Private Computation of Polynomials over Networks

Hosseinalizadeh, Teimour; Turkmen, Fatih; Monshizadeh, Nima

Published in:
2021 60th IEEE Conference on Decision and Control (CDC)

DOI:
10.1109/CDC45484.2021.9683378

Publication date:
2022

Private Computation of Polynomials over Networks

Teimour Hossienalizadeh[a], Fatih Turkmen[b], Nima Monshizadeh[a]

[a]*Engineering and Technology Institute, University of Groningen, The Netherlands*
[b]*Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen,The Netherlands*

Abstract

This study concentrates on preserving privacy in a network of agents where each agent seeks to evaluate a general polynomial function over the private values of her immediate neighbors. We provide an algorithm for the exact evaluation of such functions while preserving privacy of the involved agents. The solution is based on a reformulation of polynomials and adoption of two cryptographic primitives: Paillier as a Partially Homomorphic Encryption scheme and multiplicative-additive secret sharing. The provided algorithm is fully distributed, lightweight in communication, robust to dropout of agents, and can accommodate a wide class of functions. Moreover, system theoretic and secure multi-party conditions guaranteeing the privacy preservation of an agent's private values against a set of colluding agents are established. The theoretical developments are complemented by numerical investigations illustrating the accuracy of the algorithm and the resulting computational cost.

Keywords: Privacy, cryptography, polynomials, networked systems, multiagent systems

1. Introduction

Emerging distributed systems such as smart grids, intelligent transportation, and smart buildings provide better scalability, fault tolerance, and resource sharing compared to traditional centralized systems. A distributed dynamical system rely on peer-to-peer data exchange between individual agents. The agents wish to protect their data from being revealed since the data can contain sensitive information or can be leveraged for disrupting the system(see the case for smart metering in [1]). Therefore preserving privacy of agents in distributed dynamical systems is of crucial concern.

To preserve privacy in dynamical systems, differential privacy is a popular approach that was introduced to control system for private filtering through [2], applied to average consensus [3, 4], distributed optimization [5], plug-and-play control [6] and studied for its relation to input observability [7]. In general, however, it introduces a trade-off between privacy level and control performances, and also possible vulnerability of data disclosure through the least significant bit of the perturbed data [8]. System theory also provides solutions for preserving privacy in dynamical systems, see [9, 10, 11] in this context. Even though these solutions do not generally degrade the performance of controllers and are lightweight in computation, they are problem specific and their privacy guarantees are weaker compared to differential privacy based methods.

As another class of privacy preserving techniques, cryptography based methods have proven to be useful since they provide stronger security guarantees while maintaining an acceptable level of performance during the computation. Among the cryptographic schemes, Homomorphic Encryption (HE) schemes offer a certain appeal for control systems since they are computationally less demanding compared to other alternatives such as Garbled Circuits. HE refers to a special family of cryptosystems that supports elementary mathematical operations (such as additions or multiplications) to be executed on encrypted data. In particular, a party can encrypt its private values and outsource computations on those encrypted values to an untrusted data processing entity without allowing access to associated private values [12].

We can classify the literature of encrypted control systems, into two main categories: The first one is the typical setup of an isolated system and a cloud, where a designed policy is evaluated by the cloud using the encrypted data generated by the system. In this group and among the first known studies of encrypted control in system theory literature, we can refer to implementation of static state feedback controllers. In [13], the privacy of both controller parameters and system states are preserved by using RSA and ElGamal encryption schemes. The proposed method; however, requires the system to be heavily involved in the computation procedures. This problem is resolved in [14] by employing Paillier's scheme as a Partially HE(PHE) where only the privacy of system states are preserved. As for the nonlinear state feedback, in [15] a framework of two non-colluding clouds and Paillier's scheme are used to preserve the privacy of system states and controller parameters. Deployment of a linear dynamic controller over a cloud is investigated in [16] by employing Fully HE(FHE). Different from a static controller, the essence of recursion in a dynamic controller causes the finite-time life span problem, for which the necessity of integer coefficients in [17], and refreshment of the controller state in [18] are proposed as possible solutions. Outsourcing

Email addresses: t.hosseinalizadeh@rug.nl (Teimour Hossienalizadeh), f.turkmen@rug.nl (Fatih Turkmen), n.monshizadeh@rug.nl (Nima Monshizadeh)

the calculation of computationally demanding controllers such as the implicit model predictive controller to another party is also investigated in [19], where the privacy problem is solved mainly by Paillier's scheme.

The second class, which this study also belongs to, is related to multiagent systems where the aim is to preserve privacy of the involved agents. Compared to the first class, the communication topology and the presence of different agents (parties) impose extra constraints on achieving this goal. In this group, authors in [20] use Paillier and weight decomposition to preserve privacy in the first order consensus problems, where the solution is extended to the second order problems in [21]. Their proposed method is suitable when an agent's objective is to evaluate an affine function of her immediate neighbors and is restricted to consensus problems. In [22] a solution is proposed for privacy preserving distributed averaging using Paillier's cryptosystem for a directed graph where nodes, after receiving a public key from a trusted entity, continue computations till an agreed-upon time-step at which the private key is revealed to all parties. In the context of distributed optimization, authors in [23] use a symmetric FHE scheme (SingleMod encryption) and a third party to preserve privacy of the involved agents that are interested in evaluating polynomial functions of private variables. The existence of a central non-colluding third party poses this question that whether it is possible to solve the problem in a centralized instead of a distributed way. In other words, an FHE scheme and the third party allow the designer of the optimization scheme to devise a centralized algorithm.

The problem of privacy in cooperative linear controllers in a network system is considered in [24] and for solving it Paillier's scheme is incorporated. The proposed solution, however, reveals information about rate of changes in private values of each agent. This issue is resolved in [25] by combining additive secret sharing with Paillier's scheme. The problem is viewed more generally as a private weighted sum aggregation and discussed further in [26]. We refer the interested readers to [27] for an overview of the recent applications of cryptography in dynamical systems.

In this work, we consider the problem of privacy in the networked systems where each agent's goal is to evaluate a general polynomial of her neighboring agents' private values. Our solution is based on two cryptographic primitives: the Paillier encryption technique and secret sharing. Paillier encryption is a public key partially homomorphic encryption (PHE) [28] which allows us to evaluate the sum of two values of plaintext using their ciphertext and secret sharing enables us to distribute shares of a secret among agents in the network. The main contributions of this study are as follows:[1]

(1) The current work extends the class of computed functions from affine [26], which is customary in linear averaging protocols to general polynomials, i.e. each agent's target function is a polynomial function of her neighbors' state variable. This extension, particularly the products of the state variables, substantially complicates the problem and requires a careful analysis to ensure that no privacy-sensitive information is leaked throughout the computation. (2) Our algorithm is fully distributed and hence does not require an external party to evaluate the polynomial functions [23]. As a consequence of (1) and (2), we can accommodate fully distributed nonlinear, yet polynomial, coupling dynamics in networked systems. (3) Our proposed solution is robust to dropout of an agent and lightweight in communication due to the adopted schemes from cryptography. (4) We establish conditions for privacy preservation of an agent with respect to a set of colluding agents for the proposed algorithm using both cryptography and system theory paradigms.

The rest of the paper is organized as follows: In Section 2, we present necessary cryptographic tools for the paper; Section 3 includes a motivating example for polynomials and formulates the problem of preserving privacy for these functions, and Section 4 provides a solution for the problem. Privacy analysis of the proposed method is investigated in Section 5; numerical results are provided in Section 6, and finally, the paper closes with conclusions in Section 7.

2. Notations and preliminaries

The sets of positive integer, nonnegative integer, integer, rational and real numbers are denoted by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$, respectively. We denote the identity matrix of size n by I_n and we write $[n] := \{0, 1, 2, \dots, n\}$ for any $n \in \mathbb{N}$. We assume a network of V agents represented by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V} = \{1, 2, \dots, V\}$ and edge set $\mathcal{E}$ given by a set of unordered pairs $\{i, j\}$ of distinct nodes i and j. We denote the set of neighbors of node i by $\mathcal{N}_i := \{j \in \mathcal{V} : \{i, j\} \subseteq \mathcal{E}\}$, and $\overline{\mathcal{N}}_i := \mathcal{N}_i \cup i$. The cardinality of $\mathcal{N}_i$ denoted by $d_i := |\mathcal{N}_i|$ is equal to the degree of node i. Without loss of generality we consider the state variable of each agent i as a scalar $x_i \in \mathbb{R}$; the extension to $x_i \in \mathbb{R}^{n_i}$, $n_i \geq 2$ is straightforward. We collect the state variables of all agents as $x := \mathrm{col}(x_1, x_2, \dots, x_V) = [x_1^\top, \dots, x_V^\top]^\top$, the state variables of all agents except for agent i as $x_{-i} := \mathrm{col}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_V)$ and the state variables of agent i's neighbors as $x_{\mathcal{N}_i} := \mathrm{col}(x_j)_{j \in \mathcal{N}_i}$.

2.1. Cryptography primitives

The Paillier encryption scheme consists of three steps (Gen, Enc, Dec). 1) Gen: Given the bit-length (l), generates (N, P, Q) where $N = PQ$ and P and Q are randomly selected l-bit primes, 2) Enc: Given public key pk $= N$ and a message $m \in \mathbb{Z}_N$, pick uniform $r \leftarrow \mathbb{Z}_N^*$ and output $c := [(1+N)^m . r^N \mod N^2]$ as ciphertext, 3) Dec: Given the secret key $\mathrm{sk} = \phi(N) = (P-1)(Q-1)$ and the ciphertext c computes $m := \left[\frac{[c^{\phi(N)} \mod N^2]-1}{N} . \phi(N)^{-1} \mod N\right]$. Paillier encryption scheme is chosen plaintext attack secure based on hardness of decisional composite residuosity problem [30, p. 495-496]. It is easy to see that for any plaintext m_1 and m_2 and their respective encryptions c_1 and c_2, we have $\mathsf{Dec}(c_1 \cdot c_2) = m_1 + m_2$, i.e. the Paillier scheme is an additively HE also known as a Partially

[1] Preliminary results of this work are presented in [29]. Different from the conference article, this document presents distributed secret sharing using pseudorandom functions, considers all polynomial coefficients as private, investigates robustness to agent dropouts, provides a formal proof for Theorem 1, analyzes privacy from a system theoretic perspective (see section 5), and provides a motivating example as well as a new case study.

Homomorphic Encryption(PHE). We denote the encryption of a value m by agent i's public key pk_i as $[\![m]\!]_{\mathrm{pk}_i}$.

In (t, n)-threshold secret sharing, a third party wants to share a secret s among some set of n agents $a_1, a_2, \ldots, a_n$ by giving each one a share in such a way that only t or more users can reconstruct the secret. In other words, no coalition of fewer than t agents should get any information about s from their collective shares. When $t = n$ and s is l bit length $s \in \{0,1\}^l$, the third party chooses $s_1, \ldots, s_{n-1} \in \{0,1\}^l$ uniformly and sets $s_n = s \oplus (\oplus_{i=1}^{n-1} s_i)$, where $\oplus$ denotes bitwise exclusive [30, p. 501-502]. The share of agent a_i is s_i.

Paillier scheme only accepts nonnegative integers $\mathbb{N}_0$ in its domain while we are interested in computations over $\mathbb{R}$. Therefore, an encoding-decoding scheme satisfying homogeneity and additivity conditions is incorporated. This scheme first discretizes a value $v \in \mathbb{R}$ to $\hat{v} \in \mathbb{Q}$, then maps it to an integer $z \in \mathbb{Z}$ by choosing an appropriate scale $L \in \mathbb{N}$, then to a nonnegative integer using $m = z \mod \Omega$, where Ω is a sufficiently large number. This process is invertible, i.e. given $0 \le m \le \Omega$ we can recover $\hat{v} \in \mathbb{Q}$. To simplify the notation, we do not distinguish between the encoded value and the original real value of a quantity in modular operations.

3. Motivating example and problem formulation

We begin with an example motivating privacy considerations in computation of polynomials in networks. The problem formulation will be stated afterwards.

3.1. Motivating example

Control of networked systems and optimization on networks heavily rely on computation of functions over state variables of neighboring agents. As mentioned earlier, the focus of this work is on private computation of polynomial functions. As a case in point, we provide an example in the context of game theoretic algorithms.

Consider a group of $V = |\mathcal{V}|$ players (agents) that seek a generalized Nash equilibrium (GNE) of a noncooperative game with globally shared affine constraints [31]. Feasible decision set of all players is $X := \prod_{i=1}^{V} \Gamma_i \cap \{x \in \mathbb{R}^n : \sum_{i=1}^{V} A_i x_i \ge \sum_{i=1}^{V} b_i\}$ where x_i takes values from a local admissible set $\Gamma_i \subseteq \mathbb{R}^{n_i}$, and $A_i \in \mathbb{R}^{m \times n_i}$, $b_i \in \mathbb{R}^m$ are local parameters of player i. In this game, each player i aims at minimizing her local cost function $J_i(x_i, x_{-i})$ subject to her feasible decision set $X_i(x_{-i}) := \{x_i \in \Gamma_i : (x_i, x_{-i}) \in X\}$, i.e.,

$$\min_{x_i \in \mathbb{R}^{n_i}} J_i(x_i, x_{-i}) \quad \text{s.t. } x_i \in X(x_{-i}). \tag{1}$$

A distributed GNE seeking algorithm is proposed in [31] where at step 1 of this algorithm, each player $i \in \mathcal{V}$ updates her decision at time index k as

$$x_i(k+1) = \mathrm{proj}_{\Gamma_i}[x_i(k) - \tau_i(\nabla_{x_i} J_i(x_i(k), x_{-i}(k)) - A_i^\top \lambda_i(k))], \tag{2}$$

where $\mathrm{proj}_{\Gamma_i}(\cdot)$ is the Euclidean projection operator onto the set Γ_i, λ_i is the Lagrange multiplier, and τ_i is the step size of player i.

The gradient of the cost function $\nabla_{x_i} J_i(x_i, x_{-i})$ is generally a nonlinear function of decision variables of other players x_{-i}. Therefore, player i in the game generally needs the value of x_j with $j \in \mathcal{V}$ for running (2). Putting it differently, player j must share her decision variable x_j with player i. Sharing the decision variable x_j over time can reveal information on the cost function $J_j(x_j, x_{-j})$ which includes privacy sensitive parameters of player j.

In the case of quadratic cost functions, $\nabla_{x_i} J_i(x_i, x_{-i})$ is affine and hence the scheme developed in [26] can be used to evaluate it privately. Moreover, in the context of aggregative games, [32] and [33] have proposed noncryptographic based solutions in order to preserve privacy of decision variables. The quadratic costs and aggregative functions in the mentioned studies are special cases of the polynomial functions that we consider here. In this work, we propose a cryptography-based algorithm that enables a private computation of polynomials over networks, thereby preserve privacy for a broad range of nonlinear cost functions appearing in network game-theoretic and optimization problems. The choice of studying polynomial functions is further motivated by the fact that any function continuous on a closed bounded set can be approximated by a polynomial with a desired accuracy (see the Stone-Weierstrass approximation theorem, e.g. [34, p. 123].)

3.2. Problem formulation

We consider a scenario where at each time index $k \in [K]$ agent $i \in \mathcal{V}$ in the network $\mathcal{G}$ is interested in evaluating a $d \in \mathbb{N}$ degree polynomial $\mathcal{P}_i(x_i, x_{\mathcal{N}_i}) : \mathbb{R} \times \mathbb{R}^{|\mathcal{N}_i|} \to \mathbb{R}$; namely,

$$\mathcal{P}_i(x_i(k), x_{\mathcal{N}_i}(k)) := \sum_{(p_1,p_2,p_3,\ldots,p_m) \in \mathcal{X}_i} c_{(p_1 p_2 p_3 \ldots p_m)} x_1^{p_1}(k) x_2^{p_2}(k) \ldots x_m^{p_m}(k) \tag{3}$$

where $c_{(\cdot)} \in \mathbb{R}$ and

$$\mathcal{X}_i := \{(p_1, p_2, p_3, \ldots, p_m) \in \mathbb{N}_0^m : p_1 + p_2 + \ldots + p_m \le d, \; j \notin \overline{\mathcal{N}_i} \Rightarrow p_j = 0\}.$$

As we can see (3) depends not only on agent i's state variable x_i but also on the state variable of her neighbors x_j with $j \in \mathcal{N}_i$. We identify private and public values of the agents as: Private value of agent i is $\mathsf{Pv}_i := \{x_i, c_{(\cdot)}\}$ that includes her state variable x_i and all the coefficients $c_{(\cdot)}$ in (3), private value of agent j is $\mathsf{Pv}_j := \{x_j\}$ with $j \in \mathcal{N}_i$, and public values of agent i is the exponent of state variables in (3), i.e., $\mathcal{X}_i$. Notice that the agent i shares the public values with agents $\mathcal{N}_i$.

We now provide two privacy assumptions that clarify the adopted setup in our problem formulation.

Assumption 1 (Honest-but-curious)**.** Agents in a network $\mathcal{G}$ are honest-but-curious, also known as semi-honest, meaning that they follow the required protocol for interacting with other agents but are also interested in determining the private values in the network.

Assumption 2 (Passive Adversary)**.** An adversary $\mathcal{A}$ is probabilistic polynomial-time, passive, and communications among agents are done in her presence. The adversary $\mathcal{A}$ can be an

agent in the network or an external party observing the communication.

Our aim is to provide a *privacy* preserving protocol for the *exact* evaluation of (3) for agent i. That is to say, only agent i should be able to obtain the accurate value of $\mathcal{P}_i$ without revealing her own private value Pv_i to any other agent j and without gaining any privacy-sensitive information about Pv_j other than the target function $\mathcal{P}_i$.

4. Proposed algorithm

The solution we provide is based on PHE and secret sharing techniques. In particular, we use Paillier's scheme to protect the privacy of Pv_i, and secret sharing for preserving the privacy of Pv_j, with $j \in \mathcal{N}_i$. For adopting these schemes in our privacy preserving algorithm, we rewrite the polynomial (3) in a new from by making a distinction between its bivariate and multivariate terms. Namely, we write (3) as

$$\mathcal{P}_i(x_i(k), x_{\mathcal{N}_i}(k)) = \Big(\sum_{j\in\mathcal{N}_i} P_{ij}(x_i(k), x_j(k)) \Big) + Q_i(x_i(k), x_{\mathcal{N}_i}(k)) \tag{4}$$

where

$$P_{ij}(x_i(k), x_j(k)) = \sum_{p_i,p_j} c_{p_ip_j} x_i^{p_i}(k) x_j^{p_j}(k),$$

with $c_{p_ip_j} \in \mathbb{R}$, and $Q_i(\cdot,\cdot)$ contains the terms with at least two state variables from $x_{\mathcal{N}_i}$. Notice that P_{ij} is the summation of bivariate terms in (3), with x_i and x_j, $j \in \mathcal{N}_i$, being the corresponding two variables.

As will be observed, our algorithm leverages additive and multiplicative secret sharing to preserve the privacy of neighbors of i in the evaluation of (4). The additive secrets will be primarily used to evaluate the bivariate terms in (4) whereas the multiplicative secret sharing is exploited to evaluate the multivariate terms, i.e $Q_i(\cdot,\cdot)$. Motivated by this and to minimize the required communication in the protocol, we write Q_i as a summation of multiplicative terms, namely:

$$Q_i = \sum_{t=1}^{T} Q_i^t, \qquad Q_i^t(x_i(k), x_{\mathcal{N}_i}(k)) := \prod_{j\in\overline{\mathcal{N}}_i} W_j^t(x_j(k)) \tag{5}$$

with $W_j^t = \sum_{q_j} c_{q_j}^{(t)} x_j^{q_j}$, $Q_i^t(\cdot,\cdot) \neq 0$, $T \in \mathbb{N}$, and $c_{q_j}^{(t)} \in \mathbb{R}$. Note that W_j^t is a univariate polynomial of x_j.

Example 1. As an example assume that we have a network with $\{\{1,2\},\{1,3\},\{1,4\}\} \subseteq \mathcal{E}$ where agent 1 is interested in the evaluation of the following polynomial:

$$\mathcal{P}_1(x_1,x_2,x_3,x_4) = 2x_1^2x_2 + 3x_1x_3 + 4x_1x_4^3 + x_1x_2^2x_3^2x_4 + 3x_1x_2^2x_3x_4. \tag{6}$$

Based on the representation (4), we can specify bivariate parts as $P_{12} = 2x_1^2x_2$, $P_{13} = 3x_1x_3$, $P_{14} = 4x_1x_4^3$. To write the multivariate parts, after factoring out the term $x_1x_2^2x_4$, we obtain $W_1^1 = x_1$, $W_2^1 = x_2^2$, $W_3^1 = x_3^2 + 3x_3$, and $W_4^1 = x_4$. ◊

Private and public values in (4) remain the same as in (3); yet take the form $\mathsf{Pv}_i := \{x_i, c_{p_ip_j}, c_{q_j}^{(t)}\}$ and (p_j, q_j) for private and public values of agent i, respectively, and $\mathsf{Pv}_j := \{x_j\}$ with $j \in \mathcal{N}_i$.

4.1. Distributed secret sharing

As mentioned before, we use secret sharing for preserving the privacy of x_j, $j \in \mathcal{N}_i$, throughout computation of $\mathcal{P}_i$. In particular, additive secret sharing is used in the bivariate part (namely, P_{ij}) and multiplicative secret sharing over additively secret shared data is used in the multivariate terms in $\mathcal{P}_i$ (namely, Q_i).

To mask intermediate computations [35], we require that every agent $j \in \overline{\mathcal{N}}_i$ have shares of addition $s_j^{\mathbf{a}}(k)$ and multiplication $s_j^{\mathbf{m}}(k)$ for all $i \in \mathcal{V}$ and for all $k \in [K]$ such that

$$\prod_{j\in\overline{\mathcal{N}}_i} s_j^{\mathbf{m}}(k) \equiv 1 \mod \Omega \tag{7a}$$

$$\sum_{j\in\overline{\mathcal{N}}_i} s_j^{\mathbf{a}}(k) \equiv 0 \mod \Omega, \tag{7b}$$

where Ω is a publicly known and sufficiently large prime number and $s_j^{\mathbf{m}}(k) \neq 0$. The shares $s_j^{\mathbf{a}}(k)$ and $s_j^{\mathbf{m}}(k)$ are selected uniformly randomly from the following set:

$$\mathbb{Z}_\Omega = \{0, 1, \ldots, \Omega - 1\}. \tag{8}$$

It should be noted that based on the Fermat's little theorem every nonzero element in (8) has a multiplicative inverse, meaning $(\forall \omega \neq 0 \in \mathbb{Z}_\Omega)(\exists \omega^{-1} \in \mathbb{Z}_\Omega)$ such that $(\omega\omega^{-1} \equiv 1 \mod \Omega)$, therefore choosing $s_j^{\mathbf{a}}(k)$, $s_j^{\mathbf{m}}(k)$ in the required form (7) is feasible [36, p. 63].

We note that although the "secrets" (0 and 1) are known to the agents, the generation of the shares $s_j^{\mathbf{a}}$ and $s_j^{\mathbf{m}}$ is analogous to sharing of a secret s explained in Subsection 2.1, and as such, we occasionally refer to this scheme as secret sharing (see also [26] where a similar terminology is used for additive secret sharing).

Next, we generate the additive and multiplicative shares of the agents in a fully distributed manner. To this end, every agent $j \in \overline{\mathcal{N}}_i$ selects uniformly randomly $s_{jh}^{\mathbf{m}}(k)$ and $s_{jh}^{\mathbf{a}}(k)$ from (8) such that

$$\prod_{h\in\overline{\mathcal{N}}_i} s_{jh}^{\mathbf{m}}(k) \equiv 1 \mod \Omega \tag{9a}$$

$$\sum_{h\in\overline{\mathcal{N}}_i} s_{jh}^{\mathbf{a}}(k) \equiv 0 \mod \Omega, \tag{9b}$$

for $\forall k \in [K]$, where $i \in \mathcal{V}$. Then agent j sends $s_{jh}^{\mathbf{m}}(k)$ and $s_{jh}^{\mathbf{a}}(k)$ for $h \in \overline{\mathcal{N}}_i \backslash j$ to agent i through a secure communication channel, where agent i then sends each share to its corresponding receiver, agent h. After this step, agent j obtains the addition and multiplication shares as follows:

$$s_j^{\mathbf{m}}(k) := \prod_{h\in\overline{\mathcal{N}}_i} s_{hj}^{\mathbf{m}}(k) \bmod \Omega \tag{10a}$$

$$s_j^{\mathbf{a}}(k) := \sum_{h\in\overline{\mathcal{N}}_i} s_{hj}^{\mathbf{a}}(k) \bmod \Omega. \tag{10b}$$

Notice that the distributed shares $s_j^{\mathbf{a}}(k)$ and $s_j^{\mathbf{m}}(k)$ obtained in (10a) and (10b) satisfy the relations (7a) and (7b), respectively.

4.2. Distributed secret sharing using pseudorandom functions

Exchanging $|\mathcal{N}_i| \times |\mathcal{N}_i|$ random numbers in (9a) and (9b) for every time index k among the agents imposes extra communication loads in the network. This drawback can be circumvented using the idea of Pseudorandom Functions(PRFs)[37, p. 79-159]. A pseudorandom function $F : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l$, where $\{0,1\}^l$ denotes an l-bit sequence, accepts two arguments as its inputs: a key κ and a seed γ and returns a random number ρ. PRFs cannot be differentiated from truly random functions by any efficient procedure that can get the values of the functions at arguments of its choice.

In order to generate addition $s_j^{\mathbf{a}}(k)$ and multiplication $s_j^{\mathbf{m}}(k)$ shares for $j \in \overline{\mathcal{N}}_i$ using PRFs, every agent $j \in \overline{\mathcal{N}}_i$ randomly selects $\kappa_{jh}^{\mathbf{m}}$ and $\kappa_{jh}^{\mathbf{a}}$ from (8) such that

$$\prod_{h\in\overline{\mathcal{N}}_i} F(\kappa_{jh}^{\mathbf{m}}, \gamma_k) \equiv 1 \mod \Omega \tag{11a}$$

$$\sum_{h\in\overline{\mathcal{N}}_i} F(\kappa_{jh}^{\mathbf{a}}, \gamma_k) \equiv 0 \mod \Omega, \tag{11b}$$

for some $\gamma_k \in \mathbb{Z}_\Omega$ to be specified later. Then, agent j sends $\kappa_{jh}^{\mathbf{a}}$ and $\kappa_{jh}^{\mathbf{m}}$ to $h \in \overline{\mathcal{N}}_i \backslash j$ through agent i.

After this step, agent $j \in \overline{\mathcal{N}}_i$ computes

$$F(\kappa_j^{\mathbf{m}}) := \prod_{h\in\overline{\mathcal{N}}_i} F(\kappa_{hj}^{\mathbf{m}}, \gamma_k) \bmod \Omega \tag{12a}$$

$$F(\kappa_j^{\mathbf{a}}) := \sum_{h\in\overline{\mathcal{N}}_i} F(\kappa_{hj}^{\mathbf{a}}, \gamma_k) \bmod \Omega \tag{12b}$$

to obtain the PRFs that are needed for generating her random shares. Agent $j \in \overline{\mathcal{N}}_i$ then is able to get $s_j^{\mathbf{a}}(k)$ and $s_j^{\mathbf{m}}(k)$ by evaluating $F(\kappa_j^{\mathbf{a}}, \gamma_k)$ and $F(\kappa_j^{\mathbf{m}}, \gamma_k)$ for a specific seed γ_k.

We remark that (11) and (12) *only need to be executed once*, and the agents do not need to communicate with each other after receiving the key κ. Moreover, we note that the (initial) seed is a public value, is the same for all agents $\overline{\mathcal{N}}_i$, and should be distinct for every time index $k \in [K]$. To ensure this, before the start of the protocol, the agents can agree on a public value, namely $\gamma_0 = S \in \mathbb{Z}_\Omega$, and then use $\gamma_k = F(., s(k))$ as the seed for all time.

4.3. The protocol

Now that the additive and multiplicative shares are generated, we provide an algorithm that enables the private computation of $\mathcal{P}_i$ in (4). To simplify the presentation and ease the notation, we discuss the required steps for the case that $T = 1$ (see (5)). We explain in Remark 1 how the proposed algorithm can be extended to the case $T > 1$.

The formal steps of the algorithm is provided in the next page (see Algorithm 1.) In the sequel, we drop t and the argument k in $x_i(k)$ and $x_j(k)$ to simplify the notation. To differentiate between a generic variable x_i (x_j, respectively) and its particular value at a given time index, we denote the latter by $\boldsymbol{x_i}$ ($\boldsymbol{x_j}$). Recall that the structure of the involved polynomial functions of (x_i, x_j) are known but the values $\boldsymbol{x_i}$ ($\boldsymbol{x_j}$) are considered private. Moreover, among the neighbors of agent i a specific agent denoted by $D_i \in \mathcal{N}_i$ is distinguished and her role becomes clear later (see Step 5 and Remark 2).

S1) At the start of the algorithm, agent i chooses independently her public key pk_i and private key sk_i for the Paillier scheme; then publishes her public key pk_i.

S2) Agent i uses her public key pk_i to encrypt her private quantities that appear in $P_{ij}(x_i, x_j)$ and $W_j(x_j)$, and sends the corresponding encrypted terms, namely $[\![c_{p_ip_j}\boldsymbol{x_i}^{p_i}]\!]_{\mathrm{pk}_i}$ to agent $j \in \mathcal{N}_i$ and $[\![c_{q_j}]\!]_{\mathrm{pk}_i}$ to agent $j \in (\mathcal{N}_i \setminus D_i)$ for all p_i, p_j and q_j where p_i, p_j and q_j are the exponents of the respective polynomial for the corresponding agent. The reason behind this encryption is elaborated in Remark 3. Moreover, she computes $\mu_i = \left(s_i^{\mathbf{m}} W_i(\boldsymbol{x_i})\right) \bmod \Omega$ and records it for Step 4.

S3) Every agent $j \in (\mathcal{N}_i \backslash D_i)$ encrypts $s_j^{\mathbf{a}}$ using pk_i and evaluates the following expressions over the ciphertext

$$\sigma_j = \prod_{p_i,p_j} ([\![c_{p_ip_j}\boldsymbol{x_i}^{p_i}]\!]_{\mathrm{pk}_i})^{\boldsymbol{x_j}^{p_j}} [\![s_j^{\mathbf{a}}]\!]_{\mathrm{pk}_i} \bmod N^2 \tag{13a}$$

$$[\![\mu_j]\!]_{\mathrm{pk}_i} = \prod_{q_j} ([\![c_{q_j}]\!]_{\mathrm{pk}_i})^{s_j^{\mathbf{m}}\boldsymbol{x_j}^{q_j}} \bmod N^2, \tag{13b}$$

then sends σ_j and $[\![\mu_j]\!]_{\mathrm{pk}_i}$ to agent i. By the end of this step, all computations from the agents $j \in (\mathcal{N}_i \setminus D_i)$ are carried out[2].

S4) Agent i decrypts $[\![\mu_j]\!]_{\mathrm{pk}_i}$ received in Step 3 using sk_i, computes the value

$$\Psi_i = \prod_{j\in(\overline{\mathcal{N}}_i)\backslash D_i} \mu_j \bmod \Omega, \tag{14}$$

and sends the encrypted values $[\![c_{q_j}\Psi_i]\!]_{\mathrm{pk}_i}$ with $j = D_i$ and for all q_j to agent $j = D_i$

S5) Agent $j = D_i$, using the values received in Step 4, computes

$$\Psi_j = \prod_{q_j} [\![c_{q_j}\Psi_i]\!]_{\mathrm{pk}_i}^{(s_j^{\mathbf{m}}\boldsymbol{x_j}^{q_j})} \bmod N^2 \tag{15}$$

and (13a), and then sends $(\sigma_j)(\Psi_j) \bmod N^2$ to agent i. The reason behind this step will be made clear in Remark 2.

[2] In case coefficients c_{q_j}'s are not privacy sensitive, then agent j computes $\mu_j = (\sum_{q_j} c_{q_j} x_j^{q_j}) s_j^{\mathbf{m}} \mod \Omega$ in (13b).

S6) Agent i decrypts the received values in (13a) and values in Step 5 using her secret key sk_i to obtain

$$P_{ij}(\boldsymbol{x_i}, \boldsymbol{x_j}) + s_j^{\mathbf{a}} \mod \Omega, \quad \forall\ j \in (\mathcal{N}_i \backslash D_i)$$

$$P_{ij}(\boldsymbol{x_i}, \boldsymbol{x_j}) + s_j^{\mathbf{a}} + \prod_{j \in \overline{\mathcal{N}}_i} s_j^{\mathbf{m}} W_j(\boldsymbol{x_j}) \mod \Omega, \quad j = D_i.$$

S7) Agent i sums the received results in Step 6 and includes her own share of addition $s_i^{\mathbf{a}}$ to obtain

$$\sum_{j \in \mathcal{N}_i} P_{ij}(\boldsymbol{x_i}, \boldsymbol{x_j}) + \prod_{j \in \overline{\mathcal{N}}_i} W_j(\boldsymbol{x_j}) \mod \Omega,$$

where we have used (7a) and (7b). After decoding, the above expression reduces to $\mathcal{P}_i(\cdot)$ in (4) as desired.

Algorithm 1: The protocol for private evaluation of polynomial (4) at time index k and $T = 1$

Input: $\{\{c_{p_i p_j}, c_{q_j}\}_{j \in \mathcal{N}_i}, \{x_j, s_j^{\mathbf{a}}, s_j^{\mathbf{m}}\}_{j \in \overline{\mathcal{N}}_i}\}$
Output: Evaluation of $\mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i})$ given in (4)

1 Agent i generates pk_i and sk_i and sends pk_i to agent $j \in \mathcal{N}_i$
2 **for** $j \in \mathcal{N}_i$ **do**
  agent i using pk_i sends $[\![c_{p_i p_j} \boldsymbol{x_i}^{p_i}]\!]_{\mathrm{pk}_i}$ to each agent $j \in \mathcal{N}_i$ and sends $[\![c_{q_j}]\!]_{\mathrm{pk}_i}$ to each $j \in \mathcal{N}_i \setminus D_i$
3 **for** $j \in (\mathcal{N}_i \backslash D_i)$ **do**
  agent j computes σ_j and $[\![\mu_j]\!]_{\mathrm{pk}_i}$ given in (13a) and (13b) and sends the result to agent i
4 Agent i computes Ψ_i given in (14), and sends $[\![c_{q_j} \Psi_i]\!]_{\mathrm{pk}_i}$ to agent $j = D_i$ for all q_j
5 Agent $j = D_i$ computes σ_j and Ψ_j given in (13a) and (15), then sends $(\sigma_j)(\Psi_j) \mod N^2$ to agent i
6 Agent i decrypts the received messages from her neighbors using sk_i
7 Agent i aggregates the results to obtain $\mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i}) \mod \Omega$
8 Agent i decodes the results to obtain $\mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i})$

A few remarks are in order concerning the proposed algorithm:

Remark 1 (Extension to $T > 1$). The proposed Algorithm 1 can be easily extended to the case $T > 1$. This requires the agents D_i and i to repeat their tasks (Steps 4 and 5 of the algorithm) for every multivariate term $Q_i^t = \prod_{j \in \overline{\mathcal{N}}_i} W_j^t(x_j)$; see (5). In this case, every agent $j \in \overline{\mathcal{N}}_i$ also needs T multiplicative shares $s_j^{\mathbf{m}}$ which can be generated by (10b) or (12b). It is worth mentioning that when $T = 0$, i.e. $\mathcal{P}_i(\cdot)$ has no multivariate term, the proposed algorithm needs neither multiplicative shares $s_j^{\mathbf{m}}$ nor the presence of a distinguished neighbor. Working with (4), rather than (3), allows us to capture this special case properly.

Remark 2 (The role of distinguished neighbor). We designed Steps 4 and 5 of the algorithm such that the value of $\sum_{j \in \mathcal{N}_i} P_{ij}(x_i, x_j)$ and that of $Q_i = \sum_{t=1}^{T} Q_i^t(x_i(k), x_{\mathcal{N}_i}(k))$ in (4) remain hidden from both agent i and D_i. In fact, agent i can only evaluate the summation of these two terms, which amounts to the interested query in (4). Putting it differently, we can remove the distinguished neighbor from the algorithm at the expense of revealing the values $\sum P_{ij}(x_i, x_j)$ and Q_i^t individually. This may not readily lead to a privacy breach for other agents, but it provides agent i with extra information (beyond the query itself) that can compromise the privacy of her neighbors. Therefore, the distinguished neighbor D_i should be chosen with the consensus of all neighbors of agent i, and without involvement of agent i in this decision. We again emphasize that the current algorithm is devised such that no information other than the query $\mathcal{P}_i(\cdot)$ will be made available to the agent i.

Remark 3 (Encryption). It should be noted again that coefficients $c_{p_i p_j}$, c_{q_j} and the variables x_i in $P_{ij} = \sum_{p_i, p_j} c_{p_i p_j} x_i^{p_i} x_j^{p_j}$ and $W_j = \sum_{q_j} c_{q_j} x_j^{q_j}$ are sensitive data and their encryption are justified. For this reason, agent i in Step 2 of the proposed algorithm sends encrypted quantities $[\![c_{p_i p_j} \boldsymbol{x_i}^{p_i}]\!]_{\mathrm{pk}_i}$ and $[\![c_{q_j}]\!]_{\mathrm{pk}_i}$ to her neighbors. If P_{ij} has $\mathfrak{n}$ terms involving x_j then agent i has to encrypt $\mathfrak{n}$ values and sends them to agent j for each $k \in [K]$, resulting in $\mathfrak{n} \times K$ encrypted values. Clearly, both communication and computation costs are increased drastically with the increase of $\mathfrak{n}$. A fully homomorphic encryption such as [38] can be employed to reduce the communication since agent i can encrypt $c_{p_i p_j}$ and $x_i(k)$ for all $k \in [K]$ and allow agent j to evaluate $P_{ij}(x_i, x_j)$ over the ciphertext; leading to $\mathfrak{n} + K + 1$ encrypted values for the whole time interval. However, this benefit comes at the expense of increased computational complexity for agent j due to the high computational load of fully homomorphic schemes.

Remark 4 (Beyond polynomial functions). We can privately evaluate a wider class of functions represented by

$$P_{ij}(x_i, x_j) = \sum_{p_i, p_j} c_{p_i p_j} f_i^{(p_i)}(x_i) f_j^{(p_j)}(x_j), \qquad W_j = \sum_{q_j} c_{(q_j)} g_j^{(q_j)}(x_j),$$

where $f_i^{(\cdot)} : \mathbb{R} \to \mathbb{R}$, and $g_j^{(\cdot)} : \mathbb{R} \to \mathbb{R}$. This can be achieved by treating $f_i^{(\cdot)}(x_i)$ as x_i, and $f_j^{(\cdot)}(x_j)$ and $g_j^{(\cdot)}(x_j)$ as x_j in Algorithm 1. This generalized class of functions essentially does not introduce extra communication and computation costs since all additional computations are performed over the plain text.

For a better illustration of the protocol, we provide a simple example.

Example 1. (cont.) Consider again the polynomial in (1):

$$\mathcal{P}_1 = \underbrace{2x_1^2 x_2}_{P_{12}} + \underbrace{3x_1 x_3}_{P_{13}} + \underbrace{4x_1 x_4^3}_{P_{14}} + \underbrace{x_1}_{W_1} \underbrace{x_2^2}_{W_2} \underbrace{(x_3^2 + 3x_3)}_{W_3} \underbrace{x_4}_{W_4}.$$

For the sake of simplicity, we assume $\boldsymbol{x_j} \in \mathbb{Z}_{\geq 0}$ for $j \in \{1, 2, 3, 4\}$, otherwise an encoding-decoding scheme is used. Let node 4 be the distinguished neighbor. Based on Algorithm 1, agents 1, 2, 3, and 4 generate multiplicative and additive shares $s_j^{\mathbf{m}}$, $s_j^{\mathbf{a}}$ for $j \in \{1, 2, 3, 4\}$ either through (10) or (12). Agent 1 generates pk_1 and sk_1 and publishes pk_1.

As for the bivariate parts, agent 1 sends $[\![2\boldsymbol{x_1}^2]\!]_{\mathrm{pk}_1}$ to agent 2, $[\![3\boldsymbol{x_1}]\!]_{\mathrm{pk}_1}$ to agent 3, and $[\![4\boldsymbol{x_1}]\!]_{\mathrm{pk}_1}$ to agent 4. Here, among the

multiplicative terms, only W_3 contains privacy sensitive coefficients; hence, agent 1 sends the encrypted values $[\![1]\!]_{\mathrm{pk}_1}$ and $[\![3]\!]_{\mathrm{pk}_1}$ to agent 3.

In the next step, agent 2 computes $\sigma_2 = ([\![2\boldsymbol{x_1^2}]\!]_{\mathrm{pk}_1})^{\boldsymbol{x_2}}[\![s_2^{\mathbf{a}}]\!]_{\mathrm{pk}_1} \mod N^2$ and $\mu_2 = s_2^{\mathbf{m}}\boldsymbol{x_2^2} \mod \Omega$ and sends the results to agent 1. Meanwhile, agent 3 computes the following quantities and sends them to agent 1:

$$\sigma_3 = ([\![3\boldsymbol{x_1}]\!]_{\mathrm{pk}_1})^{\boldsymbol{x_3}}[\![s_3^{\mathbf{a}}]\!]_{\mathrm{pk}_1} \mod N^2,$$

$$[\![\mu_3]\!]_{\mathrm{pk}_1} = ([\![1]\!]_{\mathrm{pk}_1})^{s_3^{\mathbf{m}}\boldsymbol{x_3^2}}([\![3]\!]_{\mathrm{pk}_1})^{s_3^{\mathbf{m}}\boldsymbol{x_3}} \mod N^2.$$

Next, agent 1 computes $\Psi_1 = \left(s_1^{\mathbf{m}}\boldsymbol{x_1}\right)\mu_2\mu_3 \mod \Omega$ and sends $[\![\Psi_1]\!]_{\mathrm{pk}_1}$ to agent 4.

The distinguished neighbor 4 computes $\sigma_4 = ([\![4\boldsymbol{x_1}]\!]_{\mathrm{pk}_1})^{\boldsymbol{x_4^3}}[\![s_4^{\mathbf{a}}]\!]_{\mathrm{pk}_1}$ and $\Psi_4 = ([\![\Psi_1]\!]_{\mathrm{pk}_1})^{s_4^{\mathbf{m}}\boldsymbol{x_4}} \mod N^2$, and sends back $\sigma_4\Psi_4 \mod N^2$ to agent 1.

Finally, agent 1 decrypts σ_2, σ_3, and $\sigma_4\Psi_4$ and aggregates them with $s_1^{\mathbf{a}}$ to obtain $\mathcal{P}_1$. ◊

4.4. Robustness against agent dropouts

The proposed scheme is essentially robust to dropout of an agent, say j, during the execution of the algorithm. This means that agent i is able to evaluate a new polynomial $\tilde{\mathcal{P}}_i(x_i, x_{(\mathcal{N}_i\setminus j)})$ that does not include x_j. Note that $\tilde{P}_i$ can be obtained from P_i by setting $p_j = 0$ in (3).

To endow Algorithm 1 with this capability, agent i notifies the neighboring agents $\mathcal{N}_i\setminus j$ that agent j is no longer a part of the computation. By doing so, every agent $h \in \overline{\mathcal{N}}_i\setminus j$ should merge (add or multiply) her own shares with the shares of the dropped out agent. Namely,

$$s_{hh}^{\mathbf{a}}(\tilde{k}) \equiv s_{hh}^{\mathbf{a}}(\tilde{k}) + s_{hj}^{\mathbf{a}}(\tilde{k}) \bmod \Omega, \quad s_{hh}^{\mathbf{m}}(\tilde{k}) \equiv s_{hh}^{\mathbf{m}}(\tilde{k})s_{hj}^{\mathbf{m}}(\tilde{k}) \bmod \Omega,$$

where $\tilde{k}$ denotes the time index marking the dropout of agent j. Every agent $h \in \overline{\mathcal{N}}_i\setminus j$ obtains $s_h^{\mathbf{a}}(\tilde{k})$ and $s_h^{\mathbf{m}}(\tilde{k})$ from (10) by using the updated shares $s_{hh}^{\mathbf{a}}$ and $s_{hh}^{\mathbf{m}}$, and discarding the shares $s_{jh}^{\mathbf{a}}(\tilde{k})$ and $s_{jh}^{\mathbf{m}}(\tilde{k})$ which she previously generated for agent j. Clearly, the newly obtained quantities $s_h^{\mathbf{a}}(\tilde{k})$ and $s_h^{\mathbf{m}}(\tilde{k})$ satisfy (7), and can serve as the input of the algorithm from the time index $k = \tilde{k}$ onward.

5. Privacy analysis

In this section, we focus on privacy preserving properties of the proposed algorithm. To study such properties, we partition $\mathcal{V}$ into a set of corrupt $\mathcal{V}_c$ and noncorrupt agents $\mathcal{V}_{nc}$, where the corrupt agents may collude with each other and the noncorrupt agents are simply honest-but-curious. We first discuss the privacy guarantees of Algorithm 1 in the absence and presence of colluding agents. Then, we shift our focus to a network-level analysis with multiple queries.

5.1. Local privacy analysis

First, we formally prove the privacy of Algorithm 1 in the case of no collusion. This shows that no privacy sensitive information is leaked throughout the communications dictated by the algorithm.

Proposition 1. *Let $\mathcal{N}_i \cap \mathcal{V}_c = \varnothing$ and $|\mathcal{N}_i| > 1$. Then Algorithm 1 computes $\mathcal{P}_i(\cdot)$ accurately and preserve privacy of $\mathsf{Pv}_j = \{x_j\}$ for $j \in \mathcal{N}_i$ against agent i. Moreover, Algorithm 1 preserves privacy of $\mathsf{Pv}_i = \{x_i, c_{p_ip_j}, c_{q_j}^{(t)}\}$ against the set $\mathcal{N}_i$.*

Proof. The proof uses real and ideal world paradigm to show the correctness and privacy of the algorithm. Correctness of the algorithm follows from Assumption 1 and privacy follows from the security of Paillier and secret sharing schemes. See Appendix A for a formal proof. □

Privacy of the neighbors of i is susceptible to the collusion of agent i with other neighbors. The reason for the latter is that, unlike agent i that uses encryption, other agents rely on a secret sharing scheme. Hence, we formalize next the privacy guarantees when collusion occurs with agent i.

Theorem 1. *Let $i \in \mathcal{V}_c$ and assume that $D_i \in \mathcal{V}_{nc}$. Then Algorithm 1 computes $\mathcal{P}_i(\cdot)$ accurately and protect privacy of Pv_j for $j \in \mathcal{N}_i \cap \mathcal{V}_{nc}$, if*

$$|\mathcal{N}_i \cap \mathcal{V}_{nc}| > 1.$$

Proof. The proof is built on Proposition 1 and uses real and ideal world paradigm to show the correctness and privacy of the algorithm. See Appendix A. □

By Theorem 1, privacy of the neighbors of i is fully preserved as long as agent i has at least two noncorrupt agents and the distinguished agent does not collude with agent i. We note again that if the distinguished neighbor colludes with agent i, the subqueries $\sum_{j\in\mathcal{N}_i} P_{ij}(x_i, x_j)$ and Q_i in (4) can still be privately and accurately computed (see also Remark 2).

Remark 5. In the context of (average) consensus the state of the art definition for privacy is that an adversary cannot estimate the value of x_j with any accuracy (see for example the definition of privacy in [20]). At the first glance, it seems that privacy guarantees in Theorem 1 is not stringent enough compared to this definition. However, we argue that in the consensus type problems, the proposed method guarantees the same level of privacy that exists in the literature. The reason is that in the case of consensus protocols, the function $\mathcal{P}_i(\cdot)$ becomes affine, i.e, $W_j^t(\cdot) = 0$ for all $j \in \mathcal{V}$. Hence, as long as i has at least one noncorrupt neighbor $h \neq j$, an attempt of agent i to infer x_j would at best lead to a linear equation of the form $x_j + x_h = b$. It is then clear that agent i cannot estimate the value of x_j with any accuracy, i.e., x_j can belong to $(-\infty, \infty)$. On the contrary, in the case of polynomial functions, the mere knowledge of the target function $\mathcal{P}_i(\cdot)$ may provide agent i an idea about x_j; an ellipsoid being a simple example. Finally, we recall that the distinguished neighbors become redundant in the case of affine functions as they only contribute to the computation of the multivariate polynomials in (4).

5.2. Network privacy analysis

So far we have examined privacy concerns that may result from the computation of $\mathcal{P}_i(\cdot)$, for some $i \in \mathcal{V}$, following Algorithm 1. Recall that in an interconnected network each agent aims to compute a function of her neighbors. Analogous to Theorem 1, we can show that the execution of Algorithm 1 by every agent $i \in \mathcal{V}$ protects privacy of Pv_j for $j \in \mathcal{V}_{nc}$. However, depending on the class of functions to be computed, colluding agents $\mathcal{V}_c$ may be able to infer privacy sensitive variables of noncoluding agents by putting together the results of their queries and carrying out a posterior analysis. Note that such potential privacy breach is oblivious to the employed privacy-preserving algorithm and descends directly from the problem setup, namely that each agent is computing a function $\mathcal{P}_i(\cdot)$. The interest in studying such privacy considerations is to first highlight the inevitable limits in the privacy guarantees, and second to provide the designer of the control/optimization algorithm with valuable privacy related insights.

The first observation is that if the number of noncolluding agents is greater than the number of colluding ones, namely

$$|\mathcal{V}_{nc}| > |\mathcal{V}_c|\,,$$

then the colluding agents cannot uniquely infer the vector $\{x_j\}_{j\in\mathcal{V}_{nc}}$. However, the above guarantee is weak in that it does not ensure privacy of a specific noncorrupt agent. Next, we investigate more closely the conditions under which privacy of a single agent is guaranteed against the collective information obtained by colluding agents across the entire network.

Let $|\mathcal{V}_c| = n$, $|\mathcal{V}_{nc}| = m$. Observe that collusion of n corrupt agents results in a set of polynomial equations:

$$\Phi(x_c, x_{nc}) = b, \tag{16}$$

where $x_{nc} = \{x_i\}_{i\in\mathcal{V}_{nc}}$, $x_c = \{x_i\}_{i\in\mathcal{V}_c}$, $b \in \mathbb{R}^n$, and $\Phi : \mathbb{R}^{(n+m)} \to \mathbb{R}^n$. Here, x_{nc} is the indeterminate set, whereas b, and x_c, and the polynomial functions in Φ are known to the colluding agents.

For technical reasons and in order to write the results more explicitly, we assume that for each $i \in \mathcal{V}_c$, at most one variable from the set $\{x_j : j \in \mathcal{V}_{nc} \setminus i\}$ contributes to the product of W_j^t's in (4).

Moreover, without loss of generality assume that the first m agents are noncorrupt. Consequently, (16) reduces to

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} P(x_1) \\ P(x_2) \\ \vdots \\ P(x_m) \end{bmatrix} = b, \tag{17}$$

where the nonlinear map $P : \mathbb{R} \to \mathbb{R}^r$ is given by $P(\alpha) := \begin{bmatrix} \alpha & \alpha^2 & \cdots & \alpha^r \end{bmatrix}^\top$, $\forall \alpha \in \mathbb{R}$, and $a_{ij} \in \mathbb{R}^{1\times r}$. Here r is the maximum degree of the polynomials in (16), in terms of the indeterminate variables x_{nc}.

It is illustrative to first look at the special case of affine functions, where $r = 1$. Then, solutions of (17) are completely characterized by

$$x_{nc} = x_{nc}^* + (I_m - A^+A)v, \quad v \in \mathbb{R}^m.$$

where $A = [a_{ij}]$, A^+ denotes the Moore-Penrose inverse of A, and x^* is the vector containing the true values of $\{x_i\}_{i\in\mathcal{V}_{nc}}$. Consequently, the value of x_i with $i \in \mathcal{V}_{nc}$ is uniquely identified if and only if

$$e_i^\top \Pi = 0, \tag{18}$$

where $\Pi := I_m - A^+A$ and e_i is the ith unit vector of the standard basis in $\mathbb{R}^m$. Indeed if (18) holds, then $x_i = x_i^*$. Conversely, if (18) does not hold, then x_i has at least two distinct solutions x_i^* and $x_i^* + \left\|e_i^T\Pi\right\|^2$, where the latter is obtained by setting $v = e_i$ and noting that $\Pi^2 = \Pi$. The situation for $r \geq 1$ becomes more complex and gives rise to the following result:

Theorem 2. *The private variables $\{x_i\}_{i\in\mathcal{V}_{nc}}$ are uniquely identified from* (17) *if and only if*

$$(\{P(x_i^*)\} + \mathrm{im}(e_i^\top \otimes I_r)\Pi) \cap \mathrm{im}\, P = \{P(x_i^*)\}, \tag{19}$$

where "$\otimes$" denotes the Kronecker product and $\mathrm{im}\, P = \{y \in \mathbb{R}^r : \exists\alpha, y = P(\alpha)\}$.

Note that in case $r = 1$, we have $\mathrm{im}\, P = \mathbb{R}$ and the conditions reduces to $\mathrm{im}(e_i^\top\Pi)$ being zero, which is equivalent to (18).

Proof of Theorem 2: Let $y_i = P(x_i)$, and $y = \mathrm{col}(y_i)$, $i \in \mathcal{V}_{nc}$. Then, we can equivalently rewrite (17) as

$$Ay = b, \tag{20a}$$

$$y_i \in \mathrm{im}(P), \forall i. \tag{20b}$$

Clearly, any solution to (17) satisfies (20). Conversely, any solution to (20) can be mapped back to a solution of (17). Now, all solutions to (20a) are given by

$$y = y^* + (I_{mr} - A^+A)v, \quad v \in \mathbb{R}^{mr},$$

where $y^* = \mathrm{col}(y_i^*)$ with $y_i^* := P(x_i^*)$. Looking at the ith block row, we find that

$$y_i = P(x_i^*) + (e_i^\top \otimes I_r)\Pi v, \quad v \in \mathbb{R}^{mr},$$

where $\Pi = I_{mr} - A^+A$. Consequently, any solution to (20) satisfies

$$y_i \in (\{P(x_i^*)\} + \mathrm{im}(e_i^\top \otimes I_r)\Pi) \cap \mathrm{im}\, P.$$

Moreover, any y_i satisfying the above inclusion is a solution to (20). We conclude that $P(x_i)$, and thus x_i, is uniquely identifiable if and only if (19) holds. □

6. Case study

We demonstrate privacy and performance of the proposed algorithm in a networked system by considering a noncooperative game as described in subsection (3.1) with $|\mathcal{V}| = 30$. Each player aims to minimize a cost function given by

$$J_i(x_i, x_{-i}) = a_i x_i^2 + x_i\Big(\sum_{j\in\mathcal{N}_i} c_{ij,01}x_j\Big) + \prod_{j\in\overline{\mathcal{N}}_i}(c_{j,1}x_j + c_{j,2}x_j^2),$$

where x_i takes value from a local admissible set $\Gamma_i = [0, 2]$. The actions need to satisfy a global affine constraint $\sum_{j\in\mathcal{V}} x_j \geq 1$.

Moreover, we assume that the players adopt the scheme in [31] for reaching GNE (see (2)) with $\tau_i = \tau$. The dynamics of player i is then given by

$$x_i(k+1) = \text{proj}_{\Gamma_i}(x_i(k) - \tau(2a_i x_i(k) + \sum_{j\in\mathcal{N}_i} P_{ij} + \prod_{j\in\overline{\mathcal{N}}_i} W_j - \lambda_i)), \quad (21)$$

where $P_{ij} = c_{ij,01}x_j$, $W_i = c_{i,0} + 2c_{i,1}x_i$ and $W_j = c_{j,1}x_j + c_{j,2}x_j^2$ are the terms specified in (4) and a_i, c_{ij}, and c_j are player i's private cost function parameters, randomly picked from Γ_i for the simulation purposes.

The aim here is to privately evaluate (21) using Algorithm 1. To this end, we set $\tau = 0.01$, and choose the length of Paillier's key N and Ω in (8) equal to 1024 and 200 bits, respectively. We assume that player i has 3 neighbors, and thus her cost function depends explicitly on decisions of those neighboring players. The computations are performed[3] using a 2.1 GHz Intel Core i5 processor drawing on modules from Python library [39]. Moreover, we have evaluated player i's decision trajectory using plain signals, i.e. without any privacy concerns.

As it can be seen from Fig. 1 the trajectory of player i asymptotically converges to the origin using the proposed algorithm similar to the case where a public algorithm is used. This implies that the proposed algorithm introduces no systematic error in the computation, thereby certifying the correctness of the scheme (see also Theorem 1). In order to investigate the computation and communication load of the proposed protocol, we change two parameters in the algorithm: 1) the length of the Paillier's key N and 2) number of neighbors of the player. The length of N(in bits) plays an important role in the security of the Paillier cryptosystem; generally the greater the length of N is the more secure the Paillier scheme becomes. As for the change in the number of neighbors, we execute the algorithm for the case $|\mathcal{N}_i| = 9$ and $|\mathcal{N}_i| = 27$. The results of the aforementioned changes on the computation time per time-step of the algorithm are illustrated in Fig. 2.

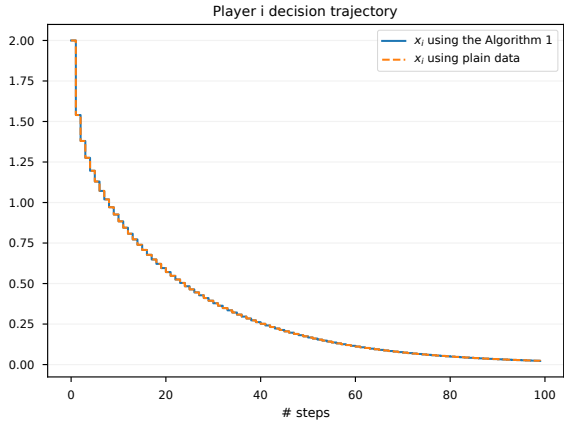

Figure 1: Trajectory of player i decision variable using Algorithm 1 and plain data

[3] https://github.com/teimour-halizadeh/polynomial-evaluation

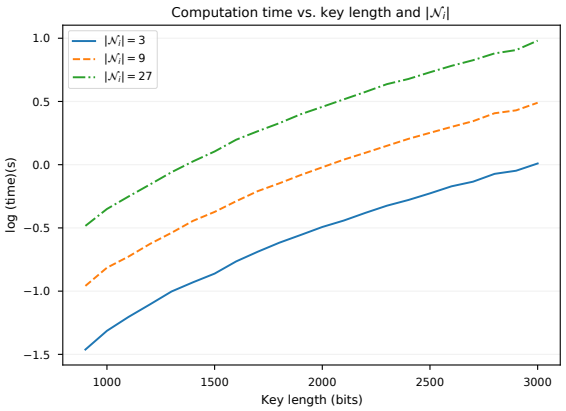

Figure 2: Required computation time for the proposed algorithm with respect to the key length of Paillier scheme and number of neighbors of an agent

As it is clear from this figure, the computation time increases linearly with respect to the number of neighbors, $O(|\mathcal{N}_i|)$ and cubically with respect to key length, $O((\text{key length})^3)$. Communication load is proportional to the size of the generated ciphertext which itself changes linearly in terms of both $|\mathcal{N}_i|$ and bit length of N. It should be mentioned we have not employed any techniques to optimize the computation time.

7. Conclusion

In this study, we have presented a fully distributed algorithm for privacy preserving evaluation of a general polynomial over a network of agents. The algorithm is based on a suitable representation of polynomials for network systems, and adopts PHE technique and multiplicative-additive secret sharing from cryptographic tools. Furthermore, we have provided sufficient privacy-preserving conditions both at the agent and the network level. As observed, the proposed algorithm is robust against dropout of agents, lightweight in communication and is extendable to a class of nonlinear schemes. The numerical investigations verifies that the algorithm can be used to protect privacy in a network subject to additional communication and computation costs. Extension to more general nonlinear functions and considering possible active adversaries are among directions for future research.

References

[1] P. Van Aubel and E. Poll, "Smart metering in the netherlands: What, how, and why," *International Journal of Electrical Power & Energy Systems*, vol. 109, pp. 719–725, 2019.

[2] J. Le Ny and G. J. Pappas, "Differentially private filtering," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 341–354, 2013.

[3] Y. Mo and R. M. Murray, "Privacy preserving average consensus," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 753–765, 2016.

[4] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design," *Automatica*, vol. 81, pp. 221–231, 2017.

[5] ——, "Differentially private distributed convex optimization via functional perturbation," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 395–408, 2016.

[6] Y. Kawano, K. Kashima, and M. Cao, "Modular control under privacy protection: Fundamental trade-offs," *Automatica*, vol. 127, p. 109518, 2021.
[7] Y. Kawano and M. Cao, "Design of privacy-preserving dynamic controllers," *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3863–3878, 2020.
[8] I. Mironov, "On significance of the least significant bits for differential privacy," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 2012, p. 650–661.
[9] C. Altafini, "A system-theoretic framework for privacy preservation in continuous-time multiagent dynamics," *Automatica*, vol. 122, p. 109253, 2020.
[10] A. Sultangazin and P. Tabuada, "Symmetries and isomorphisms for privacy in control over the cloud," *IEEE Transactions on Automatic Control*, vol. 66, no. 2, pp. 538–549, 2020.
[11] N. Monshizadeh and P. Tabuada, "Plausible deniability as a notion of privacy," in *IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1710–1715.
[12] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford university, 2009.
[13] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *IEEE 54th Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 6836–6843.
[14] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Engineering Practice*, vol. 67, pp. 13–20, 2017.
[15] M. Schulze Darup, "Encrypted polynomial control based on tailored two-party computation," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 11, pp. 4168–4187, 2020.
[16] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016.
[17] J. H. Cheon, K. Han, H. Kim, J. Kim, and H. Shim, "Need for controllers having integer coefficients in homomorphically encrypted dynamic system," in *IEEE 57th Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 5020–5025.
[18] C. Murguia, F. Farokhi, and I. Shames, "Secure and Private Implementation of Dynamic Controllers Using Semihomomorphic Encryption," *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3950–3957, 2020.
[19] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshia, P. Tabuada, and G. J. Pappas, "Cloud-based quadratic optimization with partially homomorphic encryption," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 2357–2364, 2021.
[20] M. Ruan, H. Gao, and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4035–4049, 2019.
[21] W. Fang, M. Zamani, and Z. Chen, "Secure and privacy preserving consensus for second-order systems based on paillier encryption," *Systems & Control Letters*, vol. 148, p. 104869, 2021.
[22] C. N. Hadjicostis and A. D. Dominguez-Garcia, "Privacy-Preserving Distributed Averaging via Homomorphically Encrypted Ratio Consensus," *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3887–3894, 2020.
[23] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. 314–325, 2018.
[24] M. Schulze Darup, A. Redder, and D. E. Quevedo, "Encrypted cooperative control based on structured feedback," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 37–42, 2019.
[25] A. B. Alexandru, M. Schulze Darup, and G. J. Pappas, "Encrypted cooperative control revisited," in *IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 7196–7202.
[26] A. B. Alexandru and G. J. Pappas, "Private weighted sum aggregation," *IEEE Transactions on Control of Network Systems*, 2021.
[27] M. Schulze Darup, A. B. Alexandru, D. E. Quevedo, and G. J. Pappas, "Encrypted control for networked systems: An illustrative introduction and current challenges," *IEEE Control Systems Magazine*, vol. 41, no. 3, pp. 58–78, 2021.
[28] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.
[29] T. Hosseinalizadeh, F. Turkmen, and N. Monshizadeh, "Private computation of polynomials over networks," in *IEEE 60th Conference on Decision and Control (CDC) to be presented*. IEEE, 2021.
[30] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2015.
[31] P. Yi and L. Pavel, "An operator splitting approach for distributed generalized Nash equilibria computation," *Automatica*, vol. 102, pp. 111–121, 2019.
[32] S. Gade, A. Winnicki, and S. Bose, "On privatizing equilibrium computation in aggregate games over networks," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3272–3277, 2020.
[33] M. Shakarami, C. De Persis, and N. Monshizadeh, "Privacy and robustness guarantees in distributed dynamics for aggregative games," *arXiv preprint arXiv:1910.13928*, 2019.
[34] V. K. Dzyadyk and I. A. Shevchuk, *Theory of uniform approximation of functions by polynomials*. de Gruyter, 2008.
[35] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.
[36] G. H. Hardy, E. M. Wright *et al.*, *An introduction to the theory of numbers*. Oxford university press, 1979.
[37] Y. Lindell, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Springer, 2017.
[38] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.
[39] C. Data61, "Python paillier library," *GitHub Repository*, 2013. [Online]. Available: https://github.com/data61/python-paillier

Appendix A.

To provide a formal proof, we present the definitions of view and simulator in a protocol.

Definition 1 (View)**.** [37, p. 283] Let $f(x,y) = (f_1(x,y), f_2(x,y))$ be a function, and let π be a two party protocol(or algorithm) for computing f. The view of party i ($i \in \{1,2\}$) during an execution of π on (x,y) and security parameter n is denoted by $\mathsf{View}_i^\pi(x,y,n)$ and equals $(w, r^i; m_1^i, \ldots, m_t^i)$ where $w \in \{x,y\}$, r^i is the random number used by party i, and m_j^i, represents the j-th message that she received.

Definition 2 (Simulator)**.** [37, p. 278] Let $f(x,y) = (f_1(x,y), f_2(x,y))$ be a function, and let π be a two party protocol for computing f. A simulator for party i ($i \in \{1,2\}$) Sim_i^π is a probabilistic polynomial-time algorithm which given the input and output of i , $(w, f_i(x,y))$ where $w \in \{x,y\}$ can result an output whose distribution is exactly the same as $\mathsf{View}_i^\pi(x,y,n)$.

Proof of Proposition 1: To prove this proposition, we use the simulation based paradigm also known as real/ideal world [37, Chap. 6]. For the deterministic function (4), the security of the proposed algorithm can be shown by verifying its 1) correctness and 2) privacy. The proposed algorithm is correct since the agents are honest-but-curious and hence the correct value of $\mathcal{P}_i(\cdot)$ is obtained by following the Protocol 1. To prove privacy of Pv_j for $j \in \mathcal{N}_i$ against agent i, we need to establish the existence of a simulator Sim_i^π for i. The input of agent i, meaning the information set she commits to the protocol is $\{\{c_{p_i p_j}, c_{q_j}\}_{j\in\mathcal{N}_i}, x_i, s_i^{\mathbf{a}}, s_i^{\mathbf{m}}, \mathrm{pk}_i, \mathrm{sk}_i\} := \mathcal{I}_i$ and the input of all agents involved in Algorithm 1

is $\{\{c_{p_ip_j}, c_{q_j}\}_{j\in\mathcal{N}_i}, \{x_j, s_j^{\mathbf{a}}, s_j^{\mathbf{m}}\}_{j\in\overline{\mathcal{N}}_i}, \mathrm{pk}_i, \mathrm{sk}_i\} := \mathcal{I}$. The View of agent i participating in Algorithm 1 given the set $\mathcal{I}$ is $\mathsf{View}_i^{\pi}(\mathcal{I}) = \{\mathcal{I}_i, \{\sigma_j, \mu_j\}_{j\in(\mathcal{N}_i\setminus D_i)}, \sigma_{D_i} + \Psi_{D_i}\}$, where σ_j, μ_j and $\sigma_{D_i} + \Psi_{D_i}$ are values received by agent i in Steps $3-5$ of the proposed algorithm. Given $\mathcal{I}_i$ and the output of the algorithm $\mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i})$ the simulator output is $\mathsf{Sim}_i^{\pi}(\mathcal{I}_i, \mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i})) = \{\mathcal{I}_i, \{\hat{\sigma}_j, \hat{\mu}_j\}_{j\in(\mathcal{N}_i\setminus D_i)}, \hat{\sigma}_{D_i} + \hat{\Psi}_{D_i}\}$. We claim that $\mathsf{View}_i^{\pi}(\mathcal{I}) \overset{c}{\equiv} \mathsf{Sim}_i^{\pi}(\mathcal{I}_i, \mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i}))$, that is they are computationally indistinguishable. This is true since Sim_i^{π} can pick the values $\{\{\hat{\sigma}_j, \hat{\mu}_j\}_{j\in(\mathcal{N}_i\setminus D_i)}, \hat{\sigma}_{D_i} + \hat{\Psi}_{D_i}\}$ uniformly randomly from (8) with the condition that they satisfy the output of the protocol, $\mathcal{P}_i(\boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}_i})$. The Sim_i^{π} can do so since $|\mathcal{N}_i| \geq 2$ and hence there exists at least two additive shares $s_j^{\mathbf{a}}$ and $s_h^{\mathbf{a}}$ (where $h \in \mathcal{N}_i\setminus j$), and two multiplicative shares $s_j^{\mathbf{m}}$ and $s_h^{\mathbf{m}}$ to enable it to calculate $\hat{\mu}_j$ and $\hat{\sigma}_j$ and $\hat{\sigma}_{D_i} + \hat{\Psi}_{D_i}$ with the same distribution as μ_j, σ_j and $\sigma_{D_i} + \Psi_{D_i}$. Therefore, the privacy of Pv_j for $j \in \mathcal{N}_i$ is preserved by Algorithm 1. Moreover, agent $j \in \mathcal{N}_i$ only receives as a private value $m_1^i = [\![c_{p_ip_j}\boldsymbol{x_i}^{p_i}]\!]_{\mathrm{pk}_i}$, $m_2^i = [\![c_{q_j}]\!]_{\mathrm{pk}_i} (j \neq D_i)$ and $m_3^i = [\![c_{q_j}\Psi_i]\!]_{\mathrm{pk}_i} (j = D_i)$ from agent i(Step 2 and 4 of Algorithm 1) which are encrypted values by Paillier's scheme. Since this scheme is semantically secure and agent j does not have the secret key sk_i, agent j's view is computationally indistinguishable from random numbers $\hat{m}_1^i, \hat{m}_2^i, \hat{m}_3^i \in \mathbb{Z}_{N^2}^*$. Therefore, the privacy of Pv_i is preserved by Algorithm 1. □

Proof of Theorem 1: Correctness of Algorithm 1 is similarly proved as of Proposition 1. Given the agent i, we need to prove the privacy of Pv_j for $j \in (\mathcal{N}_i \cap \mathcal{V}_{nc}) := \mathcal{V}_{nc}^i$ against $(\overline{\mathcal{N}}_i \cap \mathcal{V}_c) := \mathcal{V}_c^i$ and for that we need to establish the existence of a simulator $\mathsf{Sim}_{\mathcal{V}_c^i}^{\pi}$. We consider the worst case scenario, i.e. $|\mathcal{V}_{nc}^i| = 2$, meaning there are only 2 noncorrupt agents among the neighbors of agent i. Suppose that $\mathcal{V}_{nc}^i = \{h, D_i\}$ where $h \neq D_i$. The input of colluding agents $\mathcal{V}_c^i$ is $\{\{c_{p_ip_j}, c_{q_j}\}_{j\in\mathcal{N}_i}, \{x_j, s_j^{\mathbf{a}}, s_j^{\mathbf{m}}\}_{j\in\mathcal{V}_c^i}, \mathrm{pk}_i, \mathrm{sk}_i\} := \mathcal{I}_{\mathcal{V}_c^i}$ and the input of parties involved in Algorithm 1 is $\{\{c_{p_ip_j}, c_{q_j}\}_{j\in\mathcal{N}_i}, \{x_j, s_j^{\mathbf{a}}, s_j^{\mathbf{m}}\}_{j\in\overline{\mathcal{N}}_i}, \mathrm{pk}_i, \mathrm{sk}_i\} := \mathcal{I}$. The View of $\mathcal{V}_c^i$ participating in the proposed algorithm given the set $\mathcal{I}$ is $\mathsf{View}_{\mathcal{V}_c^i}^{\pi}(\mathcal{I}) = \{\mathcal{I}_{\mathcal{V}_c^i}, \sigma_h, \mu_h, \sigma_{D_i} + \Psi_{D_i}\}$ where σ_h, μ_h and $\sigma_{D_i} + \Psi_{D_i}$ are values received by the set $\mathcal{V}_c^i$ in Steps $3-5$ of the proposed algorithm. The simulator output is $\mathsf{Sim}_{\mathcal{V}_c^i}^{\pi}(\mathcal{I}_{\mathcal{V}_c^i}, \mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i})) = \{\mathcal{I}_{\mathcal{V}_c^i}, \hat{\sigma}_h, \hat{\mu}_h, \hat{\sigma}_{D_i} + \hat{\Psi}_{D_i}\}$, given $\mathcal{I}_{\mathcal{V}_c^i}$ and the output of the algorithm. The claim is $\mathsf{View}_{\mathcal{V}_c^i}^{\pi}(\mathcal{I}) \overset{c}{\equiv} \mathsf{Sim}_{\mathcal{V}_c^i}^{\pi}(\mathcal{I}_{\mathcal{V}_c^i}, \mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x}_{\mathcal{N}_i}))$, they are computationally indistinguishable. To see this, the simulator uses $\mathcal{I}_{\mathcal{V}_c^i}$ and $\mathcal{P}_i(\boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}_i})$ to have the evaluation of $\mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x_h}, \boldsymbol{x}_{\boldsymbol{D_i}}) = P_{ih}(\boldsymbol{x_i}, \boldsymbol{x_h}) + P_{iD_i}(\boldsymbol{x_i}, \boldsymbol{x}_{\boldsymbol{D_i}}) + \xi W_h(\boldsymbol{x_h}) W_{D_i}(\boldsymbol{x}_{\boldsymbol{D_i}})$, where $\xi := \prod_{j\in\mathcal{V}_c^i} W_j(\boldsymbol{x_j})$ is also known to the simulator. Then, the $\mathsf{Sim}_{\mathcal{V}_c^i}^{\pi}$ picks $\hat{s}_h^{\mathbf{a}}$, $\hat{s}_{D_i}^{\mathbf{a}}$, $\hat{s}_h^{\mathbf{m}}$, and $\hat{s}_{D_i}^{\mathbf{m}}$ randomly from (8) such that (7a) and (7b) hold. Next, it selects randomly $\hat{\boldsymbol{x}}_{\boldsymbol{h}}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{D_i}}$ from (8) such that $\mathcal{P}_i(\boldsymbol{x_i}, \boldsymbol{x_h}, \boldsymbol{x}_{\boldsymbol{D_i}})$ holds. Finally, the simulator outputs $\hat{\sigma}_h = P_{ih}(\boldsymbol{x_i}, \hat{\boldsymbol{x}}_{\boldsymbol{h}}) + \hat{s}_h^{\mathbf{a}}$ and $\hat{\mu}_h = \hat{s}_h^{\mathbf{m}} W_h(\hat{\boldsymbol{x}}_{\boldsymbol{h}})$ for agent h, and $\hat{\sigma}_{D_i} + \hat{\Psi}_{D_i} = P_{iD_i}(\boldsymbol{x_i}, \hat{\boldsymbol{x}}_{\boldsymbol{D_i}}) + \xi(\hat{\mu}_h)(\hat{s}_{D_i}^{\mathbf{m}} W_h(\hat{\boldsymbol{x}}_{\boldsymbol{D_i}})) + \hat{s}_{D_i}^{\mathbf{a}}$ for agent D_i. The set $\mathcal{V}_c$ cannot differentiate between $\boldsymbol{x_h}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{h}}$ for agent h, and $\boldsymbol{x}_{\boldsymbol{D_i}}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{D_i}}$ for agent D_i since σ_h, μ_h, $\sigma_{D_i} + \Psi_{D_i}$ have the same distribution as $\hat{\sigma}_h$, $\hat{\mu}_h$, $\hat{\sigma}_{D_i} + \hat{\Psi}_{D_i}$. Therefore, the privacy of Pv_j for $j \in \mathcal{V}_{nc}^i$ against $\mathcal{V}_c^i$ is preserved. □