniques. Using formal cryptographic definitions of indistinguishability, the protocol is shown to achieve computational privacy. We show the implementation results of the protocol and discuss its computational and communication complexity. We conclude this article with a discussion on privacy notions.**

***Index Terms*—Cryptography, data privacy, optimization.**

I. Introduction

The push toward increasing connectivity of devices is enabling control applications to span wide geographical areas. This increase in the number of available sensors and actuators and the amount of data has led to an increase in the controller's required computational capacity for global processing. One solution to this issue is to outsource the computations to powerful remote servers, generically called clouds. Cloud-based computation services offer the potential to aggregate and process information from a large number of agents, in domains such as machine learning, smart grids, and autonomous vehicles, with less overhead than distributed computations. This emerging Internet of things paradigm makes privacy a fundamental issue in many cloud-based applications due to the sensitive nature of the collected data. Recent examples such as data leakage and abuse by aggregating servers [1], [2] have drawn attention to the risks of storing data in the clear and urged measures against an untrustworthy cloud.

Computing on encrypted data is known as homomorphic encryption (HE). Fully HE allows the evaluation of Boolean functions over encrypted data and was introduced in [3] and further improved in, e.g., [4] and [5]. However, the computational overhead is prohibitive, due to the complexity of the cryptosystem primitives and size of the encrypted messages. Partially HE schemes are tractable but can support either only multiplications between encrypted data, such as in [6], or only additions between encrypted data, such as in [7].

HE has been recently used to design encrypted controllers, e.g. [8]–[12]. Several works have addressed gradient methods with partially HE [13]–[15] to solve unconstrained optimization problems. In contrast, our article focuses on adding constraints, which substantially complicate the optimization problem. Works such as [16] and [17] have proposed solutions with HE and secure multi-party computation for constrained distributed optimization problems, but treat the constraints evaluation differently than our article, in terms of data distribution, solution architecture, privacy requirements, and tools used. Examples such as [18] and [19] make use of differential privacy techniques in optimization algorithms, which follow different privacy guarantees than we consider.

We develop a new tractable protocol to privately solve centralized constrained quadratic optimization problems. To solve the optimization problem on encrypted data, we use an additively HE scheme, where, in short, *addition commutes with the encryption function*. The novelty is how to handle in a privacy-preserving manner the constraints, which introduce nonlinearities that cannot be supported by additively HE schemes. We show that a projected gradient method that operates on the Lagrange dual problem can alleviate this problem and can be run on encrypted data by exploiting communication between the participating parties. The main contributions of this article are the following.

1) We formally state and prove computational security guarantees for such a protocol.
2) We implement the protocol and show the computational and communication complexity produce reasonable running times.
3) We emphasize and analyze the difference between computational privacy and nonunique retrieval of the private data.

This article provides detailed security proofs, analyses, and implementations not available in the previous paper [20]. For a more detailed version and comparison with the solution in [21], the reader is directed toward the technical report [22].

Manuscript received September 16, 2019; revised January 23, 2020; accepted June 21, 2020. Date of publication June 30, 2020; date of current version April 26, 2021. This work was supported in part by the National Science Foundation (NSF) under Grant 1739816, Grant 1705135, and Grant CNS-1505799, in part by the Intel-NSF Partnership for Cyber Physical Security and Privacy, and in part by the Office of Naval Research under Grant N00014-17-1-2012. Recommended by Associate Editor Z. Chen. *(Corresponding author: Andreea B. Alexandru.)*

Andreea B. Alexandru and George J. Pappas are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: aandreea@seas.upenn.edu; pappasg@seas.upenn.edu).

Konstantinos Gatsis is with the Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, U.K. (e-mail: konstantinos.gatsis@eng.ox.ac.uk).

Yasser Shoukry is with the Department of Electrical Engineering and Computer Science, University of California, Irvine, CA 92697 USA (e-mail: yshoukry@uci.edu).

Sanjit A. Seshia is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA (e-mail: sseshia@eecs.berkeley.edu).

Paulo Tabuada is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: tabuada@ee.ucla.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2020.3005920

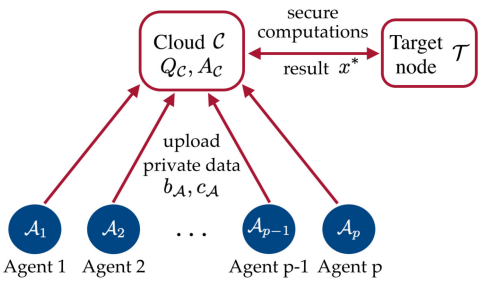

Fig. 1. Architecture of the problem: Agents are low-resource parties that have private data that they outsource to a powerful server, called the cloud. The cloud has to solve an optimization problem on the private data of the agents and send the result to a party called the target node.

II. Problem Setup

A. Motivating Examples

Quadratic optimization is a class of problems frequently employed in control systems design and operation. As a first example, consider estimating the state of a dynamical system from privacy-sensitive sensor measurements.

Let the system dynamics and sensor measurements be

$$x_{t+1} = A\,x_t + w_t, \quad y_t = C\,x_t + v_t \tag{1}$$

for $t = 0, \dots, T-1$, where w_t and v_t are the process and measurement noises. The system and sensor parameters $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{p\times n}$ can be known to the cloud, while the sensor measurements $y_0, \dots, y_{T-1}$ are privacy sensitive. The untrusted cloud has to collect the measurements and output an initial state estimate x_0, while maintaining the privacy of the sensor data and final output. A simple state estimate may be found as the solution to the least squares problem

$$\min_{x_0\in\mathbb{R}^n} \frac{1}{2}\sum_{t=0}^{T} \|y_t - CA^t\,x_0\|_2^2 = \min_{x_0\in\mathbb{R}^n} \frac{1}{2}\|y - \mathcal{O}\,x_0\|_2^2 \tag{2}$$

where $\mathcal{O} \in \mathbb{R}^{Tp\times n}$ is the system observability matrix. More general state estimation problems may also include constraints, e.g., the initial state lies in a private polyhedral set $Dx_0 \preceq b$.

As a second example, consider steering a linear dynamical system from a private initial position while guaranteeing safety constraints. The cloud has to compute a private model predictive controller. Such problems arise in vehicles that are deployed to explore hazardous environments or when different users compete against each other and want to hide their tactics.

B. Problem Statement

The above examples can be modeled as constrained quadratic optimization problems with distributed private data. We consider three types of parties involved in the problem: a number of agents $\mathcal{A}_i, i = 1, \dots, p$, a cloud server $\mathcal{C}$, and a target node $\mathcal{T}$. The purpose of this setup is to solve an optimization problem with the data provided by the agents and the computation performed at the cloud, and to send the result to the target node. The architecture is presented in Fig. 1.

Consider a strictly convex quadratic optimization problem

$$\begin{aligned} x^* = \arg\min_{x\in\mathbb{R}^n}\ & \frac{1}{2}x^\intercal\,Q_{\mathcal{C}}\,x + c_{\mathcal{A}}^\intercal x \\ \text{s.t. } & A_{\mathcal{C}}\,x \preceq b_{\mathcal{A}} \end{aligned} \tag{3}$$

which we assume to be feasible and where the variables and the parties to which they belong to are described as follows:

Agents $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_p)$: The agents are parties with low computational capabilities that possess the private information $b_{\mathcal{A}}$ and $c_{\mathcal{A}}$. The private information is decomposed across the agents as $b_{\mathcal{A}} = (b_1, \dots, b_p)$ and $c_{\mathcal{A}} = (c_1, \dots, c_p)$, with $b_i \in \mathbb{R}^{m_i}$ and $c_i \in \mathbb{R}^{n_i}$ being the private data of agent i such that $\sum_{i=1}^p m_i = m$ and $\sum_{i=1}^p n_i = n, i = 1, \dots, p$.

Cloud $\mathcal{C}$: The cloud is a party with high computational capabilities that knows the matrices $Q_{\mathcal{C}} \in \mathbb{S}_{++}^n$ and $A_{\mathcal{C}} \in \mathbb{R}^{m\times n}$.

Target node $\mathcal{T}$: The target node is a party with more computational capabilities than the agents that is interested in the optimal solution x^* of the problem.

In cloud applications, the service provider has to deliver the contracted service or otherwise the clients switch to another provider. Thus, the cloud will not alter the data it receives. Moreover, the agents' and target node's interest is to obtain the correct result from the service, so they will also not alter their data. However, the parties are not prohibited to locally process the data they receive. This model is known as semi-honest.

Definition 1 (Semi-honest model): A party is semi-honest if it does not deviate from the steps of the protocol but may store the transcript of the messages exchanged and process the data received in order to learn more information than stipulated.

The purpose of this article is to solve problem (3) using a secure multi-party computation protocol for semi-honest parties. The protocol takes as inputs the private data of the agents and the cloud's data, involves computing and exchanging messages, and eventually outputs the solution of the optimization problem to the target node. The protocol should guarantee *computational privacy*, as formally defined in the next section. We consider that all the data are represented on integers of l bits and comment on this further in Section IV-B.

III. Privacy Goals and Preliminaries

In what follows, $\{0,1\}^*$ defines a sequence of bits of unspecified length. Two sequences are computationally indistinguishable [23, Ch. 3], denoted by $\overset{c}{\equiv}$, if no efficient algorithm can distinguish between them. We use this concept when defining *two-party privacy*: a protocol privately computes a functionality if all information obtained by a party after the execution of the protocol (while also keeping a record of the intermediate computations) can be obtained only from the inputs and outputs available to that party.

Definition 2 (Two-party privacy w.r.t. semi-honest behavior [24, Ch. 7]): Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a functionality, and $f(x_1, x_2) = (f_1(x_1, x_2), f_2(x_1, x_2))$, for any inputs $x_1, x_2 \in \{0,1\}^*$. Let Π be a two-party protocol for computing f. The *view* of the ith party $(i = 1, 2)$ during an execution of Π on the inputs (x_1, x_2), denoted $V_i^{\Pi}(x_1, x_2)$, is $(x_i, coins, m_1, \dots, m_t)$, where *coins* represents the outcome of the ith party's internal coin tosses, and m_j represents the jth message it has received. For a deterministic functionality f, we say that Π *privately computes* f if there exist probabilistic polynomial-time (ppt) algorithms, called *simulators*, denoted S_i, such that

$$\{S_i(x_i, f_i(x_1, x_2))\}_{x_{1,2}\in\{0,1\}^*} \overset{c}{\equiv} \{V_i^{\Pi}(x_1, x_2)\}_{x_{1,2}\in\{0,1\}^*}.$$

For protocols that involve more than two parties, Definition 2 can be extended to *multi-party privacy*, by also taking into consideration the views of coalitions of semi-honest parties. We direct the reader to [22] and [24, Ch. 7].

We note that our privacy demands differ from [16], where input–output inference is used to assess what agents can infer about each other's data, while we require multi-party privacy.

Additively HE: Let $E(\cdot)$ define a generic encryption primitive, with domain the space of private data, called *plaintexts*, and codomain the space of encrypted data, called *ciphertexts*. For probabilistic encryption schemes, the encryption primitive also takes as input a random number. The decryption primitive $D(\cdot)$ is defined on the space of ciphertexts and takes values in the space of plaintexts. In additively homomorphic schemes, there exists an operator $\oplus$ defined on the space of ciphertexts so that

$$D(E(a) \oplus E(b)) = a + b \tag{4}$$

for any plaintexts a and b supported by the scheme. Formally, the decryption primitive $D(\cdot)$ is a homomorphism between the group of ciphertexts with the operator $\oplus$ and the group of plaintexts with addition $+$, which justifies the name of the scheme. Such a scheme also supports subtraction, by adding the additive inverse, and multiplication between an integer plaintext and an encrypted message.

In this article, we use the *Paillier cryptosystem* [7], which is a probabilistic asymmetric cryptosystem, with a public key used for the encryption of the private messages and disseminated publicly, and a private key known only to its owner, used for the decryption. In the Paillier cryptosystem, the messages are elements of the ring of integers modulo N, denoted by $\mathbb{Z}_N$, where N is a large integer of σ bits, called the Paillier modulus. The ciphertexts take values in the multiplicative group of integers modulo N^2, denoted by $\mathbb{Z}^*_{N^2}$. For a plaintext message a, we denote the Paillier encryption by $[[a]] := E(a)$, and throughout this article, we will use the following abstract notation for the operations on the encrypted space:

$$[[a]] \oplus [[b]] \stackrel{d}{=} [[a+b]], \quad b \otimes [[a]] \stackrel{d}{=} [[ba]] \tag{5}$$

for plaintexts $a, b \in \mathbb{Z}_N$, where $\stackrel{d}{=}$ means that the equality holds after applying the decryption primitive. We will use the same notation to denote encryptions and operations on vectors.

Under the assumption of decisional composite residuosity [7], the Paillier scheme is semantically secure (the formal definition can be found in [24, Ch. 5]), which means that an adversary that has the plaintext messages a and b cannot distinguish between the encryptions $[[a]]$ and $[[b]]$.

Symmetric encryption scheme: Symmetric key algorithms perform the encryption and decryption with the same key. The symmetric key cryptosystem that we use additively blinds a private value by noise. For messages of l bits, the key is generated as $sk \in \mathbb{Z}$ of length $\lambda + l$, where λ is the statistical security parameter. The encryption primitive is $E'(a) = a + sk$, with $a \in [0, 2^l) \cap \mathbb{Z}$, and the decryption is obtained as $a = D'(E'(a)) = E'(a) - sk$. The security of this scheme lies on generating a uniformly random key and on using this key for encryption only once, which yields that the distribution of $E'(a)$ is statistically indistinguishable from a random number sampled of $l + \lambda + 1$ bits. We also notice that *this symmetric cryptosystem commutes with the Paillier cryptosystem*.

IV. Secure Constrained Quadratic Optimization

For strongly convex problems, one can resort to duality theory [25, Ch. 5] to compute the projection on the feasible set and to retrieve the primal optimum from the optimal value of the dual problem. The dual of the optimization problem (3) is

$$\mu^* = \arg\max_{\mu \in \mathbb{R}^m} \; -\frac{1}{2}(A_{\mathcal{C}}^\intercal \mu + c_{\mathcal{A}})^\intercal Q_{\mathcal{C}}^{-1}(A_{\mathcal{C}}^\intercal \mu + c_{\mathcal{A}}) - \mu^\intercal b_{\mathcal{A}}$$
$$\text{s.t. } \mu \succeq 0.$$

The gradient of the dual objective function $g(\mu)$ is

$$\nabla g(\mu) = -A_{\mathcal{C}} Q_{\mathcal{C}}^{-1}(A_{\mathcal{C}}^\intercal \mu + c_{\mathcal{A}}) - b_{\mathcal{A}}. \tag{6}$$

Under standard constraint qualifications, e.g., Slater's condition [25, Ch. 5], strong duality between the primal and dual and the optimality conditions (Karush–Kuhn–Tucker—KKT) hold

$$Q_{\mathcal{C}} x^* + A_{\mathcal{C}}^\intercal \mu^* + c_{\mathcal{A}} = 0 \tag{7}$$
$$A_{\mathcal{C}} x^* - b_{\mathcal{A}} \preceq 0, \quad \mu^* \succeq 0 \tag{8}$$
$$\mu_i^*(a_i^\intercal x^* - b_i) = 0, \; i = 1, \ldots, m. \tag{9}$$

For strictly convex problems, i.e., $Q_{\mathcal{C}} \in \mathbb{S}^n_{++}$, the optimal solution of the primal problem can be obtained from (7) as $x^* = -Q_{\mathcal{C}}^{-1}(A_{\mathcal{C}}^\intercal \mu^* + c_{\mathcal{A}})$.

We use the projected gradient ascent algorithm to compute the optimum in problem (6). It is composed by iterations (10), where $\eta > 0$ is the step size and μ_{k+1} is the projected value of $\mu_k + \eta \nabla g(\mu_k)$ over the nonnegative orthant[1]

$$\mu_{k+1} = \max\{0, \mu_k + \eta \nabla g(\mu_k)\}. \tag{10}$$

A. Projected Gradient Ascent on Encrypted Data

As stated in Section II, we aim to solve an optimization problem outsourced to the cloud on private distributed data from the agents and to send the result to the target node. To protect the agents' data, we use an encryption scheme that allows the cloud to perform linear manipulations on encrypted data, as described in Section III. To this end, the target node generates a pair of keys $(pk_{\mathcal{T}}, sk_{\mathcal{T}})$ and distributes the public key to the agents and cloud, enabling them to encrypt their data.

Note that *for a quadratic objective function, only linear operations in the private data μ_k, $b_{\mathcal{A}}$, and $c_{\mathcal{A}}$ are required in order to compute the gradient (6)*. Hence, by taking advantage of the homomorphic property (5), the cloud can locally compute the gradient in the encrypted domain, for all $k = 0, \ldots, K-1$. A first challenge lies in performing the comparison with zero. Due to the modular arithmetic and probabilistic nature of the Paillier encryption scheme, *the order on the plaintext space is not preserved on the ciphertext space*, and comparison on encrypted data cannot be performed locally by the cloud. A second challenge is to then perform the update of the encrypted iterate (10) in a private way, so that the result of the maximum operation is not revealed to any of the parties involved.

Secure comparison protocol: To privately compute (10), we need to hide the comparison result between the updated iterate $\mu_k + \eta \nabla g(\mu_k)$ and zero from both the cloud and the target node. The comparison protocol described next is designed to reveal the result of the comparison to the target node. However, if we introduce an additional step (π is a random permutation on two elements)

$$[[a]], [[b]] \leftarrow \pi([[0]], [[\mu + \eta \nabla g(\mu)]]) \tag{11}$$

where the cloud randomizes the order of the two values that it wants to compare, then the target does not learn any information by knowing the result of the comparison.

The Damgård, Geisler, and Krøigaard (DGK) protocol for secure comparison of two private inputs of different parties was introduced in [26] and [27]. To this end, the authors proposed and used an additively HE scheme, which has the property that checking if an encrypted value is zero can be done more efficiently than simply decrypting the value. The authors prove the semantic security of the

[1] In (10) and in the rest of this article, when we refer to comparison of vectors, we mean elementwise comparison.

DGK cryptosystem under the hardness of factoring assumption. An extension of this protocol to the case where none of the parties knows the two numbers that have to be compared, which is of interest to us, was proposed in [28].

Let $\mathcal{C}$ have two encrypted values under the Paillier scheme, $[[a]]$ and $[[b]]$, obtained from (11), and let $\mathcal{T}$ have the decryption key. At the end, $\mathcal{T}$ will have the result of the comparison in the form of one bit t such that $(t = 1) \Leftrightarrow (a \leq b)$. Let l denote the number of bits of the unencrypted inputs a and b. The comparison protocol, that we call Protocol DGK, is based on the fact that the most significant bit of $(b - a + 2^l)$ is the bit that indicates if $(a \leq b)$. The security of this comparison protocol is proved in [26] and [28] and is based on the semantic security of the schemes used and on security of statistical blinding.

Secure update protocol: We need to ensure that when the cloud updates the value of the dual iterate at iteration $k + 1$ in (10), it does not know the new value. The solution is to have the cloud blind the values of $[[a]]$ and $[[b]]$ and send them to the target node in this order, which selects the value accordingly to the comparison result and then sends it back to the cloud. First, the blinding should be additive and effectuated with different random values. Second, a rerandomization of the encryptions should be performed so that the cloud cannot identify $[[a]]$ and $[[b]]$ from the received value. This can be done by adding an encryption of zero. Protocol 1 is the solution to the update problem. The intuition is that if $a \leq b$, then $t = 1$, and we obtain $\mu = \bar{b} - s = b$, and otherwise, $t = 0$ and we obtain $\mu = \bar{a} - r = a$.

Protocol for solving strictly convex quadratic problems: With these blocks, we build Protocol 2 that represents one iteration (10) of the dual projected gradient ascent. Line 3 ensures that the updated iterate has the required number of bits for the comparison protocol. This step is achieved by an exchange between the cloud and target node.

We finally assemble Protocol 3 that privately solves the constrained quadratic optimization problem (3) with private data and sends the optimal solution to the target node.

Protocol 1: Private Update of the Dual Variable.

Input: $\mathcal{C}$: $[[a]]$, $[[b]]$; $\mathcal{T}$: t such that $(t = 1) \Leftrightarrow (a \leq b)$
Output: $\mathcal{C}$: $[[\mu]]$

1: $\mathcal{C}$: choose two random numbers r, s
2: $\mathcal{C}$: $[[\bar{a}]] \leftarrow [[a]] \oplus [[r]]$, $[[\bar{b}]] \leftarrow [[b]] \oplus [[s]]$
3: $\mathcal{C}$: send $[[\bar{a}]]$ and $[[\bar{b}]]$ to $\mathcal{T}$
4: **if** $t = 0$ **then** $\mathcal{T}$: $[[v]] \leftarrow \widetilde{[[\bar{a}]]} = [[\bar{a}]] + [[0]]$
5: **else** $\mathcal{T}$: $[[v]] \leftarrow \widetilde{[[\bar{b}]]} = [[\bar{b}]] + [[0]]$
6: **end if** ▷ Rerandomize the ciphertext
7: $\mathcal{T}$: send $[[v]]$ and $[[t]]$ to $\mathcal{C}$
8: $\mathcal{C}$: $[[\mu]] \leftarrow [[v]] \oplus r \otimes [[t]] \oplus [[-r]] \oplus (-s) \otimes [[t]]$

Protocol 2: Private Iteration of the Dual Projected Gradient Ascent Method.

Input: $\mathcal{C}$: $A_{\mathcal{C}} \in \mathbb{R}^{m \times n}$, $Q_{\mathcal{C}} \in \mathbb{S}^n_{++}$, $[[b_{\mathcal{A}}]]$, $[[c_{\mathcal{A}}]]$, η, $[[\mu_k]]$; $\mathcal{T}$: $sk_{\mathcal{T}}$
Output: $\mathcal{C}$: $[[\mu_{k+1}]]$

1: $\mathcal{C}$: $[[\nabla g(\mu_k)]] \leftarrow (-A_{\mathcal{C}} Q_{\mathcal{C}}^{-1} A_{\mathcal{C}}^{\intercal}) \otimes [[\mu_k]] \oplus (-A_{\mathcal{C}} Q_{\mathcal{C}}^{-1}) \otimes [[c_{\mathcal{A}}]] \oplus (-1) \otimes [[b_{\mathcal{A}}]]$ ▷ Compute the encrypted gradient as in (6)
2: $\mathcal{C}$: $[[\bar{\mu}_k]] \leftarrow [[\mu_k]] \oplus \eta \otimes [[\nabla g(\mu_k)]]$ ▷ Update iterate value
3: $\mathcal{C}, \mathcal{T}$ truncate $[[\bar{\mu}_k]]$ to l bits
4: $\mathcal{C}$ executes (11): $\mathcal{C}$ gets $[[a_k]]$, $[[b_k]]$ ▷ Randomize inputs
5: $\mathcal{C}, \mathcal{T}$ execute Protocol DGK elementwise on inputs $[[a_k]]$, $[[b_k]]$: $\mathcal{T}$ gets t_k ▷ Secure comparison protocol
6: $\mathcal{C}, \mathcal{T}$ execute Protocol 1: $\mathcal{C}$ obtains $[[\mu_{k+1}]]$ ▷ Secure update protocol that ensures $\mu_{k+1} = \max\{\bar{\mu}_k, 0\}$

B. Fixed-Point Arithmetic

The optimization problem (3) is defined on real variables, whereas the Paillier encryption scheme is defined on integers. To address this issue, we adopt a fixed-point arithmetic setting. We consider a value having l_i bits for the integer part and l_f bits for the fractional part. Therefore, by multiplying the real values by 2^{l_f} and truncating the result, we obtain integers. We choose $l = l_i + l_f$ large enough such that the loss in accuracy is negligible and assume that there is no overflow. For ease of exposition, we consider this data processing done implicitly in the protocols described. The random numbers used for blinding the sensitive values (namely, in Protocols DGK and 1) are sampled uniformly from integers of $l + \lambda$ bits, where λ is the statistical security parameter, as already explained in Section II. In order to guarantee correctness of the comparison protocol, we must impose $\log_2 N > l + \lambda + 1$.

V. Privacy of Quadratic Optimization Protocol

Theorem 1: Protocol 3 achieves privacy with respect to Definition 2 for noncolluding parties.

The intuition for the proof is as follows. Consider an iteration of the gradient ascent in Protocol 3, i.e., Protocol 2. First, two Paillier ciphertexts are computationally indistinguishable to a party that does not own the decryption key. Second, the exchanges between the cloud and the target are additively blinded using a different random number uniformly sampled from a large enough range, which make the blinded messages statistically indistinguishable from random numbers. Third, the ciphertexts are refreshed after each exchange, so a party cannot infer information about the encrypted values by simply comparing the ciphertexts. Then, none of the parties can infer the magnitude or the sign of the private variables. Furthermore, we show that privacy is not broken by running an iteration multiple times. We prove that storing the exchanged messages does not give any new information on the private data using similar arguments. The detailed proof is given in Appendix A.

The definition of privacy is naturally extended in the multi-party case: even under collusions, the protocol securely computes the functionality of solving a quadratic optimization problem. No further information is revealed than what can be inferred from the coalition's inputs and outputs. However, if all agents and the cloud collude or if the cloud colludes with the target node, then they have access to all the information in the system, in which case the above result is rather vacuous. Hence, we only consider coalitions between a strict subset of the agents and the cloud or the target node.

Theorem 2: Protocol 3 achieves privacy against coalitions.

The proof of Theorem 2 is given in the technical report [22].

VI. Privacy Discussion

In this section, we discuss privacy aspects that differ from the computational notions of Definition 2. Even if the execution of a protocol does not leak anything, the known information in a coalition (e.g., the output and some of the inputs) can be used to infer the rest of the private inputs. Specifically, in our optimization problem, the private variables and the optimal solution are coupled via the optimality conditions (7)–(9), which are public knowledge, irrespective of the protocol.

Consider the following definition that concerns the retrieval of private data from adversarial/known data.

Protocol 3: Privacy Preserving Algorithm for Solving Strictly Convex Quadratic Optimization Problems.

Input: $\mathcal{A}_{i=1,\dots,p}$: $b_{\mathcal{A}} = \{b_j\}_{j=1,\dots,m}, c_{\mathcal{A}} = \{c_j\}_{j=1,\dots,n}$; $\mathcal{C}$: $A_{\mathcal{C}} \in \mathbb{R}^{m\times n}, Q_{\mathcal{C}} \in \mathbb{S}^n_{++}, \eta > 0, K$; $\mathcal{T}$: $sk_{\mathcal{T}}, K$

Output: $\mathcal{T}$: x^*

1: **for** i=1,...,p **do**
2: $\mathcal{A}_i$: encrypt the private information $msg_i \leftarrow ([[b_i]], [[c_i]])$
3: $\mathcal{A}_i$: send the encrypted messages to $\mathcal{C}$
4: **end for**
5: $\mathcal{C}$: Construct the vectors $[[b_{\mathcal{A}}]]$ and $[[c_{\mathcal{A}}]]$ from the messages
6: $\mathcal{C}$: $\eta \leftarrow 1/\lambda_{max}(A_{\mathcal{C}} Q_{\mathcal{C}}^{-1} A_{\mathcal{C}}^\intercal)$
7: $\mathcal{C}$: Choose a random positive initial value μ_0 for the dual variable and encrypt it: $[[\mu_0]]$
8: **for** each $k = 0, \dots, K-1$ **do**
9: $\mathcal{C}, \mathcal{T}$ execute Protocol 2: $\mathcal{C}$ gets $[[\mu_{k+1}]]$ ▷ $\mathcal{C}, \mathcal{T}$ securely effectuate an iteration of the dual projected gradient ascent
10: **end for**
11: $\mathcal{C}$: $[[x^*]] \leftarrow (-Q_{\mathcal{C}}^{-1} A_{\mathcal{C}}^\intercal) \otimes [[\mu_K]] \oplus (-Q_{\mathcal{C}}^{-1}) \otimes [[c_{\mathcal{A}}]]$ and send it to $\mathcal{T}$ ▷ Compute the primal optimum from the dual optimum
12: $\mathcal{T}$: Decrypt $[[x^*]]$ and output x^*

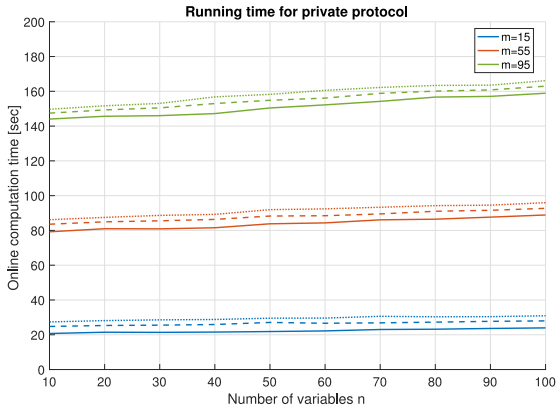

Fig. 2. Average running times of Protocol 3 for problem instances with the number of variables on the abscissa and the number of constraints in the legend. The full lines correspond to no communication delay, the dashed lines correspond to a 10-ms delay, and the dotted lines to a 20-ms delay. The simulation is run for 30 iterations, a 1024-bit key, and 32-bit messages, with 16-bit precision. The statistical parameter for additive blinding is 100 bits.

Definition 3 (Nonunique retrieval): Let p be the private inputs of a problem and let an algorithm $A(p)$ solve that problem. Let $\mathcal{K}$ be some adversarial knowledge, which can contain public information, some private information and the output of algorithm A for the adversary, denoted by $A^{\mathcal{K}}(p)$. We say p cannot be uniquely retrieved by the adversary if there exists a set $\mathcal{U}$, such that $p \in \mathcal{U}$, $|\mathcal{U}| \geq 2$, and $\forall p' \in \mathcal{U}$, $A^{\mathcal{K}}(p) = A^{\mathcal{K}}(p')$.

Definition 2 imposes stronger requirements but is not concerned with the output–input relation, whereas Definition 3 also explores what inputs of a coalition can reveal about the inputs of the honest parties.

In what follows, we carry out an algebraic analysis on a black-box protocol that, given the agents' private data $b_{\mathcal{A}}, c_{\mathcal{A}}$ and the cloud's matrices $Q_{\mathcal{C}}, A_{\mathcal{C}}$, outputs the solution x^* of Problem (3) to the target node. We provide conditions such that a coalition cannot uniquely determine unknown private inputs in the sense of Definition 3. This analysis applies to Protocol 3, which, assuming it runs for sufficient iterations, outputs the desired result x^* to the target node.

Suppose without loss of generality that a coalition between $\bar{p}$ agents ($1 \leq \bar{p} < p$) has access to the elements $b_1, \dots, b_{\bar{m}}$ with $0 \leq \bar{m} \leq m$, and $c_1, \dots, c_{\bar{n}}$ with $0 \leq \bar{n} \leq n$. Then, let us define the decomposition of the matrix $A_{\mathcal{C}}$ as

$$A_{\mathcal{C}} = \left[\begin{array}{c|c} A_1 & \\ \hline A_{21} & A_{22} \end{array}\right] \tag{12}$$

with $A_1 \in \mathbb{R}^{\bar{m}\times n}$, $A_{21} \in \mathbb{R}^{(m-\bar{m})\times \bar{n}}$, and $A_{22} \in \mathbb{R}^{(m-\bar{m})\times(n-\bar{n})}$.

Proposition 1: Consider a protocol solving Problem (3) and a coalition between the target node and agents with access to $\bar{m}$ of the values of $b_{\mathcal{A}}$ and $\bar{n}$ of the values of $c_{\mathcal{A}}$. Suppose that the cost and constraint matrices $A_{\mathcal{C}}$ and $Q_{\mathcal{C}}$ are public. Then, we have the following.

1) If $\bar{m} < m$ and there exists a vector $\delta \in \mathbb{R}^{m-\bar{m}}$ such that $\delta \neq 0, \delta \succeq 0$ and $A_{21}^\intercal \delta = 0$, then the coalition cannot uniquely retrieve the value of $b_{\mathcal{A}}$.
2) If additionally $\bar{n} < n$ and $A_{22}^\intercal \delta \neq 0$, then the coalition cannot uniquely retrieve the value of $c_{\mathcal{A}}$.

Proposition 2: Consider a protocol solving Problem (3) and a coalition between the cloud and agents with access to $\bar{m}$ of the values of $b_{\mathcal{A}}$ and $\bar{n}$ of the values of $c_{\mathcal{A}}$. Then, a coalition that satisfies $\bar{m} < m$ and $\bar{n} < n$ cannot uniquely retrieve the values of $b_{\mathcal{A}}, c_{\mathcal{A}}$, and x^*.

The proof of Proposition 1 is given in Appendix B. The proof of Proposition 2 follows from the fact that $Q_{\mathcal{C}}$ is a positive-definite matrix and $A_{\mathcal{C}}$ does not have columns or rows of zeros. A coalition between the cloud and the $\bar{p} < p$ agents cannot solve (3), as it lacks the data to define it, so it cannot uniquely retrieve x^* and the rest of the agents' private data.

VII. Implementation

The efficiency of a secure multi-party protocol is measured in the complexity of the computations performed by each party, as well as in the number of rounds of communication. While the former is relevant from the perspective of the level of computational power required, the latter is relevant when the communication network is unreliable.

The agents are low-power platforms and are only required to effectuate one encryption and send one message, but the cloud and the target node are platforms with higher computational capabilities. Protocol 3 involves $\mathcal{O}(K)$ plaintext matrix-encrypted vector products and $\mathcal{O}(Km)$ encrypted scalar sums performed at the cloud, $\mathcal{O}(Kml)$ DGK decryptions and encryptions, $\mathcal{O}(Km)$ Paillier decryptions and encryptions, and $\mathcal{O}(Km)$ encrypted scalar sums and plaintext scalar-encrypted scalar products at the target node, with $\mathcal{O}(K)$ batches of messages communicated in-between.

The tradeoff between privacy and communication emerges when adding an artificial delay of 10 ms (respectively, 20 ms) to simulate the delays that can occur in communication networks. Because the blinded communication between the parties is added to ensure privacy, the communication delay impacts more a fully private protocol than a less private one.

We implemented the protocol proposed in Section IV-A in Python 3 and ran it on a 2.2-GHz Intel Core i7 processor [29]. Fig. 2 depicts the average online running time of Protocol 3, for random instances of the data in Problem (3), with 0-, 10-, and 20-ms delay for communication, run for 30 iterations of the gradient ascent algorithm, for ease of comparison.

APPENDIX

A. Proof of Theorem 1

In what follows, we successively discuss the views of each type of party participating in the protocol: agents, cloud and target node. We avoid mentioning the public key $pk_{\mathcal{T}}$, number of iterations K and number of bits l in the views, since they are public. We denote by $\mathcal{I}$ the inputs of all the parties

$$\mathcal{I} = \{b_{\mathcal{A}}, c_{\mathcal{A}}, A_{\mathcal{C}}, Q_{\mathcal{C}}, sk_{\mathcal{T}}\}.$$

Proposition A.1: Protocol 2 is secure in the semi-honest model, according to Definition 2.

The proof of Proposition A.1 is contained in the proof of Theorem 1 and resembles the argmax protocol in [30].

1) Simulator for Agent $\mathcal{A}_i$: Agent $\mathcal{A}_i$, $i = 1, \ldots, p$, has inputs $I_{\mathcal{A}_i} = (\{b_j\}_{j=1,\ldots,m_i}, \{c_j\}_{j=1,\ldots,n_i})$ and view

$$V_{\mathcal{A}_i}(\mathcal{I}) := (b_j, c_j, [[b_j]], [[c_j]], \text{coins})$$

where coins represent the random values used for encryption.

The agents are only online to send their encrypted data to the cloud, and they do not have any role and output afterward. A simulator $S_{\mathcal{A}_i}$ simply generates the random values necessary to encrypt its inputs, and output the view obtained. We denote by $\widetilde{\cdot}$ the quantities of the simulator, which are different than the quantities of the agents, but follow the same distribution

$$S_{\mathcal{A}_i} := (b_j, c_j, \widetilde{[[b_j]]}, \widetilde{[[c_j]]}, \widetilde{\text{coins}}).$$

It follows that the protocol is secure with respect to the agents.

Next, we construct a sequence of algorithms to obtain that the views of the cloud and the target node after the execution of K iterations are the same as the view of simulators that simply execute K iterations with random exchanged messages. For ease of exposition, we will treat μ, $b_{\mathcal{A}}$, and $c_{\mathcal{A}}$ as scalars. The same steps are repeated for every element in the vectors.

2) Simulator for the Cloud $\mathcal{C}$: The view of the cloud during the execution of lines 5 and 6 is

$$V_{\mathcal{C}}^{-1}(\mathcal{I}) = (A_{\mathcal{C}}, Q_{\mathcal{C}}, \eta, [[b_{\mathcal{A}}]], [[c_{\mathcal{A}}]], [[\mu_0]], \text{coins}) =: I_{\mathcal{C}}^{-1}.$$

Furthermore, we construct the view of the cloud at iteration $k = 0, \ldots, K-1$ during the execution of an instance of Protocol 2, on the inputs of all parties: the inputs $\mathcal{I}$ and the data the parties had at iteration $k-1$. We denote the view of the cloud at iteration $k-1$ by $I_{\mathcal{C}}^{k-1}$ which, along with $\mathcal{I}$ and the view of the target node at iteration $k-1$, denoted by $I_{\mathcal{T}}^{k-1}$ (15), will be the input of the view at iteration k

$$\bar{\mathcal{I}}^{k-1} := \mathcal{I} \cup I_{\mathcal{C}}^{k-1} \cup I_{\mathcal{T}}^{k-1} \tag{13}$$

$$I_{\mathcal{C}}^{k} := V_{\mathcal{C}}^{k}(\bar{\mathcal{I}}^{k-1}) = \Bigg(I_{\mathcal{C}}^{k-1}, [[\mu_k]], [[\bar{\mu}_k]], \underbrace{\pi_k, \text{coins}_{1k}}_{(11)}$$

$$\underbrace{\rho_k, [[t_k]], \text{m}^{\text{comp}_k}, \text{coins}_{2k}}_{\text{Protocol DGK}}, \underbrace{r_k, s_k, [[v_k]], [[\mu_{k+1}]], \text{coins}_{3k}}_{\text{Protocol 1}} \Bigg)$$

where m^{comp_k} are messages exchanged in the DGK protocol, and coins_{jk}, for $j = 1, 2, 3$, are the random numbers generated in the corresponding protocol. Finally, the view of the cloud after the execution of line 11 in Protocol 3 is

$$V_{\mathcal{C}}^{K}(\bar{\mathcal{I}}^{K-1}) = \left(I_{\mathcal{C}}^{K-1}, [[x^*]]\right). \tag{14}$$

Therefore, the view of the cloud during the whole execution of Protocol 3 is $V_{\mathcal{C}}(\mathcal{I}) := V_{\mathcal{C}}^{K}(\bar{\mathcal{I}}^{K-1})$.

We first construct a simulator on the inputs $I_{\mathcal{C}} = \{A_{\mathcal{C}}, Q_{\mathcal{C}}\}$ that mimics $V_{\mathcal{C}}^{-1}(\mathcal{I})$:

1) generate $n+m$ random numbers of l bits $\widetilde{b_{\mathcal{A}}}, \widetilde{c_{\mathcal{A}}}$;
2) generate a random positive initial value $\widetilde{\mu_0}$;
3) generate $n+m+1$ uniformly random numbers for the Paillier encryption and denote them $\widetilde{\text{coins}}$;
4) compute $\widetilde{[[b_{\mathcal{A}}]]}, \widetilde{[[c_{\mathcal{A}}]]}, \widetilde{[[\mu_0]]}$;
5) compute η following line 6;
6) output $S_{\mathcal{C}}^{-1}(I_{\mathcal{C}}) = (A_{\mathcal{C}}, Q_{\mathcal{C}}, \widetilde{[[b_{\mathcal{A}}]]}, \widetilde{[[c_{\mathcal{A}}]]}, \eta, \widetilde{[[\mu_0]]}, \widetilde{\text{coins}}) =: \widetilde{I}_{\mathcal{C}}^{-1}$.

Proposition A.1 states that there exists a ppt simulator for the functionality of Protocol 2 on inputs $(A_{\mathcal{C}}, Q_{\mathcal{C}}, [[b_{\mathcal{A}}]], [[c_{\mathcal{A}}]], \eta, [[\mu_k]])$ and output $[[\mu_{k+1}]]$. However, we need to show that we can simulate the functionality of consecutive calls of Protocol 2 or, equivalently, of one call of Protocol 2 but on the augmented input that contains the data of the cloud in the previous iterations. Denote such a simulator $S_{\mathcal{C}}^{k}$, which on the input $I_{\mathcal{C}}^{k-1}$ mimics $V_{\mathcal{C}}^{k}(\bar{\mathcal{I}}^{k-1})$ in (15), for $k = 0, \ldots, K-1$:

1) Compute $[[\nabla g(\mu_k)]]$ and $[[\bar{\mu}_k]]$ as in lines 1 and 2 of Protocol 2 from $[[\mu_k]]$, $[[b_{\mathcal{A}}]]$, and $[[c_{\mathcal{A}}]]$ which are included in $I_{\mathcal{C}}^{k-1}$.
2) Generate a random permutation $\widetilde{\pi_k}$ and apply it on $([[0]], [[\bar{\mu}_k]])$ as in (11).
3) Follow Protocol DGK and replace the messages by encryptions of random values to obtain $\widetilde{\rho_k}, \widetilde{\text{m}^{\text{comp}_k}}$.
4) Generate random bit $\widetilde{t_k}$ and its encryption $[[\widetilde{t_k}]]$.
5) Generate random values $\widetilde{r_k}$ and $\widetilde{s_k}$ as in line 1 in Protocol 1 and their encryptions $[[\widetilde{r_k}]], [[\widetilde{s_k}]]$.
6) Obtain $[[\widetilde{v_k}]]$ by choosing between the elements of $\widetilde{\pi_k}([[0]], [[\bar{\mu}_k]]) + (\widetilde{r_k}, \widetilde{s_k})$ according to the generated $\widetilde{t_k}$.
7) Compute $\widetilde{[[\mu_{k+1}]]}$ as in line 8 of Protocol 1.
8) Denote the rest of the random values used for encryption and blinding by $\widetilde{\text{coins}_k}$.
9) Output $S_{\mathcal{C}}^{k}(I_{\mathcal{C}}^{k-1}) = (I_{\mathcal{C}}^{k-1}, [[\bar{\mu}_k]], [[\widetilde{\pi_k}]], [[\widetilde{z_k}]], \widetilde{\text{m}^{\text{comp}_k}}, [[\widetilde{t_k}]], [[\widetilde{r_k}]], [[\widetilde{s_k}]], [[\widetilde{v_k}]], \widetilde{[[\mu_{k+1}]]}, \widetilde{\text{coins}_k}) =: \widetilde{I}_{\mathcal{C}}^{k}$.

Finally, a simulator $S_{\mathcal{C}}^{K}(I_{\mathcal{C}}^{K-1}) =: \widetilde{I}_{\mathcal{C}}^{K}$ for $V_{\mathcal{C}}^{K}(\bar{\mathcal{I}}^{K-1})$ is obtained by simply performing line 11 on the inputs.

Proposition A.2: $S_{\mathcal{C}}^{k}(I_{\mathcal{C}}^{k-1}) \overset{c}{\equiv} V_{\mathcal{C}}^{k}(\bar{\mathcal{I}}^{k-1})$, for $k = -1, \ldots, K$, where $I_{\mathcal{C}}^{-2} := I_{\mathcal{C}}$.

The proof is given in the technical report [22]. Thus, we obtained that $I_{\mathcal{C}}^{k} \overset{c}{\equiv} \widetilde{I}_{\mathcal{C}}^{k}$, for $k = -1, \ldots, K$.

Corollary A.1: $S_{\mathcal{C}}^{k}(\widetilde{I}_{\mathcal{C}}^{k-1}) \overset{c}{\equiv} S_{\mathcal{C}}^{k}(I_{\mathcal{C}}^{k-1})$, which is equivalent to $S_{\mathcal{C}}^{k+1}(S_{\mathcal{C}}^{k}(I_{\mathcal{C}}^{k-1})) \overset{c}{\equiv} S_{\mathcal{C}}^{k+1}(V_{\mathcal{C}}^{k}(\bar{\mathcal{I}}^{k-1}))$, for $k = 0, \ldots, K-1$.

Finally, we construct a simulator $S_{\mathcal{C}}(I_{\mathcal{C}})$ for the execution of Protocol 3 and show that its view is computationally indistinguishable from $V_{\mathcal{C}}(\mathcal{I})$. To this end, we define the following sequence of views:

$$\begin{aligned}
V_{\mathcal{C}}(\mathcal{I}) = H_{-1}(\mathcal{I}) &= V_{\mathcal{C}}^{K}(\bar{\mathcal{I}}^{K-1}) \\
H_0(\mathcal{I}) &= S_{\mathcal{C}}^{K} \circ V_{\mathcal{C}}^{K-1}(\bar{\mathcal{I}}^{K-2}) \\
H_1(\mathcal{I}) &= S_{\mathcal{C}}^{K} \circ S_{\mathcal{C}}^{K-1} \circ V_{\mathcal{C}}^{K-2}(\bar{\mathcal{I}}^{K-3}) \\
&\vdots \\
H_K(\mathcal{I}) &= S_{\mathcal{C}}^{K} \circ S_{\mathcal{C}}^{K-1} \circ \ldots \circ S_{\mathcal{C}}^{0} \circ V_{\mathcal{C}}^{-1}(\mathcal{I}) \\
S_{\mathcal{C}}(I_{\mathcal{C}}) = H_{K+1}(I_{\mathcal{C}}) &= S_{\mathcal{C}}^{K} \circ S_{\mathcal{C}}^{K-1} \circ \ldots \circ S_{\mathcal{C}}^{0} \circ S_{\mathcal{C}}^{-1}(I_{\mathcal{C}}).
\end{aligned}$$

By transitivity, H_{-1} and H_{K+1} are computationally indistinguishable if

$$H_{-1} \overset{c}{\equiv} H_0 \overset{c}{\equiv} \cdots \overset{c}{\equiv} H_k \overset{c}{\equiv} H_{k+1} \overset{c}{\equiv} H_{k+2} \overset{c}{\equiv} \cdots \overset{c}{\equiv} H_{K+1}.$$

This result follows from induction on Corollary A.1. In conclusion, we obtain that $S_{\mathcal{C}}(I_{\mathcal{C}}) \stackrel{c}{\equiv} V_{\mathcal{C}}(\mathcal{I})$, which verifies that Protocol 3 achieves privacy with respect to the cloud.

3) Simulator for the Target Node $\mathcal{T}$: The input and output of the target node in Protocol 3 are $I_{\mathcal{T}} = (sk_{\mathcal{T}}, x^*)$. The view of the target node during iteration k is

$$I_{\mathcal{T}}^k := V_{\mathcal{T}}^k(\bar{\mathcal{I}}^{k-1}) = (sk_{\mathcal{T}}, \underbrace{z_k, t_k, \mathrm{m}^{\mathrm{comp}_k}, \mathrm{coins}_{2k}}_{\text{Protocol DGK}} \underbrace{\bar{a}_k, \bar{b}_k, v_k, \mathrm{coins}_{3k}}_{\text{Protocol 1}}). \tag{15}$$

The view of the target node during the last step is

$$V_{\mathcal{T}}^K(\bar{\mathcal{I}}^{K-1}) = (I_{\mathcal{T}}^{K-1}, [[x^*]]). \tag{16}$$

As before, the view of the target node during the execution of Protocol 3 is $V_{\mathcal{T}}(\mathcal{I}) := V_{\mathcal{T}}^K(\bar{\mathcal{I}}^{K-1})$.

In order to construct a simulator for the target node, we show that the target node is not capable of inferring new relevant information about the private data although it has access to the data from multiple iterations and to the optimal solution x^*. Apart from the last message, which is the encryption of the solution $[[x^*]]$, and the comparison results t_k, all the values the target node receives are blinded with different values at each iteration. The target node knows that $Q_{\mathcal{C}}x^* = A_{\mathcal{C}}^{\intercal}\mu_K - c_{\mathcal{A}}$ and $\mu_K = v_K - \bar{r}_K$, for some random value $\bar{r}_K$. However, although it has access to x^* and v_K, it cannot infer any new information about $c_{\mathcal{A}}$, thanks to the randomization that blinds μ_K. Due to the random permutation $a_{K-1}, b_{K-1} = \pi_{K-1}(0, \bar{\mu}_{K-1})$, the target node cannot identify the sign of $\bar{\mu}_{K-1}$ and magnitude of μ_K, even if it knows t_{K-1}. Therefore, having access to $x^* = x_K$ does not bring any information about the values in the intermediary steps.

We now investigate the relation between the messages from consecutive iterations. From Protocol DGK, $\mathcal{T}$ receives z_k, which is the additively blinded value of $b_k - a_k + 2^l$, and other blinded values, denoted in (15) as $\mathrm{m}^{\mathrm{comp}_k}$. Provided the blinding noises are refreshed at every iteration, $\mathcal{T}$ cannot infer any information about $b_k - a_k$, as follows from Section III. Furthermore, from the update Protocol 1, $\mathcal{T}$ knows the values of v_k and t_k, but not of $\pi_k, \bar{\mu}_k, r_k, s_k$, and π_{k+1}. Then, $\mathcal{T}$ cannot construct v_{k+1} from v_k. This guarantees that an integer $\tilde{v}_k^i$ selected uniformly at random from $[2^{l+\lambda}, 2^{l+\lambda+1})$ will have the distribution statistically indistinguishable from v_k^i. Moreover, the target cannot retrieve μ_{k+2} from v_k and v_{k+1}. Similar arguments hold for the blinded messages in Protocol DGK.

We now build a simulator $S_{\mathcal{T}}$ that applies the steps of the protocol on randomly generated values. Proposition A.1 states that there exists a ppt simulator for the functionality of Protocol 2 on $I_{\mathcal{T}}^{-1} = \{sk_{\mathcal{T}}\}$. However, we need to show that we can simulate the functionality of consecutive calls of Protocol 2, or, equivalently, of one call of Protocol 2 but on inputs $(I_{\mathcal{T}}^k, x^*)$. Denote such a simulator $S_{\mathcal{T}}^k$, which on the inputs $(I_{\mathcal{T}}^k, x^*)$ should output a view that is statistically indistinguishable from $V_{\mathcal{T}}^k(\bar{\mathcal{I}}^{k-1})$ in (15), for $k = 0, \ldots, K-1$. We already showed that although the target node knows the output x^* and the messages from all the iterations of Protocol 3, it cannot extract information from them or correlate the messages to the iteration they arise from, so x^* is only relevant for $S_{\mathcal{T}}^K$.

1) Generate a $\lambda + l$ bits random integer $\widetilde{\rho_k}$, add 2^l and get $\widetilde{z_k}$.
2) Generate a random bit $\widetilde{t_k}$.
3) Choose a random bit $\widetilde{\delta_{\mathcal{T}}}$. If it is 0, generate l nonzero random values, else generate $l-1$ nonzero random values and one 0 value. Denote them by $\widetilde{\mathrm{m}^{\mathrm{comp}_k}}$ (for Protocol DGK [28]).
4) Generate random integers of length $l + \lambda + 1$ $\widetilde{\bar{a}_k}$ and $\widetilde{\bar{b}_k}$.
5) Compute $\widetilde{v_k}$ according to $\widetilde{t_k}$.
6) Denote all Paillier and DGK generated coins by $\widetilde{\mathrm{coins}}$.
7) Output $S_{\mathcal{T}}^k(I_{\mathcal{T}}^{k-1}, x^*) = (I_{\mathcal{T}}^{k-1}, \widetilde{z_k}, \widetilde{t_k}, \widetilde{\mathrm{m}^{\mathrm{comp}_k}}, \widetilde{\bar{a}_k}, \widetilde{\bar{b}_k}, \widetilde{v_k}, \widetilde{\mathrm{coins}}) =: \widetilde{I}_{\mathcal{T}}^k$.

Finally, a simulator $S_{\mathcal{T}}^K(I_{\mathcal{T}}^{K-1}, x^*) =: \widetilde{I}_{\mathcal{T}}^K$ for $V_{\mathcal{T}}^K(\bar{\mathcal{I}}^{K-1})$ is obtained by simply generating an encryption of x^* and outputting: $(I_{\mathcal{T}}^{K-1}, \widetilde{[[x^*]]})$.

Proposition A.3: $S_{\mathcal{T}}^k(I_{\mathcal{T}}^{k-1}, x^*) \stackrel{c}{\equiv} V^k(\bar{\mathcal{I}}^{k-1})$, $k = 0, \ldots, K$.

The proof is given in the technical report [22]. Thus, we obtained that $\widetilde{I}_{\mathcal{T}}^k \stackrel{c}{\equiv} I_{\mathcal{T}}^k$, for $k = 0, \ldots, K$.

Corollary A.2: $S_{\mathcal{T}}^k(\widetilde{I}_{\mathcal{T}}^{k-1}, x^*) \stackrel{c}{\equiv} S_{\mathcal{T}}^k(I_{\mathcal{T}}^{k-1}, x^*)$, which is equivalent to $S_{\mathcal{T}}^{k+1}(S_{\mathcal{T}}^k(I_{\mathcal{T}}^{k-1}, x^*), x^*) \stackrel{c}{\equiv} S_{\mathcal{T}}^{k+1}(V^k(\bar{\mathcal{I}}^{k-1}), x^*)$, for $k = 1, \ldots, K-1$.

Finally, we construct a simulator $S_{\mathcal{T}}(I_{\mathcal{T}})$ for the execution of Protocol 3 and we show that its view is statistically indistinguishable from $V_{\mathcal{T}}(\mathcal{I})$. To this end, we define the following sequence, from which we drop the input x^* to the simulators to not overburden the notation:

$$\begin{aligned}
V_{\mathcal{T}}(\mathcal{I}) = H_0(\mathcal{I}) &= V_{\mathcal{T}}^K(\bar{\mathcal{I}}^{K-1}) \\
H_1(\mathcal{I}, x^*) &= S_{\mathcal{T}}^K \circ V_{\mathcal{T}}^{K-1}(\bar{\mathcal{I}}^{K-2}) \\
H_2(\mathcal{I}, x^*) &= S_{\mathcal{T}}^K \circ S_{\mathcal{T}}^{K-1} \circ V^{K-2}(\bar{\mathcal{I}}^{K-3}) \\
&\vdots \\
H_K(\mathcal{I}, x^*) &= S_{\mathcal{T}}^K \circ S_{\mathcal{T}}^{K-1} \circ \cdots \circ S^1 \circ V_{\mathcal{T}}^0(\mathcal{I}) \\
S_{\mathcal{T}}(I_{\mathcal{T}}) = H_{K+1}(I_{\mathcal{T}}) &= S_{\mathcal{T}}^K \circ S_{\mathcal{T}}^{K-1} \circ \cdots \circ S^1 \circ S_{\mathcal{T}}^0(I_{\mathcal{T}}).
\end{aligned}$$

By transitivity, H_0 and H_{K+1} are computationally indistinguishable if

$$H_0 \stackrel{c}{\equiv} H_1 \stackrel{c}{\equiv} \cdots \stackrel{c}{\equiv} H_k \stackrel{c}{\equiv} H_{k+1} \stackrel{c}{\equiv} H_{k+2} \stackrel{c}{\equiv} \cdots \stackrel{c}{\equiv} H_{K+1}.$$

The result follows from induction on Corollary A.2. In conclusion, we obtain that $S_{\mathcal{T}}(I_{\mathcal{T}}) \stackrel{c}{\equiv} V_{\mathcal{T}}(\mathcal{I})$, which verifies that Protocol 3 achieves privacy with respect to the target node.

The proof of Theorem 1 is now complete. ■

B Proof of Proposition 1

The coalition has access to the following data, which is fixed: $A_{\mathcal{C}}, Q_{\mathcal{C}}, x^*, \{b_i\}_{i=1,\ldots,\bar{m}}, \{c_i\}_{i=1,\ldots,\bar{n}}$.

Proof of 1): We address two cases: the nonstrict satisfaction and the equality satisfaction of the constraints.

I) Suppose that there exists a solution $(\mu, \{b_i\}_{i=\bar{m}+1,\ldots,m}, \{c_i\}_{i=\bar{n}+1,\ldots,n})$ to the KKT conditions such that $a_i^{\intercal}x^* < b_i$ for some $\bar{m}+1 \le i \le m$. In particular, this implies that $\mu_i = 0$. Then define $c' := c_{\mathcal{A}}$, $\mu' := \mu$ and b' such that $b'_j := b_j$ for all $j \neq i$ and $b'_i \ge a_i^{\intercal}x^*$. The new set of points $(\mu,' b,' c')$ is also a solution to the KKT conditions, by construction.

II) Suppose that there exists a solution $(\mu, \{b\}_{i=\bar{m}+1,\ldots,m}, \{c\}_{i=\bar{n}+1,\ldots,n})$ to the KKT conditions such that $a_j^{\intercal}x^* = b_j$ for all $j = \bar{m}+1, \ldots, m$. Consider that there exists a vector δ that satisfies $\delta \succeq 0$ and $A_{21}^{\intercal}\delta = 0$. Compute $\epsilon \ge 0$ as

$$\epsilon = \min_{\delta_k > 0, k=\bar{m}+1,\ldots,m} (\mu_k/\delta_k).$$

Then, construct $\mu' := \mu - \epsilon[\,0\ \delta^{\intercal}\,]^{\intercal}$ that satisfies $\mu' \succeq 0$ and $\mu'_i = 0$ for some $\bar{m}+1 \le i \le m$ that is the argument of the above minimum. Furthermore, define $c' := c_{\mathcal{A}} + \epsilon[\,0\ \delta^{\intercal}A_{22}\,]^{\intercal}$ and b' such that $b'_j := b_j$ for all $j \neq i$ and b'_i to be any value $b'_i > b_i$. Then, $(b,' c,' \mu')$ is also

a solution to the KKT conditions. More specifically, the complementarity slackness condition holds for all $j = \bar{m}+1, \ldots, m, j \neq i$: $\mu'_j\ (a_j^\intercal x^* - b'_j) = \mu'_j(a_j^\intercal x^* - b_j) = 0$ and $\mu'_i(a_i^\intercal x^* - b'_i) = 0$. Then, we can check the gradient condition

$$Q_{\mathcal{C}} x^* + A_{\mathcal{C}}^\intercal \mu' + c' = Q_{\mathcal{C}} x^* + A_{\mathcal{C}}^\intercal \mu + c_{\mathcal{A}} = 0.$$

Hence, $b' \neq b_{\mathcal{A}}$ satisfies the KKT conditions, and the coalition cannot uniquely determine $b_{\mathcal{A}}$.

Proof of 2): Consider a solution $(\mu, \{b\}_{i=\bar{m}+1,\ldots,m}, \{c\}_{i=1,\bar{n}+1,\ldots,n})$ to the KKT conditions. For some $\epsilon > 0$, define $\mu' := \mu + \epsilon[\,0\ \delta^\intercal\,]^\intercal$ and $c' := c_{\mathcal{A}} - \epsilon[\,0\ \delta^\intercal A_{22}\,]^\intercal$. Define b' such that for all j, it holds that $b'_j = a_j^\intercal x^*$. Then, $(\mu,' b,' c')$ is also a solution to the KKT conditions. Specifically, it follows that $\mu' \succeq 0$. Moreover, the complementarity slackness condition holds by construction of b', and as before, the gradient condition holds. Hence, $c' \neq c_{\mathcal{A}}$ satisfies the KKT solution, and the coalition cannot uniquely determine $c_{\mathcal{A}}$. ■

Acknowledgment

A. B. Alexandru would like to thank Brett Hemenway for helpful discussions.

References

[1] D. A. Fernandes, L. F. Soares, J. V. Gomes, M. M. Freire, and P. R. Inácio, "Security issues in cloud environments: A survey," *Int. J. Inf. Secur.*, vol. 13, no. 2, pp. 113–170, 2014.

[2] "List of data breaches," [Online]. Available: http://wikipedia.org/wiki/List_of_data_breaches, Accessed on: Jan. 26, 2019.

[3] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.

[4] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 24–43.

[5] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM J. Comput.*, vol. 43, no. 2, pp. 831–871, 2014.

[6] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. Workshop Theory Appl. Cryptographic Techn.*, 1984, pp. 10–18.

[7] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.

[8] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *Proc. IEEE 54th Conf. Decis. Control*, 2015, pp. 6836–6843.

[9] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Eng. Pract.*, vol. 67, pp. 13–20, 2017.

[10] J. Kim *et al.*, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016.

[11] M. S. Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted MPC for linear constrained systems," *IEEE Control Syst. Lett.*, vol. 2, no. 2, pp. 195–200, Apr. 2018.

[12] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based MPC with encrypted data," in *Proc. IEEE 57th Conf. Decis. Control*, 2018, pp. 5014–5019.

[13] S. Han, W. K. Ng, L. Wan, and V. C. Lee, "Privacy-preserving gradient-descent methods," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 6, pp. 884–899, Jun. 2010.

[14] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv:1711.10677v1*.

[15] C. Zhang, M. Ahmad, and Y. Wang, "ADMM based privacy-preserving decentralized optimization," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 565–580, Mar. 2019.

[16] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. 314–325, 2018.

[17] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 724–738.

[18] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Trans. Autom. Control*, vol. 62, no. 1, pp. 50–64, Jan. 2017.

[19] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via functional perturbation," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 1, pp. 395–408, Mar. 2018.

[20] A. B. Alexandru, K. Gatsis, and G. J. Pappas, "Privacy preserving cloud-based quadratic optimization," in *Proc. IEEE 55th Allerton Conf. Commun., Control, Comput.*, 2017, pp. 1168–1175.

[21] Y. Shoukry *et al.*, "Privacy-aware quadratic optimization using partially homomorphic encryption," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 5053–5058.

[22] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshia, P. Tabuada, and G. J. Pappas, "Cloud-based quadratic optimization with partially homomorphic encryption," 2018, *arXiv:1809.02267v2*.

[23] O. Goldreich, *Foundations of Cryptography: Basic Tools*, vol. 1. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[24] O. Goldreich, *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[25] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[26] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2007, pp. 416–430.

[27] I. Damgård, M. Geisler, and M. Krøigaard, "A correction to "Efficient and secure comparison for on-line auctions"," *Int. J. Appl. Cryptography*, vol. 1, no. 4, pp. 323–324, 2009.

[28] T. Veugen, "Improving the DGK comparison protocol," in *Proc. IEEE Workshop Inf. Forensics Secur.*, 2012, pp. 49–54.

[29] [Online]. Available: https://github.com/andreea-alexandru/QPHE

[30] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 4325–4338.