2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Encrypted Cooperative Control Revisited

Andreea B. Alexandru[1] Moritz Schulze Darup[2] George J. Pappas[1]

Abstract— **Distributed systems are ubiquitous in present-day technologies like smart cities. Such applications require decentralized control, which reduces the load on a single central party, but requires communication and data sharing between the participating agents. However, agents might not trust their peers with their private data. We propose secure multi-party computation schemes that ensure the private computation of the control updates of each agent, without leaking any other information about the states and controls of their neighbors. To this end, we make use of homomorphic encryption and private sum aggregation schemes. We analyze the conditions such that a dishonest agent cannot observe the rest of the network. Finally, we present implementations of the proposed schemes and showcase their efficiency.**

I. Introduction

The recent drive towards increasing interconnectivity of systems has determined more and more systems to be operated using distributed control schemes. Examples such as smart grids, water-supply systems, robot swarms, or intelligent transportation systems benefit from this distributed computing framework. Applying distributed but cooperative controllers requires communication between the various subsystems or agents. In the resulting networked control system, sensible data is transmitted via possibly public networks and processed at neighboring agents, which can pose a privacy threat. Recent examples of data leakage and abuse in (critical) infrastructures, such as disturbing the normal functionality of power plants or inferring people's presence at home from smart meter measurements, have drawn attention to the risks of sharing data in the clear. The challenge thereby is to solve the conflict of ensuring individual privacy while simultaneously allowing for cooperation.

Secure communication between agents can be achieved using standard (symmetric) encryption schemes, such as AES or TLS. However, if the agents' individual data should not be available to their peers, to avoid the previously mentioned exploitations, more sophisticated cryptosystems are required. We will focus on solutions that employ homomorphic encryption schemes, which enable carrying out elementary mathematical operations on encrypted data [1]–[3].

A. Related work

Homomorphic encryption forms the basis for many recent cloud-based encrypted control schemes such as [4]–[9]. However, encrypted distributed control calls for different techniques than encrypted cloud-based control. For instance, different keys are required for different agents. Recent works on encrypted consensus [10]–[12], on encrypted distributed optimization [13]–[16] and in smart grid encrypted data aggregation [17]–[19] give insights about homomorphically encrypted distributed computation, but treat different aspects than the ones we are interested in for this paper.

It has recently been shown in [20] that encrypted cooperative control is capable of solving the privacy conflict using homomorphic encryption and allows secure interaction between the participating agents. However, while [20] demonstrates that private cooperative control is realizable, their proposed encrypted control scheme reveals more information about some participants' private local data than required to evaluate the local control laws. This results in a privacy leak that is difficult to address with existing approaches.

B. Our contribution

This paper designs a scheme that we call *private control update aggregation* (pCUA), which ensures that an agent learns nothing apart from the unconcealable information, i.e., the sum of contributions from its neighbors. As a corollary, we close the identified privacy leak of the scheme in [20]. More specifically, the values of the states, control actions and control gains of each agent are hidden from the rest of the participants. To this end, we design a *private weighted sum aggregation* (pWSA) scheme with secret weights. The solution relies on secret sharing and additively homomorphic encryption. This scheme involves an offline centralized step, to which we also propose an online fully distributed alternative. We implement the proposed solution and showcase its efficiency. Finally, we analyze the observability of the system from the perspective of malicious agents.

Organization. The private cooperative control problem statement and goal are described in Section II. In Section III, we describe a pWSA scheme where neither the aggregator nor the agents know the weights, yet the aggregator is capable of obtaining the weighted sum of the contributions of the agents. Section IV shows how to obtain the secrets needed for this scheme in a distributed way. In Section V, we show how to obtain the pCUA scheme from the pWSA scheme. Then, in Section VI, we compare the unaggregated case and the aggregated case in terms of observability. Finally, in Section VII, we discuss the implementation of the solution and the numerical results and conclude in Section VIII.

C. Notation

We use bold-face lower case for vectors, e.g. $\mathbf{x}$, and bold-face upper case for matrices, e.g. $\mathbf{A}$. For a positive integer

[1]Andreea B. Alexandru and George J. Pappas are with the Department of Electrical and Systems Engineering, University of Pennsylvania {aandreea,pappasg}@seas.upenn.edu

[2]Moritz Schulze Darup is with the Automatic Control Group, Department of Electrical Engineering, Universität Paderborn moritz.schulze.darup@upb.de

n, let $[n] := \{1, 2, \dots, n\}$. $\mathbb{N}$ denotes the set of non-negative integers, $\mathbb{Z}$ denotes the set of integers, $\mathbb{Z}/N\mathbb{Z}$ denotes the additive group of integers modulo N and $(\mathbb{Z}/N\mathbb{Z})^*$ denotes the multiplicative group of integers modulo N. λ denotes the security parameter. $\mathrm{E}(x)$ denotes an encrypted scalar value x. An encryption of a matrix $\mathbf{A}$ is denoted by $\mathrm{E}(\mathbf{A})$ and signifies the element-wise encryptions of its elements. $\phi(N)$ denotes Euler's totient function and for $N = pq$, with p, q primes, $\phi(N) = (p-1)(q-1)$.

II. Problem statement

We consider a control scheme tailored for multi-agent systems with linear dynamics. Specifically, consider a system with M agents that obey the local dynamics:

$$\mathbf{x}_i(t+1) = \mathbf{A}_i\mathbf{x}_i(t) + \mathbf{B}_i\mathbf{u}_i(t), \quad \mathbf{x}_i(0) = \mathbf{x}_{i,0}, \tag{1}$$

with $\mathbf{x}_i \in \mathbb{R}^{n_i}$ and $\mathbf{u}_i \in \mathbb{R}^{m_i}$, for every $i \in [M]$. Assume further that the agents are part of a simple, connected, fixed and undirected communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with the vertex $\mathcal{V} = [M]$ and the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. An edge $(i, j) \in \mathcal{E}$ specifies that agent i can communicate with agent j, in which case we say agent i and agent j are neighbors.

Following the cooperative structured control described in [20], we use the local control laws to stabilize the systems:

$$\mathbf{u}_i(t) = \mathbf{K}_{ii}\mathbf{x}_i(t) + \sum_{j \in \mathcal{N}_i} \mathbf{K}_{ij}\mathbf{x}_j(t), \tag{2}$$

where $\mathcal{N}_i := \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ represents the set of neighbors of agent i. The local control laws (2) result from the design of a centralized linear controller of the form:

$$\begin{pmatrix} \mathbf{u}_1(t) \\ \vdots \\ \mathbf{u}_M(t) \end{pmatrix} = \begin{pmatrix} \mathbf{K}_{11} & \dots & \mathbf{K}_{1M} \\ \vdots & \ddots & \vdots \\ \mathbf{K}_{M1} & \dots & \mathbf{K}_{MM} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1(t) \\ \vdots \\ \mathbf{x}_M(t) \end{pmatrix} \tag{3}$$

that takes into account the structural constraints of the communication graph by requiring $\mathbf{K}_{ij} = \mathbf{0}$ whenever $j \notin \mathcal{N}_i \cup \{i\}$. A method for the computation of such structured feedback matrices can, for instance, be found in [21].

A. Privacy requirements

Each agent $i \in \mathcal{V}$ has to compute its local control action, based on its local state and the states of its neighbors, according to (2), i.e., it locally aggregates the contributions of its neighbors $j \in \mathcal{N}_i$. Under privacy requirements, we call such a scheme a private control update aggregation (pCUA), which we would like to achieve the following:

- agent i can learn only its local state and control action $\mathbf{x}_i(t)$ and $\mathbf{u}_i(t)$ at each time period, and nothing else about $\mathbf{x}_j(t), \mathbf{K}_{ij}$, including partial information like $\mathbf{K}_{ij}\mathbf{x}_j(t)$;
- without knowing agent i's aggregator capability, the other agents cannot learn anything about the private states and control gains of the other participants in the computation;
- if agent i colludes with a subset of the agents, it inevitably learns the sum of the contributions of the remaining honest agents, but learns nothing more about their private data.

The adversarial model we consider is honest but curious, which means that an adversary wants to infer the private data of the honest agents, without diverging from the established protocol. Such a model is reasonable since all agents are interested in obtaining the correct result of the computation.

Furthermore, we require the pCUA scheme to be efficient:

- the control action $\mathbf{u}_i(t)$ has to be privately computed in less time than the sampling period.

Shortcomings of the previous scheme. Roughly speaking, [20] introduces a private computation and exchange of the "input portions" from the local control laws (2):

$$\mathbf{v}_{ij}(t) := \mathbf{K}_{ij}\mathbf{x}_j(t) \in \mathbb{R}^{m_i} \tag{4}$$

that reveal neither the exact local state $\mathbf{x}_j$ nor the local controller matrix $\mathbf{K}_{ij}$ to agent i, but can at least leak the relative rate of decrease/increase of some signals of its neighbors over multiple time steps.

However, in order to compute the local control action $\mathbf{u}_i$, agent i does not require access to the individual $\mathbf{v}_{ij}$. In fact, if we want to ensure privacy of the data that is unknown to agent i, the best we can do is to make agent i to only be able to compute the "aggregated portions":

$$\mathbf{u}_i(t) - \mathbf{v}_{ii}(t) = \mathbf{v}_{i,\mathcal{N}_i}(t) := \sum_{j \in \mathcal{N}_i} \mathbf{v}_{ij}(t). \tag{5}$$

Providing $\mathbf{v}_{i,\mathcal{N}_i}$ instead of $\mathbf{v}_{ij}$, without revealing any information about $\mathbf{v}_{ij}$ (for $|\mathcal{N}_i| \geq 2$) is non-trivial. There has been substantial work dealing with the underlying problem of private sum aggregation. The combination of these methods with the encrypted control scheme from [20] is likewise demanding due to the requirement that the control gains are concealed from all the agents.

For simplicity, in the schemes in Sections III and IV, we assume agent i is the aggregator – computes the local control update – and agents j, for $j \in \mathcal{N}_i$, are the other participants. Moreover, we consider states and control gains to be scalars $x_j(t), k_{ij}$. Then, in Section V, we show how each agent calls the scheme for their own local control update and how to deal with the fact that and control actions are vectors.

III. Private weighted sum aggregation

Private sum aggregation (pSA) allows an untrusted aggregator to compute the sum of the private data contributed by some users, without learning the individual contributions. pSA was introduced in [22], [23] and improvements have been proposed, e.g., in [24]–[26].

Shi et al. [22] also introduced a formal definition of *aggregator obliviousness* that pSA schemes have to satisfy. Essentially, the privacy notion of aggregator obliviousness states that the aggregator is not allowed to learn anything more than the aggregate value of the contribution of the participants, for all time steps. Moreover, if an adversary corrupts both the aggregator and some participants, it is not capable of learning anything more than the aggregate contribution of the honest participants.

Private sum aggregation is of independent interest and has countless applications in both centralized and distributed systems. In particular, for the cooperative control scheme we discussed in Section II, *if agent j also had access to the associated control gain k_{ij}*, i.e., the control portion $v_{ij}(t)$ could be computed in plaintext at agent j, we would use pSA. In this case, the private sum aggregation problem can be described as follows: let there be $|\mathcal{N}_i|$ participants and an aggregator i. In every time step, denoted by $t \in \mathbb{N}$, each participant $j \in \mathcal{N}_i \cup i$ holds a private value $x_j(t)$ and k_{ij}. Define $v_{ij}(t) := k_{ij}x_j(t)$. The aggregator wants to compute the aggregate statistics over the private values, specifically: $u_i(t) = \sum_{j\in\mathcal{N}_i} v_{ij}(t) + v_{ii}(t)$. However, since the control gains are unknown to the agents, we require a private weighted sum aggregation scheme.

A private weighted sum aggregation (pWSA) scheme for weights unknown to all participants is composed of algorithms $\mathrm{pWSA} = (\mathrm{Setup}, \mathrm{Enc}, \mathrm{AggrDec})$. We will describe them in detail in Section III-B and show that they satisfy the privacy definition in Section III-A.

A. Formal privacy definition

We give a description of the privacy definition from Section II-A as a typical cryptographic game between an adversary and a challenger, where the adversary $\mathcal{A}$ can corrupt agents. The security game pWSAO (private Weighted Sum Aggregator Obliviousness) is as follows:

Setup. The challenger runs the Setup algorithm and gives the public parameters prm to the adversary.

Queries. The adversary can submit compromise queries and encryption queries that are answered by the challenger. In the case of compromise queries, the adversary submits an index $j \in \mathcal{N}_i \cup i$ to the challenger and receives sk_j, which means the adversary corrupts agent j. The set of the corrupted agents is denoted by $\mathcal{C}$. In the case of encryption queries, the adversary is allowed one query per time step t and per agent $j \in \mathcal{N}_i$. The adversary submits $(j, t, k^a_{ij}, x_j(t))$ and the challenger returns $\mathrm{Enc}(\mathrm{prm}, \mathrm{sk}_j(k^a_{ij}), t, x_j(t))$. The set of participants for which an encryption query was made by the adversary at time t is denoted by $\mathcal{E}(t)$.

Challenge. The adversary chooses a specific time step t^*. Let $\mathcal{U}^*$ denote the set of participants that were not compromised at the end of the game and for which no encryption query was made at time t^*, i.e., $\mathcal{U}^* = (\mathcal{N}_i \cup i) \setminus (\mathcal{C} \cup \mathcal{E}(t^*))$. The adversary specifies a subset of participants $\mathcal{S}^* \subseteq \mathcal{U}^*$. At this time t^*, for each agent $j \in \mathcal{S}^* \setminus i$, the adversary chooses two plaintext series $x^0_j(t^*)$ and $x^1_j(t^*)$, along with $k^{*,0}_{ij}$ and $k^{*,1}_{ij}$, and sends them to the challenger. If $\mathcal{S}^* = \mathcal{U}^*$ and $i \notin \mathcal{S}^*$, i.e., the aggregator has been compromised, then, the values submitted by the adversary have to satisfy $\sum_{j\in\mathcal{S}^*} k^{*,0}_{ij} x^0_j(t^*) = \sum_{j\in\mathcal{S}^*} k^{*,1}_{ij} x^1_j(t^*)$. The challenger flips a random bit $b \in \{0,1\}$ and computes $\mathrm{Enc}(\mathrm{prm}, \mathrm{sk}_j(k^{*,b}_{ij}), t, x^b_j(t^*)), \forall j \in \mathcal{S}^*$. The challenger then returns the ciphertexts to the adversary.

Guess. The adversary outputs a guess $b' \in \{0,1\}$ on whether b is 0 or 1. The advantage of the adversary is defined as:

$$\mathbf{Adv}^{\mathrm{pWSAO}}(\mathcal{A}) := \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

The adversary wins the game if it correctly guesses b.

Definition 1: A scheme $\mathrm{pWSA} = (\mathrm{Setup}, \mathrm{Enc}, \mathrm{AggrDec})$ achieves *weighted sum aggregator obliviousness* if no probabilistic polynomial-time adversary has more than negligible advantage in winning this security game:

$$\mathbf{Adv}^{\mathrm{pWSAO}}(\mathcal{A}) \leq \eta(\lambda). \qquad \diamond$$

B. Solution

The neighbors of agent i do not have access to the plaintext contribution they have to send to agent i: the control gain k_{ij} should be private from all participants (encrypted), so neighbor j has to find a way to send an encryption of the masked product $k_{ij}x_j(t)$ to agent i, and the latter still has to decrypt the result. This suggests that:

- k_{ij} should be encrypted with an additively homomorphic encryption that agent i knows how to decrypt;
- the layer of encryption introduced in Enc should be compatible with the inner additively homomorphic layer;
- agent i should not be able to decrypt the individual contributions it receives from its neighbors, despite having the secret key of the homomorphic encryption scheme.

To achieve the solution, for the outer layer of encryption, we can use one-time pads, which are compatible with the additively homomorphic property. For the inner layer of encryption, we need an asymmetric additive homomorphic encryption scheme. We will instantiate it with the Paillier cryptosystem [1], due to its simplicity and popularity. More details about Paillier's cryptosystem can be found in Appendix A. We will denote the Paillier encryption primitive by $\mathrm{E}(\cdot)$ and the decryption primitive by $\mathrm{D}(\cdot)$.

Hence, the steps of the algorithms in pWSA are:

- $\mathrm{Setup}(1^\lambda, \mathcal{N}_i, \{k_{ij}\}_{j\in\mathcal{N}_i\cup i}, T)$: given security parameter λ, get a pair of Paillier keys $(\mathfrak{pk}, \mathfrak{sk})$: generate two equal-size prime numbers p, q and set $N = pq$ such that $\lfloor \log_2 N \rfloor = \lambda$, $\gcd(\phi(N), N) = 1$. Set $g = 1 + N$ and:
$$\mathfrak{pk} = (g, N), \ \mathfrak{sk} = \big(\phi(N), \phi(N)^{-1} \bmod N\big).$$
For every $t \in [T]$, generate $|\mathcal{N}_i| + 1$ shares of zero:
$$\sum_{j\in\mathcal{N}_i\cup i} s_j(t) = 0 \bmod N, \quad s_j(t) \in (\mathbb{Z}/N\mathbb{Z})^*.$$
Denote by $\mathbf{s}_j$ the vector of $s_j(t)$ for all $t \in [T]$. Encrypt k_{ij} for $i \neq j$ using $\mathfrak{pk}$: $\mathrm{E}(k_{ij}) = g^{k_{ij}} r^N \bmod N^2$, for r randomly sampled from $(\mathbb{Z}/N^2\mathbb{Z})^*$. Finally, set $\mathrm{prm} = (\lambda, \mathfrak{pk})$, $\mathrm{sk}_j = (\mathbf{s}_j, \mathrm{E}(k_{ij}))$ and $\mathrm{sk}_i = (\mathfrak{sk}, \mathbf{s}_i, k_{ii})$.
- $\mathrm{Enc}(\mathrm{prm}, \mathrm{sk}_j, t, x_j(t))$: for $x_j(t) \in \mathbb{Z}/N\mathbb{Z}$, compute:
$$c_j(t) = \mathrm{E}(k_{ij})^{x_j(t)} \mathrm{E}(s_j(t)) = \mathrm{E}(k_{ij}x_j(t) + s_j(t)).$$
- $\mathrm{AggrDec}(\mathrm{prm}, \mathrm{sk}_i, t, \{c_j(t)\}_{j\in\mathcal{N}_i})$: compute $V(t) = \prod_{j\in\mathcal{N}_i} c_j(t) \bmod N^2$ and then set:
$$u_i(t) = \big(\mathrm{D}(V(t)) + s_i(t)\big) \bmod N + k_{ii}x_i(t).$$

An assumption that we make for the rest of the paper is:

Assumption 1: For each time step t, $x_j(t), k_{ij}(t), v_{ij}(t), u_i(t) \in \mathbb{Z}/N\mathbb{Z}$, $\forall i, j \in \mathcal{V}$, i.e., there is no overflow. $\diamond$

Correctness: $\mathrm{D}(V(t)) = \sum_{j\in\mathcal{N}_i} k_{ij}x_j(t) + s_j(t)$ follows from the correct execution of Paillier operations in Enc. Then, $\mathrm{D}(V(t)) + s_i(t) = \sum_{j\in\mathcal{N}_i} k_{ij}x_j(t) \bmod N$ from the generation of shares of zero. After adding $k_{ii}x_i(t)$, agent i obtains the desired control action $u_i(t)$ as specified in (2).

Theorem 1: The pWSA scheme achieves weighted sum aggregator obliviousness w.r.t. Definition 1. $\diamond$

Proof: We are going to treat two cases: I, the adversary does not corrupt the aggregator and II, the adversary corrupts the aggregator:

$$\Pr[b' = b] = \frac{1}{2}\Pr[b' = b | i \notin \mathcal{C}] + \frac{1}{2}\Pr[b' = b | i \in \mathcal{C}].$$

A function $\eta : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ is called negligible if for all $c \in \mathbb{R}_{>0}$, there exists $n_c \in \mathbb{Z}_{\geq 1}$ such that for all integers $n \geq n_c$, we have $|\eta(n)| \leq n^{-c}$.

We will consider the stronger case where $\mathcal{S}^* = \mathcal{U}^*$; the weaker case where $\mathcal{S}^* \subseteq \mathcal{U}^*$ easily follows.

I. $i \notin \mathcal{C}$. From the compromise queries, the adversary holds the following information $\{\lambda, \mathfrak{pk}, \{s_j(t)\}_{j\in\mathcal{C}}, \{k_{ij}\}_{j\in\mathcal{C}}\}_{t\in[T]}$ and $\sum_{j\in\mathcal{U}} s_j(t) = -\sum_{j\in\mathcal{C}} s_j(t)$, for all $t \in [T]$. From the encryption queries at time t, the adversary knows $\{c_j(t) = \mathrm{E}(k^a_{ij}x_j(t)) + s_j(t))\}_{j\in\mathcal{E}(t)}$. Then, the adversary chooses $t^* \in T$ and a series of $\{x^0_j(t^*)\}_{j\in\mathcal{U}^*}$ and $\{x^1_j(t^*)\}_{j\in\mathcal{U}^*}$, along with $k^{*,0}_{ij}$ and $k^{*,1}_{ij}$ and receives from the challenger $\{c_j(t^*) = \mathrm{E}(k^{*,b}_{ij}x^b_j(t^*)) + s_j(t^*))\}_{j\in\mathcal{U}^*}$.

Because the adversary doesn't have the secret key of the Paillier scheme and does not have the individual secrets of the uncorrupted agents:

$$\begin{aligned} &\Pr[\mathcal{A} \text{ breaks Paillier scheme}] \leq \eta_1(\lambda),\\ &\Pr[\mathcal{A} \text{ breaks secret sharing}] \leq \eta_2(\lambda),\\ &\Pr[b' = b | i \notin \mathcal{C}] \leq \frac{1}{2} + \eta_1(\lambda)\eta_2(\lambda), \end{aligned} \tag{6}$$

where $\eta_1(\lambda), \eta_2(\lambda)$ are negligible functions, according to Theorems A.1 and A.2.

II. $i \in \mathcal{C}$. From the compromise queries, the adversary holds the following information $\{\lambda, \mathfrak{pk}, \{s_j(t)\}_{j\in\mathcal{C}}, \{k_{ij}\}_{j\in\mathcal{C}}, \mathfrak{sk}\}_{t\in[T]}$, and $\sum_{j\in\mathcal{U}} s_j(t) = -\sum_{j\in\mathcal{C}} s_j(t)$, for all $t \in [T]$. From the encryption queries, and after using $\mathfrak{sk}$ to decrypt, the adversary knows $\{p_j(t) = k^a_{ij}x_j(t) + s_j(t)\}_{j\in\mathcal{E}(t)}$. Then, the adversary chooses $t^* \in T$ and a series of $\{x^0_j(t^*)\}_{j\in\mathcal{U}^*}$ and $\{x^1_j(t^*)\}_{j\in\mathcal{U}^*}$, along with $k^{*,0}_{ij}$ and $k^{*,1}_{ij}$, such that $\sum_{j\in\mathcal{U}^*} k^{*,0}_{ij}x^0_j(t^*) = \sum_{j\in\mathcal{U}^*} k^{*,1}_{ij}x^1_j(t^*)$ and receives from the challenger $\{c_j(t^*) = \mathrm{E}(k^{*,b}_{ij}x^b_j(t^*)) + s_j(t^*)\}_{j\in\mathcal{U}^*}$. The adversary uses the secret key of the Paillier scheme to decrypt the individual ciphertexts and obtains $p_j(t^*) = k^{*,b}_{ij}x^b_j(t^*) + s_j(t^*)$, for $j \in \mathcal{U}^*$. Because the secret shares of zero are different for each time $t \neq t^*$, the adversary cannot infer information about the challenge query from the previous encryption queries.

Then, the probability that the adversary wins is the probability that the adversary breaks secret sharing:

$$\begin{aligned} &\Pr[\mathcal{A} \text{ breaks secret sharing}] \leq \eta_2(\lambda),\\ &\Pr[b' = b | i \in \mathcal{C}] \leq \frac{1}{2} + \eta_2(\lambda). \end{aligned} \tag{7}$$

From (6) and (7): $\mathbf{Adv}^{\mathrm{pWSA}}(\mathcal{A}) \leq \eta_2(\lambda)$. ■

The above scheme is appealing due to its simplicity, but involves demanding communication, because different secret shares of zero are required at every time step t for all the neighbors of agent i. The Setup is executed by an incorruptible trusted third party, called dealer. This dealer cannot be online at every time step to distribute the shares because, otherwise, this party could act as a trusted centralized controller. A more reasonable assumption is that, prior to the online computations, the dealer computes the shares for T time steps and sends them to the agents, who have to store them. Alternatively, we also offer a solution to generate the secret shares of zero in a distributed way, without the need of a trusted third party.

IV. Distributed generation of zero shares

Assume each agent knows the agents that are two hops away from itself. The scheme for distributedly generating shares of zero for the update computed at agent i for time t has the following steps:

1) At time $t-1$, each agent $j \in \mathcal{N}_i \cup i$ sends shares of zero $\sigma^i_{jl}(t) \in (\mathbb{Z}/N\mathbb{Z})^*$ to itself and to the agents in the intersection of its neighbors and the neighbors of agent i:

$$\sum_{l\in(\mathcal{N}_j\cup j)\cap(\mathcal{N}_i\cup i)} \sigma^i_{jl}(t) = 0 \bmod N. \tag{8}$$

2) At time t, each agent $j \in \mathcal{N}_i \cup i$ sums its own share and the shares it received meant for the aggregation at i:

$$s_{ij}(t) := \sum_{l\in(\mathcal{N}_j\cup j)\cap(\mathcal{N}_i\cup i)} \sigma^i_{lj}(t). \tag{9}$$

From (8), it is clear that:

$$\sum_{j\in\mathcal{N}_i\cup i}\ \sum_{l\in(\mathcal{N}_j\cup j)\cap\mathcal{N}_i} \sigma^i_{jl}(t) = 0 \bmod N.$$

Then, we confirm that we obtained shares of zero for agent i and $j \in \mathcal{N}_i$:

$$\sum_{j\in\mathcal{N}_i\cup i} s_{ij}(t) = 0 \bmod N.$$

For the centralized solution of creating shares of zero considered in Section III-B, the privacy of one agent $j \in \mathcal{N}_i \cup i$ is guaranteed as long as the number of colluding agents is strictly less than $|\mathcal{N}_i \cup i| - 1$ (otherwise the share of agent j can be computed from the shares of the colluding agents). For the decentralized solution, the privacy of one agent $j \in \mathcal{N}_i \cup i$ is guaranteed as long as the number of colluding agents is strictly less than $|(\mathcal{N}_j \cup j) \cap (\mathcal{N}_i \cup i)|$. Hence, this scheme is more robust the more neighbors an agent has. At the same time, the more neighbors an agent has, the larger the number of messages it sends and receives.

If each agent is sufficiently connected (has a vertex degree larger than a desired threshold), the above observation does not represent a problem. On the other hand, if the agents are not sufficiently connected, we can enforce dummy connections such that each agent reaches a desired vertex degree, by connecting them to neighbors of the aggregator agent.

Furthermore, the bandwidth overhead can be reduced if, instead of sending the full secret $\sigma \in (\mathbb{Z}/N\mathbb{Z})^*$, the agents send only a smaller seed $\tau \in \mathbb{Z}/w\mathbb{Z}$, $w << N$ for a pseudorandom generator function (e.g. a hash function) $H : \mathbb{Z}/w\mathbb{Z} \to (\mathbb{Z}/N\mathbb{Z})^*$, that is publicly known. The above scheme can be modified as follows:

1) At time $t-1$, each agent $j \in \mathcal{N}_i \cup i$ generates random seeds for each agent $l \in \mathcal{N}_j \cap (\mathcal{N}_i \cup i)$: $\tau^i_{jl}(t) \in \mathbb{Z}/w\mathbb{Z}$, and computes for itself:
$$\sigma^i_{jj}(t) = -\sum_{l \in \mathcal{N}_j \cap (\mathcal{N}_i \cup i)} H\big(\tau^i_{jl}(t)\big) \bmod N.$$
2) At time t, each agent $j \in \mathcal{N}_i \cup i$ sums the outputs of the hash function on the seeds it received from its neighbors that participated in the computation and its own share:
$$s_{ij}(t) := \sum_{l \in \mathcal{N}_j \cap (\mathcal{N}_i \cup i)} H\big(\tau^i_{lj}(t)\big) + \sigma^i_{jj}(t). \tag{10}$$

V. Control Update Aggregation Scheme

We now show how to apply the weighted sum aggregation scheme, defined for scalars, in our control scenario in order to obtain a private control update scheme. The states and control actions are vectors, and the control gains are matrices, hence the algorithms in pWSA have to be run multiple times.

Consider the private control update aggregation scheme $\text{pCUA} = (\text{Setup}_{\text{pCUA}}, \text{Enc}_{\text{pCUA}}, \text{AggrDec}_{\text{pCUA}})$:

- $\text{Setup}_{\text{pCUA}}(1^\lambda, \mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{B}, \text{cost}, T)$: The dealer generates the control gain $\mathbf{K}$ as in (3) using $\mathbf{A}, \mathbf{B}$ and a cost. For every agent $i \in \mathcal{V}$, run Setup and generate a pair of keys $\mathfrak{pk}_i = (g_i, N_i)$, $\mathfrak{sk}_i = \big(\phi(N_i), \phi(N_i)^{-1} \bmod N_i\big)$. Denote by $\mathfrak{pk}$ the vector of all public keys. For every $t \in [T]$, generate with Setup m_i sets of shares of zero:
$$\sum_{j \in \mathcal{N}_i \cup i} s^k_{ij}(t) = 0 \bmod N_i,\ s^k_{ij}(t) \in (\mathbb{Z}/N_i\mathbb{Z})^*,\ \forall k \in [m_i].$$
Denote by $\mathbf{S}_{ij}$ the matrix of $s^k_{i,j}(t)\ \forall t \in [T]$ and $k \in [m_i]$. Encrypt $\mathbf{K}_{ji}$ for $i \neq j$ using $\mathfrak{pk}_i$, for $j \in \mathcal{N}_i$. Finally, set $\text{prm} = (\lambda, \mathfrak{pk})$, $\text{sk}_i = (\mathfrak{sk}_i, \mathbf{K}_{ii}, \mathbf{S}_{ii}, \text{E}(\mathbf{K}_{ji}), \mathbf{S}_{j \in \mathcal{N}_i, i})$.
- $\text{Enc}_{\text{pCUA}}(\text{prm}, \text{sk}_i, t, \mathbf{x}(t))$: For each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, set, for each $k \in [m_i]$, agents j compute:
$$\begin{aligned} c^k_{i,j}(t) &= \prod_{l=1}^{n} \text{Enc}\big(\text{prm}, (s^k_{ij}(t), \mathbf{K}^{k,:}_{ij}), t, \mathbf{x}^l_j(t)\big) \\ &= \text{E}(\mathbf{K}^{k,:}_{ij}\mathbf{x}_j(t) + s^k_{ij}(t)). \end{aligned}$$
Denote the vector of $c^k_{ij}(t)$ for all $k \in [m_i]$ as $\mathbf{c}_{ij}(t)$.
- $\text{AggrDec}_{\text{pCUA}}(\text{prm}, \text{sk}_i, t, \{\mathbf{c}_{ij}(t)\}_{i,j \in \mathcal{N}_i})$: For $i \in \mathcal{V}$:
$$\begin{aligned} u^k_i(t) &= \text{AggrDec}(\text{prm}, (s^k_{ii}(t), \mathbf{K}^{k,:}_{ii}), t, \{\mathbf{c}^k_{ij}(t)\}_{j \in \mathcal{N}_i}) \\ &= \sum_{j \in \mathcal{N}_i} \mathbf{K}^{k,:}_{ij}\mathbf{x}_j(t) + \mathbf{K}^{k,:}_{ii}\mathbf{x}_i(t). \end{aligned}$$

In a pWSAO security game, the adversary can also corrupt other agents $j \notin \mathcal{N}_i \cup i$, but this brings no extra information: the games are independent for different local control update aggregations because the keys are different and independent. Hence, the security of the pCUA scheme follows from the composition of the pWSA schemes for each agent.

VI. Observability Analysis

This section motivates the benefits of aggregation for the privacy of the agents from a system-theoretic point of view.

A. Available information without aggregation

In the encrypted control scheme [20], agent i has access to the input portions $\mathbf{v}_{ij}(t)$ of all neighboring agents $j \in \mathcal{N}_i$. Thus, it can use (4) and stored data to estimate the matrices $\mathbf{K}_{ij}$ or the neighboring states $\mathbf{x}_j(t)$. Under the assumption that $m_i \leq n_i$, the corresponding system of equations:
$$\begin{pmatrix}\mathbf{v}_{ij}(0) & \dots & \mathbf{v}_{ij}(t)\end{pmatrix} = \mathbf{K}_{ij}\begin{pmatrix}\mathbf{x}_j(0) & \dots & \mathbf{x}_j(t)\end{pmatrix} \tag{11}$$
is underdetermined for every t. However, taking application-related restrictions on $\mathbf{K}_{ij}$ and $\mathbf{x}_j$ into account, it might be possible to reconstruct the constant matrix $\mathbf{K}_{ij}$. Moreover, especially for systems with similar agents, such as robot swarms, agent i might obtain information about the dynamics of agent j in terms of $\mathbf{A}_j$ and $\mathbf{B}_j$. The ability to reconstruct $\mathbf{x}_j$ from observations $\mathbf{v}_{ij}$ given $\mathbf{K}_{ij}$, $\mathbf{A}_j$, and $\mathbf{B}_j$ then becomes an observability problem. In fact, observing $\mathbf{x}_j$ requires full rank of the observability matrix:
$$\mathbf{V}_{ij} := \begin{pmatrix}\mathbf{K}_{ij}\mathbf{A}^0_j \\ \vdots \\ \mathbf{K}_{ij}\mathbf{A}^{n_j-1}_j\end{pmatrix}.$$

B. Available information with aggregation

In the aggregated case described in Section III, less data is available to agent i. The system of equations (11) becomes:
$$\begin{pmatrix}\mathbf{v}_{i,\mathcal{N}_i}(0) & \dots & \mathbf{v}_{i,\mathcal{N}_i}(t)\end{pmatrix} = \mathbf{K}_{i,\mathcal{N}_i}\begin{pmatrix}\mathbf{x}_{\mathcal{N}_i}(0) & \dots & \mathbf{x}_{\mathcal{N}_i}(t)\end{pmatrix},$$
$$\text{with } \mathbf{K}_{i,\mathcal{N}_i} := \begin{pmatrix}\mathbf{K}_{ij_1} & \dots & \mathbf{K}_{ij_{|\mathcal{N}_i|}}\end{pmatrix}, \quad \mathbf{x}_{\mathcal{N}_i} := \begin{pmatrix}\mathbf{x}_{j_1} \\ \vdots \\ \mathbf{x}_{j_{|\mathcal{N}_i|}}\end{pmatrix},$$
and $\mathcal{N}_i = \{j_1, \dots, j_{|\mathcal{N}_i|}\}$. For $|\mathcal{N}_i| > 1$, the degree of freedom here is significantly higher than in (11). Still, it might again be possible to infer the constant matrix $\mathbf{K}_{i,\mathcal{N}_i}$. In addition, agent i might know the neighboring dynamics:
$$\mathbf{A}_{\mathcal{N}_i} := \textbf{blkdiag}(\mathbf{A}_{j_1}, \dots, \mathbf{A}_{j_{|\mathcal{N}_i|}}).$$
However, the observability of $\mathbf{x}_{\mathcal{N}_i}$ now requires full rank of:
$$\mathbf{V}_{i,\mathcal{N}_i} := \begin{pmatrix}\mathbf{K}_{i,\mathcal{N}_i}\mathbf{A}^0_{\mathcal{N}_i} \\ \vdots \\ \mathbf{K}_{i,\mathcal{N}_i}\mathbf{A}^{n_j-1}_{\mathcal{N}_i}\end{pmatrix}.$$

It is then straightforward to obtain the following result.

Lemma 1: Having full rank of $\mathbf{V}_{i,\mathcal{N}_i}$ implies full rank of $\mathbf{V}_{ij}$ for every $j \in \mathcal{N}_i$. ◇

Lemma 1 states that observability in the aggregated case implies observability in the unaggregated case. In other words, observability in the aggregated case is less likely, which increases the privacy of the agents. We omit a formal proof due to space restrictions. We stress, however, that:
$$\mathbf{K}_{i,\mathcal{N}_i}\mathbf{A}^k_{\mathcal{N}_i} = \begin{pmatrix}\mathbf{K}_{ij_1}\mathbf{A}^k_{j_1} & \dots & \mathbf{K}_{ij_{|\mathcal{N}_i|}}\mathbf{A}^k_{j_{|\mathcal{N}_i|}}\end{pmatrix},$$

in combination with an analysis of the $|\mathcal{N}_i|$ column-blocks of $\mathbf{V}_{i,\mathcal{N}_i}$ and the Cayley-Hamilton theorem leads to Lemma 1.

Note that the converse statement of Lemma 1 is, in general, not true.

VII. Numerical examples

As usual when operating with encryption schemes defined on groups of modular residues, we need to use a fixed-point representation scheme in order to quantize the real values and encode them into integers. The effect of this quantization on the cooperative control scheme is described in [20]. In the simulations, we choose a representation on $l = 64$ bits: 32 integer bits and 32 fractional bits, which makes the quantization errors negligible. We also choose all the Paillier moduli to have 1024 bits. This ensures Assumption 1, i.e., all values will be represented on fewer bits than $\min_{i\in[M]} \log N_i$.

For illustration purposes, we consider a network of 50 agents, with each agent having local states of dimension 4 and local control inputs of dimension 2. We simulate pCUA for various values of the average node degree in the network, obtained by varying the probability of drawing edges between agents. Simulations were run in Python 3 on a 2.2 GHz Intel Core i7 processor.

Figure 1 shows the online running times (averaged over 500 instances) for the local computation at each agent in a time step using the scheme described in Section III-B. The computation times are of the order of at most tens of milliseconds for all neighbor degrees considered. However, the offline setup phase becomes slower as the average number of neighbors and time steps increase; the offline share generation ranges from 16 to 83 seconds for the average degree of 4 to 23 agents for 100 time steps.

If the offline phase computed for a large number of time steps is slower than how long it will take to actually execute those time steps (for example when the sampling time is very small), it is better to have the agents compute their shares of zero online. Figure 2 shows the running times (averaged over 500 instances) for the local computation at each agent in a time step using the scheme described in Section IV jointly with the scheme from Section III-B. As expected, the execution times increased roughly tenfold, but the scheme is still competitive time-wise. Moreover, in the offline phase, only one set of shares of zero is now computed (to be used in the first time step), ranging from 0.7 to 1.4 seconds.

VIII. Conclusions

The main contributions of this work are the following. We proposed a private weighted sum aggregation scheme with secret weights, that can be used to achieve the private control update aggregation scheme, where each agent acts as an aggregator for the contributions of its neighbors. We gave a formal privacy proof of the scheme, using the information-theoretic privacy of secret sharing and the semantic security of the additively homomorphic cryptosystem. We also proposed a method for generating the secrets in a distributed way, rather than by a third party. Finally, we

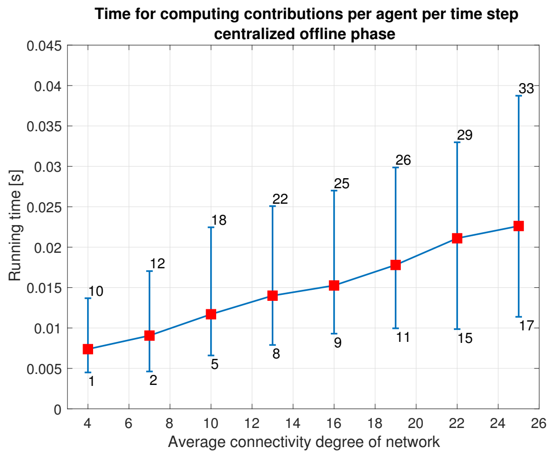

Fig. 1. Average running times for the pCUA scheme with the steps described in Section III-B on a network of 50 agents. The average degree of the network is given on the x axis, while the minimum and maximum degrees in the network are shown on the plot.

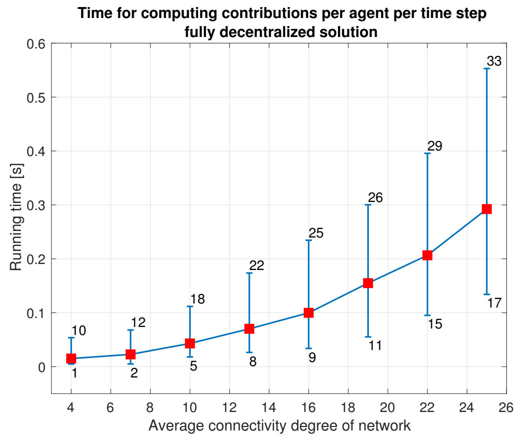

Fig. 2. Average running times for the pCUA scheme with the steps described in Section IV on a network of 50 agents. The average degree of the network is given on the x axis, while the minimum and maximum degrees in the network are shown on the plot.

provided numerical results for the schemes for various connectivity degrees of a network and showed their efficiency.

The proposed scheme achieves the desired privacy goal, but generates secret shares of zero at every time step. We will investigate schemes such as in [26] with two secret keys but only one zero share for the whole period of time.

Appendix

A. Additively homomorphic encryption

Consider the additive group of integers modulo N, $\mathbb{Z}/N\mathbb{Z}$, where $N = pq$ is a large modulus composed of two prime numbers of equal bit-length, p and q, such that $\gcd(\phi(N), N) = 1$. The order of $\mathbb{Z}/N\mathbb{Z}$ is $\phi(N) = (p-1)(q-1)$. Now consider the multiplicative group of integers modulo N^2, $(\mathbb{Z}/N^2\mathbb{Z})^*$. The order of $(\mathbb{Z}/N^2\mathbb{Z})^*$ is $N\phi(N)$. An important subgroup of $(\mathbb{Z}/N^2\mathbb{Z})^*$ is:

$$\begin{aligned}\Gamma_N :=& \{(1+N)^\alpha \bmod N^2 | \alpha \in \{0,\dots,N-1\}\}\\ =& \{1+\alpha N | \alpha \in \{0,\dots,N-1\}\},\end{aligned} \tag{12}$$

where the equality follows from the binomial theorem: $(1+N)^\alpha = 1+\alpha N \bmod N^2$. Computing discrete logarithms in Γ_N is easy [1], [24]: given $x, y \in \Gamma_N$, we can find β such that $y = x^\beta \bmod N^2$ by $\beta = (y-1)/(x-1) \bmod N$.

Another important subgroup in $(\mathbb{Z}/N^2\mathbb{Z})^*$ is:

$$\mathfrak{G}_N := \{x^N \bmod N^2 | x \in (\mathbb{Z}/N\mathbb{Z})^*\}. \quad (13)$$

$\mathfrak{G}_N$ has order $\phi(N)$. Computing discrete logarithms in $\mathfrak{G}_N$ is as hard as computing discrete logarithms in $(\mathbb{Z}/N\mathbb{Z})^*$ [24].

We also have the modular equalities for $x \in (\mathbb{Z}/N^2\mathbb{Z})^*$:

$$x^{\phi(N)} = 1 \bmod N, \quad x^{N\phi(N)} = 1 \bmod N^2. \quad (14)$$

The Paillier scheme is defined using the previously described concepts. Specifically, the plaintext space is $\mathbb{Z}/N\mathbb{Z}$ and the ciphertext space is $(\mathbb{Z}/N^2\mathbb{Z})^*$. The public key is (g, N), where g is usually selected to be $1 + N$, and the secret key $\big(\phi(N), (\phi(N))^{-1} \bmod N\big)$. The encryption is:

$$\mathrm{E}(x) = g^x r^N \bmod N^2,$$

where r is sampled uniformly at random from $(\mathbb{Z}/N^2\mathbb{Z})^*$. For a ciphertext $c \in (\mathbb{Z}/N^2\mathbb{Z})^*$, decryption uses equations (12) and (14):

$$\mathrm{D}(c) = (c^{\phi(N)} - 1)/N \cdot \phi(N)^{-1} \bmod N.$$

Thanks to the definition of the encryption primitive, the Paillier scheme allows for homomorphic additions and multiplication by plaintexts, as follows:

$$\mathrm{D}\big(\mathrm{E}(x) \cdot \mathrm{E}(y)\big) = \mathrm{D}\big(\mathrm{E}(x+y)\big) = x + y \bmod N$$
$$\mathrm{D}\big((\mathrm{E}(x))^y\big) = \mathrm{D}\big(\mathrm{E}(xy)\big) = xy \bmod N.$$

Under the Decisional Composite Residuosity assumption (i.e., distinguishing between an element from $\mathfrak{G}_N$ and an element from $(\mathbb{Z}/N^2\mathbb{Z})^*$ is hard), the following holds:

Theorem A.1: The Paillier cryptosystem is semantically secure [1]. $\diamond$

B. Secret sharing

Secret sharing is a tool that distributes a secret message to a number of parties, by splitting it into random shares. Specifically, t-out-of-n secret sharing splits a secret message into n shares and distributes them to different parties; then, the secret message can be reconstructed by an authorized subset of parties, which have to combine at least t shares.

One common scheme is the additive 2-out-of-2 secret sharing scheme, which involves a party splitting its secret message $m \in \mathbb{Z}/N\mathbb{Z}$ into two shares, in the following way: generate uniformly at random an element $s \in \mathbb{Z}/N\mathbb{Z}$, subtract it from the message and then distribute the shares s and $m - s$. This resembles a one-time pad scheme. Both shares are needed in order to recover the secret.

Theorem A.2: Secret sharing is information-theoretically secure [27]. $\diamond$

References

[1] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[2] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[3] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

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

[5] 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.

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

[7] M. Schulze Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted mpc for linear constrained systems," *IEEE Control Systems Letters*, vol. 2, no. 2, pp. 195–200, 2018.

[8] 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 Conference on Decision and Control (CDC)*, 2018, pp. 5020–5025.

[9] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based MPC with encrypted data," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 5014–5019.

[10] M. Ruan, H. Gao, and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Transactions on Automatic Control*, 2019.

[11] M. Kishida, "Encrypted average consensus with quantized control law," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 5850–5856.

[12] C. N. Hadjicostis, "Privacy preserving distributed average consensus via homomorphic encryption," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 1258–1263.

[13] N. M. Freris and P. Patrinos, "Distributed computing over encrypted data," in *54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 1116–1122.

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

[15] H. Xiao, Y. Yu, and S. Devadas, "On privacy-preserving decentralized optimization through alternating direction method of multipliers," *arXiv preprint arXiv:1902.06101*, 2019.

[16] W. Zheng, R. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *IEEE Symposium on Security and Privacy (SP)*, vol. 1, 2019.

[17] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *First International Conference on Smart Grid Communications*. IEEE, 2010, pp. 327–332.

[18] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2011, pp. 175–191.

[19] Z. Erkin, J. R. Troncoso-Pastoriza, R. L. Lagendijk, and F. Pérez-González, "Privacy-preserving data aggregation in smart metering systems: An overview," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 75–86, 2013.

[20] 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.

[21] F. Lin, M. Fardad, and M. R. Jovanović, "Augmented Lagrangian approach to design of structured optimal state feedback gains," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2923–2929, 2011.

[22] E. Shi, H. T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Network & Distributed System Security Symposium (NDSS)*. Internet Society., 2011.

[23] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *ACM SIGMOD International Conference on Management of data*, 2010, pp. 735–746.

[24] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 111–125.

[25] F. Benhamouda, M. Joye, and B. Libert, "A new framework for privacy-preserving aggregation of time-series data," *ACM Transactions on Information and System Security*, vol. 18, no. 3, p. 10, 2016.

[26] D. Becker, J. Guajardo, and K.-H. Zimmermann, "Revisiting private stream aggregation: Lattice-based PSA," in *Network & Distributed System Security Symposium (NDSS)*, 2018.

[27] R. Cramer, I. Damgård, and J. B. Nielsen, "Secure multiparty computation and secret sharing-an information theoretic approach," *Book draft*, 2012.