Differentially Private Formation Control: Privacy and Network Co-Design

Calvin Hawkins and Matthew Hale*

Abstract—As multi-agent systems proliferate, there is increasing need for coordination protocols that protect agents' sensitive information while still allowing them to collaborate. Often, a network system and controller are first designed and implemented, and then privacy is only incorporated after that. However, the absence of privacy from the design process can make it difficult to implement without significantly harming system performance. To address this need, this paper presents a co-design framework for multi-agent networks and private controllers that we apply to the problem of private formation control. Agents' state trajectories are protected using differential privacy, which is a statistical notion of privacy that protects data by adding noise to it. Privacy noise alters the performance of the network, which we quantify by computing a bound on the steady-state mean square error for private formations. Then, we analyze trade-offs between privacy level, system performance, and connectedness of the network’s communication topology. These trade-offs are used to formulate a co-design optimization framework to design the optimal communication topology and privacy parameters for a network running private formation control. Simulation results illustrate the scalability of our proposed privacy/network co-design problem, as well as the high quality of formations one can attain, even with privacy implemented.

I. INTRODUCTION

Multi-agent systems, such as robotic swarms and social networks, require agents to share information to collaborate. In some cases, the information shared between agents may be sensitive. For example, self-driving cars may share location data to be routed to a destination. Geo-location data and other data streams can be quite revealing about users and sensitive data should be protected, though this data must still be useful for multi-agent coordination. Thus, privacy in multi-agent control must simultaneously protect agents’ sensitive data while guaranteeing that privatized data enables the network to achieve a common task.

This type of privacy has recently been achieved using differential privacy. Differential privacy comes from the computer science literature, where it was originally used to protect sensitive data when databases are queried [1], [2]. Differential privacy is appealing because it is immune to post-processing and robust to side information [1]. These properties mean that privacy guarantees are not compromised by performing operations on differentially private data, and that they are not weakened by much by an adversary with additional information about data-producing agents [3].

Recently, differential privacy has been applied to dynamic systems [4]–[12]. One form of differential privacy
in dynamic systems protects sensitive trajectory-valued data, and this is the notion of differential privacy used in this paper. Privacy of this form ensures that an adversary is unlikely to learn much about the state trajectory of a system by observing its outputs. In multi-agent control, this lets an agent share its outputs with other agents while protecting its state trajectory from those agents and eavesdroppers \[4\]–\[7\].

In this paper, we develop a framework for private multi-agent formation control using differential privacy. Formation control is a well-studied network control problem that can represent, e.g., robots physically assembling into geometric shapes or non-physical agents maintaining relative state offsets. For differential privacy, agents add privacy noise to their states before sharing them with other agents. The other agents use privatized states in their update laws, and then this process repeats at every time step.

This paper focuses on private formation control, though the methods presented can be used (with only minor modifications) to design and analyze private consensus-style protocols, which underlie many multi-agent control and optimization algorithms, as well as coverage controllers and others with linear Laplacian dynamics \[13\], \[14\]. The private formation control protocol we present can be implemented in a completely distributed manner, and, contrary to some existing privacy approaches, it does not require a central coordinator.

In many control applications, privacy is only a post-hoc concern that is incorporated after a network and/or a controller is designed, which can make privacy difficult to implement. Therefore, this paper formulates a co-design problem to design a network topology and a differential privacy implementation together. This problem accounts for (i) the strength of privacy protections, (ii) the formation control error induced by privacy, and (iii) the topology of the network that runs the formation control protocol. The benefits of co-design have been illustrated for problems of security in control systems \[15\] and the co-design framework in this paper brings these same benefits to problems in privacy.

A preliminary version of this paper appeared in \[16\]. This paper adds the co-design framework, closed-form solution to the steady-state formation error covariance, new simulations, and proofs of all results. The rest of this paper is organized as follows. Section II gives graph theory and differential privacy background. Section III provides formal problem statements and outlines how privacy can be implemented in formation control. Section IV analyzes the performance of the private formation control protocol. In Section V, we define, analyze, and provide methods to solve the privacy/network co-design problem. Next, in Section VI we provide numerical examples of privacy/network co-design, and Section VII provides concluding remarks.

### Notation

\[I_a \in \mathbb{R}^{a \times a}\] is the identity matrix in \(a\) dimensions, and \(\mathbf{1}\) is the vector of all ones in \(\mathbb{R}^N\). Other symbols are defined as they are used.

## II. Background and Preliminaries

In this section we briefly review the required background on graph theory and differential privacy.

### A. Graph Theory Background

A graph \(G = (V,E)\) is defined over a set of nodes \(V\) and edges are contained in the set \(E\). For \(N\) nodes, \(V\) is indexed over \(\{1,...,N\}\). The edge set of \(G\) is a subset \(E \subseteq V \times V\), where the pair \((i,j) \in E\) if nodes \(i\) and \(j\) share a connection and \((i,j) \notin E\) if they do not. This paper considers undirected, weighted, simple graphs. Undirectedness means that an edge \((i,j) \in E\) is not distinguished from \((j,i) \in E\). Simplicity means
that \((i, i) \notin E\) for all \(i \in V\). Weightedness means that the edge \((i, j) \in E\) has a weight \(w_{ij} = w_{ji} > 0\). Of particular interest are connected graphs.

**Definition 1 (Connected Graph):** A graph \(G\) is connected if, for all \(i, j \in \{1, ..., N\} \), \(i \neq j\), there is a sequence of edges one can traverse from node \(i\) to node \(j\).

This paper uses the weighted graph Laplacian, which is defined with weighted adjacency and weighted degree matrices. The weighted adjacency matrix \(A(G) \in \mathbb{R}^{N \times N}\) of \(G\) is defined element-wise as

\[
A(G)_{ij} = \begin{cases} w_{ij} & (i, j) \in E \\ 0 & \text{otherwise} \end{cases}.
\]

Because we only consider undirected graphs, \(A(G)\) is symmetric. The weighted degree of node \(i \in V\) is defined as \(d_i = \sum_{j(i, j) \in E} w_{ij}\). The maximum degree is \(d_{\text{max}} = \max_i d_i\). The degree matrix \(D(G) \in \mathbb{R}^{N \times N}\) is the diagonal matrix \(D(G) = \text{diag}(d_1, ..., d_N)\). The weighted Laplacian of \(G\) is then defined as \(L(G) = D(G) - A(G)\).

Let \(\lambda_k(\cdot)\) be the \(k^{th}\) smallest eigenvalue of a matrix. By definition, \(\lambda_1(L(G)) = 0\) for all graph Laplacians and

\[
0 = \lambda_1(L(G)) \leq \lambda_2(L(G)) \leq \cdots \leq \lambda_N(L(G)).
\]

The value of \(\lambda_2(L(G))\) plays a key role in this paper.

**Definition 2 (Algebraic Connectivity [17]):** The algebraic connectivity of a graph \(G\) is the second smallest eigenvalue of its Laplacian and \(G\) is connected if and only if \(\lambda_2(L(G)) > 0\).

Node \(i\)'s neighborhood set \(N_i\) is the set of all agents that agent \(i\) communicates with, denoted \(N_i = \{j \mid (i, j) \in E\}\).

### B. Differential Privacy Background

This section provides a brief description of the differential privacy background needed for the remainder of the paper. More complete expositions can be found in [4], [18]. Overall, the goal of differential privacy is to make similar pieces of data appear approximately indistinguishable from one another. Differential privacy is appealing because its privacy guarantees are immune to post-processing [18]. For example, private data can be filtered without threatening its privacy guarantees [4], [19]. More generally, arbitrary post-hoc computations on private data do not harm differential privacy. In addition, after differential privacy is implemented, an adversary with complete knowledge of the mechanism used to implement privacy has no advantage over another adversary without mechanism knowledge [1], [2].

In this paper we use differential privacy to privatize state trajectories of mobile autonomous agents. We consider vector-valued trajectories of the form \(Z = (Z(1), Z(2), ..., Z(k), ...)\), where \(Z(k) \in \mathbb{R}^d\) for all \(k\). The \(\ell_2\) norm of \(Z\) is defined as

\[
\|Z\|_{\ell_2} = \left(\sum_{k=1}^{\infty} \|Z(k)\|_2^2\right)^{\frac{1}{2}},
\]

where \(\|\cdot\|_2\) is the ordinary 2-norm on \(\mathbb{R}^d\).

We consider privacy over the set of trajectories

\[
\hat{\ell}_2^d = \{Z \mid \|Z(k)\|_2 < \infty \text{ for all } k\}.
\]

This set is similar to the ordinary \(\ell_2\)-space, except that the entire trajectory need not have finite \(\ell_2\)-norm. Instead, only each entry of a trajectory must have finite 2-norm in \(\mathbb{R}^d\). Thus, the set \(\hat{\ell}_2^d\) contains trajectories that do not converge, which admits a wide variety of trajectories seen in control systems.

We consider a network of \(N\) agents, where agent \(i\)'s state trajectory is denoted by \(x_i\). The \(k^{th}\) element of agent \(i\)'s state trajectory is \(x_i(k) \in \mathbb{R}^d\) for \(d \in \mathbb{N}\), and agent \(i\)'s state trajectory belongs to \(\hat{\ell}_2^d\).
The goal of differential privacy is to make “similar” pieces of data approximately indistinguishable, and an adjacency relation is used to quantify when pieces of data are “similar.” In this work, we provide privacy to trajectories of single agents. That is, each agent is only concerned with its own privacy and agents will privatize their own state trajectories before they are ever shared. To reflect this setup, our choice of adjacency relation is defined for single agents. This is in contrast to some work that privatizes collections of trajectories at once. The approach we consider, which is sometimes called input perturbation in the literature [4], has also been widely used, and it amounts to privatizing data, then using it in some computation, rather than performing some computation with sensitive data and then privatizing its output.

In this work we used a parameterized adjacency relation with parameter $b_i$.

**Definition 3 (Adjacency):** Fix an adjacency parameter $b_i > 0$ for agent $i$. $\text{Adj}_{b_i} : \tilde{\ell}_2^{d} \times \tilde{\ell}_2^{d} \to \{0, 1\}$ is defined as

$$\text{Adj}_{b_i}(v_i, w_i) = \begin{cases} 1 & \|v_i - w_i\|_{\ell_2} \leq b_i \\
0 & \text{otherwise.} \end{cases} \quad \triangle$$

In words, two state trajectories that agent $i$ could produce are adjacent if and only if the $\ell_2$-norm of their difference is upper bounded by $b_i$. This means that every state trajectory within distance $b_i$ from agent $i$’s actual state trajectory must be made approximately indistinguishable from it to enforce differential privacy.

To calibrate differential privacy’s protections, agent $i$ selects privacy parameters $\epsilon_i$ and $\delta_i$. Typically, $\epsilon_i \in [0.1, \ln 3]$ and $\delta_i \leq 0.05$ for all $i$ [5]. The value of $\delta_i$ can be regarded as the probability that differential privacy fails for agent $i$, while $\epsilon_i$ can be regarded as the information leakage about agent $i$.

This work provides differential privacy for each agent individually using input perturbation, i.e., by adding noise to sensitive data directly. Noise is added by a privacy mechanism, which is a randomized map. We next provide a formal definition of differential privacy. First, fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We consider outputs in $\tilde{\ell}_2^{d}$ and use a $\sigma$-algebra over $\tilde{\ell}_2^{d}$, denoted $\Sigma_d^{\tilde{\ell}_2}$ [20].

**Definition 4 (Differential Privacy):** Let $\epsilon_i > 0$ and $\delta_i \in [0, \frac{1}{2})$ be given. A mechanism $M : \tilde{\ell}_2^{d} \times \Omega \to \tilde{\ell}_2^{d}$ is $(\epsilon_i, \delta_i)$-differentially private if, for all adjacent $x_i, x'_i \in \tilde{\ell}_2^{d}$, we have

$$\mathbb{P}[M(x_i) \in S] \leq e^{\epsilon_i} \mathbb{P}[M(x'_i) \in S] + \delta_i$$

for all $S \in \Sigma_d^{\tilde{\ell}_2}$. \triangle

The Gaussian mechanism will be used to implement differential privacy in this work. The Gaussian mechanism adds zero-mean i.i.d. noise drawn from a Gaussian distribution pointwise in time. Stating the required distribution uses the $Q$-function, defined as $Q(y) = \frac{1}{\sqrt{2\pi}} \int_y^\infty e^{-\frac{z^2}{2}} dz$.

**Lemma 1 (Gaussian Mechanism [4]):** Let $b_i > 0$, $\epsilon_i > 0$, and $\delta_i \in (0, \frac{1}{2})$ be given, fix the adjacency relation $\text{Adj}_{b_i}$, and let $x_i \in \tilde{\ell}_2^{d}$. When sharing the state trajectory $x_i$ itself, the Gaussian mechanism takes the form $\tilde{x}_i(k) = x_i(k) + v_i(k)$. Here $v_i$ is a stochastic process with $v_i(k) \sim \mathcal{N}(0, \Sigma_{v_i})$, where $\Sigma_{v_i} = \sigma_i^2 I_d$ with

$$\sigma_i \geq \frac{b_i}{2\epsilon_i} (K_{\delta_i} + \sqrt{K_{\delta_i}^2 + 2\epsilon_i})$$

and $K_{\delta_i} = Q^{-1}(\delta_i)$. This mechanism provides $(\epsilon_i, \delta_i)$-differential privacy to $x_i$. \[ For convenience, let $\kappa(\delta_i, \epsilon_i) = \frac{1}{2\epsilon_i} (K_{\delta_i} + \sqrt{K_{\delta_i}^2 + 2\epsilon_i})$. We next formally define the problems that are the focus of the rest of the paper. \]

**III. Problem Formulation**

In this section we state and analyze the differentially private formation control problem. We begin with the
problem statement itself, then elaborate on the underlying technical details.

A. Problem Statement

Problem 1: Consider a network of $N$ agents with communication topology modeled by the undirected, simple, connected, and weighted graph $G$. Let $x_i(k) \in \mathbb{R}^d$ be agent $i$’s state at time $k$, $N_i$ be agent $i$’s neighborhood set, $\gamma > 0$ be a stepsize, and $w_{ij}$ be a positive weight on the edge $(i, j) \in E$. Let $x_i(k) \in \mathbb{R}^d$ denote agent $i$’s state at time $k$, and let $n_i(k)$ denote the process noise in agent $i$’s state dynamics at time $k$. We define $\Delta_{ij} \in \mathbb{R}^d$ for all $(i, j) \in E$ as the desired relative state offset between agents $i$ and $j$. Do each of the following:

i. Implement the formation control protocol

$$x_i(k+1) = x_i(k) + \gamma \sum_{j \in N_i} w_{ij} (x_j(k) - x_i(k) - \Delta_{ij}) + n_i(k),$$

in a differentially private, decentralized manner.

ii. Bound the performance of the network in terms of the privacy parameters of each agent and the algebraic connectivity of the underlying communication topology; use those bounds to quantify tradeoffs between privacy, connectedness, and network performance.

iii. Use those tradeoffs to formulate an optimization problem to co-design the communication topology and privacy parameters of the network.

Before solving Problem 1, we give the necessary definitions for formation control. First, we define agent- and network-level dynamics and detail how each agent will enforce differential privacy. Then, we explain how differentially private communications affect the performance of a formation control protocol and how to quantify the quality of a formation.

B. Multi-Agent Formation Control

The goal of formation control is for agents in a network to assemble into some geometric shape or set of relative states. Multi-agent formation control is a well-researched problem and there are several mathematical formulations one can use to achieve similar results [13], [21]–[26]. We will define relative offsets between agents that communicate and the control objective is for all agents to maintain these relative offsets to each of their neighbors. This approach is similar to that of [23].

For the formation to be feasible, we require $\Delta_{ij} = -\Delta_{ji}$ for all $(i, j) \in E$. The network control objective is driving $\lim_{k \to \infty} (x_j(k) - x_i(k)) = \Delta_{ij}$ for all $(i, j) \in E$. The formation can be centered around any point in $\mathbb{R}^d$ and meet this requirement, i.e., we allow formations to be translationally invariant [13].

Now we define the agents’ update law. We model agents as single integrators, i.e.,

$$x_i(k + 1) = x_i(k) + u_i(k) + n_i(k),$$

where $n_i(k) \sim \mathcal{N}(0, \sigma^2 I_d)$ is process noise and $u_i(k) \in \mathbb{R}^d$ is agent $i$’s input.

Let $\{p_1, \ldots, p_N\}$ be any collection of points in formation such that $p_j - p_i = \Delta_{ij}$ for all $(i, j) \in E$ and let $p = (p_1^T, \ldots, p_N^T)^T \in \mathbb{R}^{Nd}$ be the network-level formation specification. We consider the formation control protocol in (1). At the network level, let $x(k) = (x_1(k)^T, \ldots, x_N(k)^T)^T \in \mathbb{R}^{Nd}$, $n(k) = (n_1(k)^T, \ldots, n_N(k)^T)^T \in \mathbb{R}^{Nd}$, and let $\bar{x}(k) = x(k) - p$ with $\bar{x}_i(k) = x_i(k) - p_i$. Then we analyze

$$\bar{x}(k + 1) = (I_N - \gamma L(G) \otimes I_d) \bar{x}(k) + n(k).$$

Let $\bar{x}_{ij}$ be the $l^{th}$ scalar element of $\bar{x}_i$. Then

$$\bar{x}_{i[l]} = [\bar{x}_{1[l]}, \ldots, \bar{x}_{N[l]}]^T$$

is the vector of all agents’ states in the $l^{th}$ dimension,
and
\[ \bar{n}_{[l]} = [n_{1[l]}, \ldots, n_{N[l]}]^T \]
is the vector of corresponding noise terms. The protocol
\[ \bar{x}(k + 1) = ((I_N - \gamma L(G)) \otimes I_d) \bar{x}(k) + n(k) \]
is equivalent to running the protocol
\[ x_{[l]}(k + 1) = (I_N - \gamma L(G))x_{[l]}(k) + n_{[l]}(k) \quad (2) \]
for all \( l \in \{1, \ldots, d\} \) simultaneously.

C. Private Communications for Formations (Solution to Problem 1.ii)

To privately implement the protocol in (1), agent \( j \) starts by selecting privacy parameters \( \epsilon_j > 0 \), \( \delta_j \in (0, \frac{1}{2}) \), and adjacency relation \( Adj_b \) with \( b_j > 0 \). Then, agent \( j \) privatizes its state trajectory \( x_j \) with the Gaussian mechanism. Let \( \tilde{x}_j \) denote the differentially private version of \( x_j \), where, pointwise in time, \( \tilde{x}_j(k) = x_j(k) + v_j(k) \), with \( v_j(k) \sim \mathcal{N}(0, \Sigma_{v_j}) \), where \( \Sigma_{v_j} = \sigma_j^2 I_d \) and \( \sigma_j \geq \kappa(\delta_j, \epsilon_j)b_j \). Lemma 1 shows that this setup keeps agent \( j \)'s state trajectory \((\epsilon_j, \delta_j)\)-differentially private. Agent \( j \) then shares \( \tilde{x}_j(k) = \tilde{x}_j(k) - p_j \) with its neighbors, and this is also \((\epsilon_j, \delta_j)\)-differentially private because subtracting \( p_j \) is merely post-processing \([18]\). This process is shown in Figure 1.

In this privacy implementation, each agent is concerned with privatizing its own trajectory rather than implementing privacy for the network-level trajectory \( x \). That is, each agent privatizes its own information and then shares it with the other agents. In the privacy literature, the protection of one agent’s information is sometimes referred to as local differential privacy \([27]\), and it means that privacy guarantees are provided at the agent level.

We emphasize here that our differential privacy implementation differs from existing works that privatize each state value individually. In particular, we implement the trajectory-level notion of differential privacy used in \([4]\) and \([28]\). This form of differential privacy protects elements of \( \tilde{\gamma}_\Delta \), which are infinite-length trajectories. It does not seek to protect single states in \( \mathbb{R}^d \) as was done in \([29]\) and other works. In those other works, correlations among state values over time cause privacy to weaken. However, the trajectory-level privacy that we use is designed to mask differences between entire infinite-length trajectories that an agent’s dynamics could produce, and it does not weaken over time.

When each agent implements privacy as illustrated above, agent \( i \) only has access to \( \tilde{x}_j(k) \) for \( j \in N_i \). Plugging this into the node-level formation control protocol in (1) gives
\[ \tilde{x}_i(k + 1) = \tilde{x}_i(k) + \gamma \sum_{j \in N_i} w_{ij}(\tilde{x}_j(k) - \tilde{x}_i(k)) + n_i(k) \quad (3) \]
This solves Problem 1.ii.

To solve Problems 1.iii and 1.iii, we will analyze this protocol at the network level. For analysis, let \( x_{[l]} = [x_{1[l]}, \ldots, x_{N[l]}]^T \), \( p_{[l]} = [p_{1[l]}, \ldots, p_{N[l]}]^T \), and \( \bar{x}_{[l]} = x_{[l]} - p_{[l]} \). Also let \( \Sigma_n = \text{diag}(\sigma_1^2, \ldots, \sigma_N^2) \) and \( \Sigma_n = \text{diag}(s_1^2, \ldots, s_N^2) \) We begin by formulating the network-
level dynamics in each coordinate.

**Lemma 2:** Let a network of \(N\) agents communicate over the weighted, simple graph \(G\) with Laplacian \(L(G)\) and adjacency matrix \(A(G)\). Suppose that agent \(j\) uses privacy parameters \(\epsilon_j > 0\) and \(\delta_j \in (0, 1/2)\) and the adjacency parameter \(b_j > 0\). Suppose it uses the Gaussian mechanism to generate private states via \(\tilde{x}_j(k) = x_j(k) + v_j(k)\), where \(v_j(k) \sim \mathcal{N}(0, \sigma_j^2 I_d)\) and \(\sigma_j \geq \kappa(\delta_j, \epsilon_j) b_j\) as in Lemma 1. Then when each agent implements the protocol in (3), the network-level dynamics are

\[
\tilde{x}_l(k+1) = (I_N - \gamma L(G))\tilde{x}_l(k) + z_l(k),
\]

for each \(l \in \{1, \ldots, d\}\), where \(z_l(k) \sim \mathcal{N}(0, \Sigma_z)\), \(\Sigma_z = \gamma^2 A(G)\Sigma_v A(G) + \Sigma_n, \Sigma_v = \text{diag}(\sigma_1^2, \ldots, \sigma_N^2)\) and \(\Sigma_n = \text{diag}(s_1^2, \ldots, s_N^2)\).

**Proof:** See Appendix A.

Then to analyze network-level performance, let \(\beta_l(k) := \frac{1}{N} \tilde{x}_l^T(k)\tilde{x}_l(k)\), which is the state vector the protocol in (2) would converge to with initial state \(x_l(k)\) and without privacy or process noise. Also let

\[
e_l(k) = \tilde{x}_l(k) - \beta_l(k),
\]

which is the offset of the current state from the state the protocol would converge to without noise. We analyze this error term in the next section.

**IV. PERFORMANCE OF DIFFERENTIALLY PRIVATE FORMATION CONTROL**

In this work we use the total mean square error of the network at steady-state, denoted \(e_{ss}\), to quantify performance. Agent \(i\)'s private formation control protocol in (3) is in \(\mathbb{R}^d\), and each agent in the network runs this protocol. This is equivalent to running \(d\) identical copies of (2), which is in \(\mathbb{R}^N\). Thus using (2), we can compute the mean square error in dimension \(l\) and then multiply by \(d\) to compute \(e_{ss}\). The mean-square error in dimension \(l\) is equal to \(\lim_{k \to \infty} \frac{1}{N} \sum_{i=1}^{N} E[e_{l,i}^2(k)]\), where \(e_{l,i}(k)\) is the \(i\)th element of \(e_{l}(k)\). Then we have

\[
e_{ss} := \lim_{k \to \infty} \frac{d}{N} \sum_{i=1}^{N} E\left[ e_{l,i}^2(k) \right].
\]

**A. Connections with the Lyapunov Equation**

The main error bound in this paper uses the fact that we can represent the total error in the system as the trace of a covariance matrix. We define \(\Sigma_{e_l}(k) = E\left[ e_l(k)e_l(k)^T \right]\) and \(\Sigma_{\infty} = \lim_{k \to \infty} \Sigma_{e_l}(k)\). Then we have \(e_{ss} = \frac{d}{N} \text{Tr}(\Sigma_{\infty})\). Now we will analyze the dynamics of \(e_l(k)\) and \(\Sigma_{e_l}(k)\). For a given \(\gamma > 0\) and a given graph \(G\), let \(M = I_N - \gamma L(G) - \frac{1}{N} \mathbb{1}\mathbb{1}^T\). Then we have the following.

**Lemma 3:** Let \(N\) agents communicate over a given weighted, undirected, simple graph \(G\) with Laplacian \(L(G)\) and adjacency matrix \(A(G)\). Suppose for all \(i\) that agent \(i\) implements differential privacy using the Gaussian mechanism in Lemma 1 with privacy parameters \(\epsilon_i > 0\) and \(\delta_i \in (0, \frac{1}{2})\). When agent \(i\) implements the private formation control protocol (3), the network level error \(e_{l}(k)\) in (4) evolves via

\[
e_{l}(k+1) = \mathcal{M}\Sigma_{e_l}(k) + (I_N - \frac{1}{N} \mathbb{1}\mathbb{1}^T)z_l(k),
\]

\(\Sigma_{e_l}(k)\) evolves according to

\[
\Sigma_{e_l}(k+1) = \mathcal{M}\Sigma_{e_l}(k)\mathcal{M} + (I_N - \frac{1}{N} \mathbb{1}\mathbb{1}^T)\Sigma_z(I_N - \frac{1}{N} \mathbb{1}\mathbb{1}^T),
\]
and \( \Sigma_{e[i]}(k) \) can be computed via

\[
\Sigma_{e[i]}(k) = \sum_{i=0}^{k-1} M^i(I_N - \frac{1}{N} I I^T)\Sigma_z(I_N - \frac{1}{N} I I^T)M^i.
\]

**Proof:** See Appendix B.

**Lemma 4:** Let \( N \) agents communicate over a given weighted, undirected, simple, connected graph \( G \) with Laplacian \( L(G) \) and \( M = I_N - \gamma L(G) - \frac{1}{N} I I^T \). Then the eigenvalues of \( M \) are strictly less than 1. Furthermore, the maximum singular value of \( M \) is given by

\[
\sigma_{\text{max}}(M) = 1 - \gamma \lambda_2(L).
\]

**Proof:** See Appendix C.

We now show that the steady-state error covariance matrix, \( \Sigma_\infty \), is a solution to a discrete-time Lyapunov equation.

**Theorem 1:** Let \( N \) agents communicate over a given undirected, simple graph \( G \) with weighted Laplacian \( L(G) \) and adjacency matrix \( A(G) \). Suppose that agent \( i \) implements differential privacy using the Gaussian mechanism with privacy parameters \( \epsilon_i > 0 \) and \( \delta_i \in (0, \frac{1}{2}) \) and implements the private formation control protocol \( \{e_i, \delta_i\} \). If the underlying graph \( G \) is connected, \( \Sigma_\infty \) is equal to the unique solution to the discrete time Lyapunov equation

\[
\Sigma_\infty = Q + M \Sigma_\infty M,
\]

where \( Q = (I_N - \frac{1}{N} I I^T)\Sigma_z(I_N - \frac{1}{N} I I^T) \).

**Proof:** From Lemma 3 we have

\[
\Sigma_\infty = \lim_{k \to \infty} \Sigma_{e[i]}(k) = \sum_{i=0}^{\infty} M^i(I_N - \frac{1}{N} I I^T)\Sigma_z(I_N - \frac{1}{N} I I^T)M^i.
\]

Taking the first term out of the sum gives

\[
\Sigma_\infty = Q + \sum_{i=1}^{\infty} M^i Q M^i.
\]

Factoring out \( M \) on both sides of the sum gives

\[
\Sigma_\infty = Q + M \left( \sum_{i=1}^{\infty} M^i Q M^{i-1} \right) M.
\]

The remaining infinite sum is precisely \( \Sigma_\infty \). Thus, we arrive at the equation \( \Sigma_\infty = Q + M \Sigma_\infty M \), which is the discrete-time Lyapunov equation. From Lemma 4, \( M \) has eigenvalues strictly less than 1 for any undirected, connected graph \( G \) and \( Q \) is positive definite. Using Proposition 2.1 from [30], \( \Sigma_\infty \) has a unique, symmetric solution \( \Sigma_\infty \).

**Theorem 1** along with the fact that \( e_{ss} = \frac{d}{N} \text{Tr}(\Sigma_\infty) \), allows us to solve for \( e_{ss} \). That is, given a communication topology \( G \) and set of privacy parameters \( \{(\epsilon_i, \delta_i)\}_{i=1}^{N} \), we can determine the performance of the network, encoded by \( e_{ss} \), before runtime.

In this work, we are interested in designing a communication topology that allows agents to be as private as possible while meeting performance constraints. These performance constraints will take the form of \( e_{ss} \leq e_R \) where \( e_R \) is the maximum allowable error at steady-state. While Theorem 1 provides a means to evaluate the performance of a given network, it does not help us in designing a network to meet a specified performance requirement of the form \( e_{ss} \leq e_R \). For example, if we are given a communication topology \( G \) and set of privacy parameters \( \{(\epsilon_i, \delta_i)\}_{i=1}^{N} \), we can use Theorem 1 to compute \( e_{ss} \) and can check if \( e_{ss} \leq e_R \), but Theorem 1 does not provide a direct way to design a network that achieves \( e_{ss} \leq e_R \) if the bound is not already met. In the next subsection, we find a scalar bound on \( e_{ss} \) that will be used to solve network design problems of this type in Section V.

**B. Analytical Result and Bounds**

In this section we find a scalar bound on \( e_{ss} \) in terms of the privacy parameters \( \{(\epsilon_i, \delta_i)\}_{i=1}^{N} \), and properties of
the communication topology $\mathcal{G}$ that will allow us to design networks that achieve given performance constraints. In the following theorem, the main bound is a result of $\Sigma_\infty$ being the solution to a Lyapunov equation as shown in Theorem [1]. Several bounds and properties of Lyapunov equations have been explored in the literature, and some of these results have been surveyed in [31]. We have the following bound on $e_{ss}$.

**Theorem 2:** Let all the conditions from Theorem [1] hold. With

$$e_{ss} := \lim_{k \to \infty} \frac{d}{N} \sum_{i=1}^{N} E \left[ \epsilon_{[i],i}(k) \right],$$

we have

$$e_{ss} \leq \frac{\gamma d \sum_{i=1}^{N} \left( \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N} \right) \sigma_i^2 + \frac{N-1}{N} \sum_{i=1}^{N} s_i^2}{N \lambda_2(L) \left( 2 - \gamma \lambda_2(L) \right)}.$$

**Proof:** See Appendix D.

Theorem 2 solves Problem 1(ii). This result gives a scalar bound on $e_{ss}$ that depends on the privacy parameters $\{(\epsilon_i, \delta_i)\}_{i=1}^{N}$ through $\sigma_i$, where $\sigma_i \geq \kappa(\delta_i, \epsilon_i) b_i$, the edge weights in the graph $w_{ij}$, the weighted degree of each agent, the algebraic connectivity $\lambda_2(L)$, and the variance of the process noise $s_i^2$.

In the previous subsection we detailed how Theorem 1 was not sufficient for designing networks to meet performance constraints of the form $e_{ss} \leq \epsilon_R$. Theorem 2 provides a method to enforce these constraints. For example, if the network must achieve $e_{ss} \leq \epsilon_R$ this can be achieved by requiring that

$$\frac{\gamma d \sum_{i=1}^{N} \left( \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N} \right) \sigma_i^2 + \frac{N-1}{N} \sum_{i=1}^{N} s_i^2}{N \lambda_2(L) \left( 2 - \gamma \lambda_2(L) \right)} \leq \epsilon_R.$$

This requirement will impose constraints on the communication topology $\mathcal{G}$, the weights in the communication topology $w_{ij}$, and privacy parameters $\{(\epsilon_i, \delta_i)\}_{i=1}^{N}$. Overall, the bound on $e_{ss}$ in Theorem 2 gives us a method to translate a performance requirement into a joint constraint on the communication topology and privacy parameters. In the next section, we use this constraint to formulate an optimization problem where the privacy parameters and communication topology are the decision variables to design a private formation control network that allows agents to be as private as possible while meeting global performance requirements.

### V. PRIVACY AND NETWORK CO-DESIGN

In this section we solve Problem 1(iii). Given the aforementioned bounds on formation error, we now focus on designing networks for performing private formation control. Our goal is to design a network, through selecting entries of $L(\mathcal{G})$, and privacy scheme, through selecting the values of $(\epsilon_i, \delta_i)_{i=1}^{N}$, that meet global performance requirements, agent-level privacy requirements, and other constraints. Here we keep $\delta_i$ fixed and tune $\epsilon_i$ to achieve the desired level of privacy. Since $\delta_i$ is the probability that differential privacy fails, the existing privacy literature is primarily focused on tuning $\epsilon_i$ and we do so here.

The key tradeoff in designing a private formation control network is balancing agent-level privacy requirements with global performance. For example, if some agents use very strong privacy, then the high-variance privacy noise they use will make global performance poor, even if many other agents have only weak privacy requirements. These effects can also be amplified or attenuated by the communication topology of the network, e.g., if an agent with strong privacy sends very noisy messages to other agents along heavily weighted edges. Privacy/network co-design thus requires unifying and balancing these tradeoffs while designing a weighted, undirected graph and the privacy levels that agents use.
A. Co-Design Search Space

In this section we consider the following setup. We are given $N$ agents that wish to implement private formation control and each agent has a minimum strength of privacy that it will accept. We are tasked with designing a network that allows agents to be as private as possible while meeting a global performance requirement. In many settings, each agent’s neighborhood set will be fixed a priori based on hardware compatibility or physical location, and these neighborhoods specify an unweighted, undirected graph of edges that can be used.

As network designers, we must design the privacy parameters for each agent and all edge weights for the given graph. That is, we can assign a zero or non-zero weight to each edge that is present, but we cannot attempt to assign non-zero weight to an edge that is absent. We denote the given unweighted graph by $G$ and denote its unweighted Laplacian by $L$. We define $L(L_0)$ as the space of all weighted graph Laplacians $L$ such that $L_{ij} = 0$ if $L_{0,ij} = 0$.

Thus, if the original undirected graph has $|E|$ edges, we must select $|E| + N$ parameters, namely one weight for each edge and the values of $\{\epsilon_i\}_{i=1}^N$. Designing these parameters manually for large networks will be infeasible and thus we develop a numerical framework for their design.

B. Co-Design Problem Statement

Below we formally state the privacy/network co-design problem and then describe its features.

**Problem 2 (Co-design Problem):** Given an input undirected, unweighted, simple graph $G_0$ with Laplacian $L_0$, a required global error bound $\epsilon_R$, a minimum connectivity parameter $\lambda_2L$, weighting factor $\vartheta$, and a minimum level of privacy $\epsilon_i^{max}$ for agent $i$ for all $i \in [N]$, to co-design privacy and agents’ communication topology, solve

$$\min_{L(G) \in L(L_0), \{\epsilon_i\}_{i=1}^N} \text{Tr}(L(G)) + \vartheta \sum_{i=1}^N \epsilon_i^2$$

subject to

$$\gamma d \sum_{i=1}^N (\sum_{j=1}^N w_{ij}^2 - \frac{d_i^2}{N}) \sigma_i^2 + \frac{N-1}{N} \sum_{i=1}^N s_i^2 \leq \epsilon_R$$

$$\epsilon_i \leq \epsilon_i^{max} \quad \text{for all } i$$

$$\lambda_2(L) \geq \lambda_2L.$$  

First, we describe the objective function of Problem [2]. In the objective function, the purpose of $\text{Tr}(L(G))$ is to produce a network that uses as little edge weight as possible. In Theorem[2] the numerator of the error bound grows as $w_{ij}$ grows. With $\text{Tr}(L(G)) = \sum_{i \in [N]} d_i$, minimizing this will produce a solution where each agent has a small degree, which promotes better network-level performance. If edge weights represent or are correlated to a monetary cost, minimizing $\text{Tr}(L(G))$ also produces a solution that will use as little cost as possible. For the other term in the objective function, $\sum_{i \in [N]} \epsilon_i^2$, minimizing this function will make each agent’s privacy level as strong as possible; since the strength of privacy grows as $\epsilon_i$ shrinks, this term will force the solution to have a strong level of privacy for each agent. The weighting factor $\vartheta \in \mathbb{R}$ allows the user to prioritize a solution with a strong level of privacy or small degrees for each agent.

We now describe the constraints of Problem [2]. The first constraint requires that the total mean square error of the network, $e_{ss}$, is less than some user-defined value $\epsilon_R$. A sufficient condition for $e_{ss} \leq \epsilon_R$ can be found by requiring that the bound in Theorem 2 is less than $\epsilon_R$, which is what we implement here. The second constraint requires that each agent’s privacy level is at least as strong as that agent’s weakest acceptable privacy level,
which is set by $\epsilon_i^{\text{max}}$. The last constraint requires that the solution produces a connected graph. Connectivity for the weighted graph can be ensured by $\lambda_2(L(G)) > 0$, but we implement the constraint $\lambda_2(L(G)) \geq \lambda_{2L}$ for some user defined $\lambda_{2L} > 0$ so the user has some control over how connected the resulting graph is. For example, $\lambda_{2L}$ can be set to the algebraic connectivity of the line graph on $N$ nodes, which has the least algebraic connectivity among graphs on $N$ nodes.

Overall, solving Problem 2 will produce a communication topology and privacy parameters that will meet global performance and connectivity constraints while allowing agents to be as private as possible. This solves Problem 1.

C. Numerically Solving Problem 2

We have built a MATLAB program to solve Problem 2 which is available on GitHub [32]. Due to the non-linearities in the problem, i.e., some of our constraints are in terms of the second largest eigenvalue of one of our decision variables, we found MATLAB and fmincon to perform well for problems of this kind. When optimizing over $L(G) \in L(L_0)$, since we only consider undirected, symmetrically weighted graphs, we need only find the upper triangular elements of $L(G)$, which helps reduce computation time. Further commentary and examples are provided in the next section.

VI. SIMULATIONS

In this section we provide simulation results for optimal privacy/network co-design. There are four main parameters that we have control over in Problem 2 (i) $e_R$, to control the performance of the system, (ii) $\epsilon_i^{\text{max}}$, to specify the weakest allowable privacy level for agent $i$, (iii) $\lambda_{2L}$, to control the connectivity of the designed network, and (iv) $\vartheta$, which weights optimizing performance versus privacy. In this section, we first define an input topology as the undirected, simple graph $G$ shown in Figure 2. Then, we manually tune the parameters $e_R, \epsilon_i^{\text{max}}, \lambda_{2L}$, and $\vartheta$, and run privacy/network co-design for various sets of parameters to obtain a weighted graph and set of privacy parameters $\{\epsilon_i\}_{i=1}^N$.

Throughout this section the smaller a node is drawn the more private it is, i.e., $\epsilon_i$ gets smaller as node $i$ shrinks, and the thicker an edge is drawn the more weight it has. In simulation, edges with weights satisfying $w_{ij} < 10^{-4}$ are considered deleted. We begin by manually adjusting $e_R$ with all other parameters fixed.

Example 1: (Trading off privacy and performance) Fix the input graph $G$ shown in Figure 2. Fix

$$\epsilon^{\text{max}} = [0.4, 0.9, 0.55, 0.35, 0.8, 0.45, 0.7, 0.5, 0.52, 0.58]^T,$$

$$\gamma = 1/2N, \vartheta = 10, \lambda_{2L} = 0.2, \delta_i = 0.05, \text{ and } b_i = 1$$

for all $i$. Now let $e_R$ take on the values

$$e_R \in \{2, 4, 8, 16, 32, 64\}.$$

For each of these values, privacy/network co-design was
used to design the weighted graphs shown in Figure 3 and the privacy parameters shown in Figure 4.

In Figure 3 as we allow weaker performance, quantified by larger $e_R$, the agents are able to use a stronger level of privacy. This is illustrated by the nodes shrinking as $e_R$ increases from Figure 3a to 3f. Furthermore, privacy/network co-design makes the resulting graph less connected when we allow weaker performance. For example, when comparing Figure 3a to Figure 3i, the output network topology has fewer edges when weaker performance is allowed.

In Figure 4, we can see that as the required level of performance decreases, co-design allows the agents to be more private. This trend persists for all agents in varying magnitude which is influenced by the topology and each $\epsilon_i$. We can also see that $\epsilon_i$ decreases rapidly from $e_R = 2$ to $e_R = 8$, and then decreases slower for $e_R > 8$. This shows that relatively small changes in $e_R$ can lead to large changes in the resulting privacy level, i.e., if we relax performance slightly it is possible to gain a much stronger level of privacy, which occurs when $\epsilon_i$ is small for each agent.

**Example 2: (The Maximum Level of Privacy)** Here we fix all of the parameters other than $\epsilon_i^{\max}$. Specifically, fix $\gamma = 1/2N$, $\lambda_{2L} = 0.2$, $\vartheta = 10$, $e_R = 2$, $\delta_i = 0.05$, and $b_i = 1$ for all $i$. We consider the homogeneous case where each agent has the same required level of privacy, i.e., $\epsilon_i^{\max} = \epsilon_i^{\max}$ for $i \in \{1, \ldots, N\}$. Privacy/network co-design was run for $\epsilon_i^{\max} \in \{0.03, 0.04, 0.05\}$. Figure 5 presents the output communication topologies and Figure 6 shows the output privacy parameters for each agent as designed by privacy/network co-design.

In Figure 5, we can see that as we allow each agent to be less private, co-design actually removes edges. Specifically the edges $(8, 9), (2, 4), \text{ and } (4, 5)$ in Figure 5a when $\epsilon_i^{\max} = 0.03$ are not present in Figure 5e when $\epsilon_i^{\max} = 0.05$. This shows that co-design is trying to use as little edge weight or as few edges as possible to meet the constraints. In others words, when each agent uses less privacy, i.e., larger $\epsilon_i$ and $\epsilon_i^{\max}$ for each agent $i$, co-design designs a network with less communication to achieve the same level of performance.

In Figure 6, we can see that when $\epsilon_i^{\max} = 0.03$, each $\epsilon_i$ is close to $\epsilon_i^{\max}$, i.e., the agents are using the weakest privacy possible. This occurs because, with $e_R = 2$, we require relatively strong performance, which limits the strength of agents’ privacy. As a result, when $\epsilon_i^{\max} = 0.03$ privacy/network co-design finds the optimal network to be one in which each agent uses the weakest privacy possible, i.e., $\epsilon_i$ is small and significantly less than $\epsilon_i^{\max}$ for each $i$. This is because with a stronger level of privacy, the communication topology needs to be more connected, which causes $\text{Tr}(L(G))$ in the objective function of Problem 2 to grow. However, when $\epsilon_i^{\max} \in \{0.04, 0.05\}$ the agents are using stronger privacy than the weakest level they specified, as the resulting $\epsilon_i$’s are not at their constrained maximum.

**Example 3: (The Minimum Level of Connectivity)** Here we fix all of the parameters other than $\lambda_{2L}$. Specifically, fix $\gamma = 1/2N$, $\lambda_{2L} = 0.2$, $\vartheta = 10$, $e_R = 1$, $\delta_i = 0.05$, $b_i = 1$ for all $i$, and $\epsilon_i^{\max} = (0.4, 0.9, 0.55, 0.35, 0.8, 0.45, 0.7, 0.5, 0.52, 0.58)^T$.

Then we solve the privacy/network co-design problem for $\lambda_{2L} \in \{0.1, 0.5, 1\}$. Figure 7 gives the output communication topologies and Figure 8 shows the output privacy parameters for each agent.

In Figure 7, we can see that as we require the output network to be more connected, more edge weight is used and each agent uses weaker privacy, as illustrated by the growing nodes and edges from Figure 7a to Figure 7c.
Fig. 3: The outputs of privacy/network co-design with fixed parameters specified in Example 1 for different values of $e_R \in \{2, 4, 8, 16, 32, 64\}$. The smaller a node is drawn, the more private it is, i.e., its value of $\epsilon_i$ is smaller. The thicker an edge is drawn, the larger the edge weight, i.e., $w_{ij}$ is larger. We can see that when we allow weaker performance, indicated by larger $e_R$, each agent has a stronger level of privacy and the topology uses less weight as illustrated by the nodes shrinking from Figure 3a to Figure 3f. Furthermore, privacy/network co-design makes the resulting graph less connected when we allow weaker performance. For example, when comparing Figure 3a to Figure 3f, the output network topology has fewer edges when weaker performance is allowed.
Fig. 4: The numerical values of privacy parameters as designed by privacy/network co-design with fixed parameters specified in Example 1 and $e_R \in \{2, 16, 64\}$. As the required level of performance decreases, co-design allows the agents to be more private. This trend persists for all agents.

We can also see that co-design is adding weight to certain edges more than others. Specifically, the weight of the edge (1, 7) drastically increases while the weight of the edge (9, 10) does not increase much. Agents 9 and 10 are more private than agents 1 and 7 when $\lambda_{2L} = 1.0$ as illustrated by the size of the nodes in Figure 7. Thus, privacy/network co-design is using smaller edge weights for agents with stronger privacy. This makes intuitive sense since agents with strong privacy inject higher-variance noise into the formation control protocol then agents with weaker privacy, and reducing the weights of edges connected to those agents with weaker noise helps mitigate this impact.

In Figure 8 as $\lambda_{2L}$ is increased, $\epsilon_i$ increases for most agents, i.e., as the network is required to be more connected, the agents use weaker privacy. Here privacy/network co-design adds the necessary edge weight to meet the connectivity constraint, and then weakens privacy to meet the performance constraint. Thus privacy/network co-design is trading off privacy and performance as desired.

Example 4: (Tuning the Objective Function) Here we fix all parameters other than $\vartheta$. Specifically, fix $\gamma = 1/2N$, $\lambda_{2L} = 0.2$, $\lambda_{2L} = 0.05$, $e_R = 1$, $\delta_i = 0.05$, and $b_i = 1$ for all $i$, and $e^{\max} = [0.4, 0.9, 0.55, 0.35, 0.8, 0.45, 0.7, 0.5, 0.52, 0.58]^T$.

Then we solve the co-design problem for $\vartheta \in \{1, 100, 1000\}$. Figure 9 presents the output communication topologies and Figure 10 shows the output privacy parameters for each agent.

In Figure 9 we can see that as we increase $\vartheta$ from $\vartheta = 1$ in Figure 9a to $\vartheta = 1000$ in Figure 9c each of the edge weights increase slightly. Intuitively, as we increase $\vartheta$, we are prioritizing minimizing $\sum_{i=1}^N \epsilon_i^2$ rather than $\text{Tr}(L(G))$. As $\vartheta$ changes, the weights do not change very much, though the privacy levels change drastically as illustrated in Figure 10.

In Figure 10 we can see that as $\vartheta$ is increased, the output privacy parameters are much lower for each agent. When $\vartheta = 1$, the privacy parameters $\epsilon_i$ are near their constrained maxima, $\epsilon_i^{\max}$. When $\vartheta = 1000$, we are prioritizing minimizing $\sum_i \epsilon_i^2$ rather than $\text{Tr}(L(G))$, and thus the output privacy parameters are much smaller. Overall, this shows that increasing $\vartheta$ can be used to allow the agents to achieve a stronger level of privacy.

VII. CONCLUSIONS

In this paper we have studied the problem of differentially private formation control. This work enables agents to collaboratively assemble into formations with bounded steady state error and provides methods for solving for the error covariance matrix at steady-state. This work also develop and solves an optimization problem to design the optimal network and
privacy parameters for differentially private formation control. Future work includes generalizing to other privacy/performance co-design problems and implementation on mobile robots.

REFERENCES

[1] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” Foundations and Trends® in Theoretical Computer Science, vol. 9, no. 3–4, pp. 211–407, 2014.

[2] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in Theory of cryptography conference. Springer, 2006, pp. 265–284.

[3] S. P. Kasiviswanathan and A. Smith, “On the’semantics’ of differential privacy: A bayesian formulation,” Journal of Privacy and Confidentiality, vol. 6, no. 1, 2014.

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

[5] K. Yazdani, A. Jones, K. Leahy, and M. Hale, “Differentially private lq control,” arXiv preprint arXiv:1807.05082, 2018.

[6] M. T. Hale and M. Egerstedt, “Cloud-enabled differentially private multiagent optimization with constraints,” IEEE Transactions on Control of Network Systems, vol. 5, no. 4, pp. 1693–1706, 2017.

[7] J. Le Ny and M. Mohammady, “Differentially private mimo filtering for event streams,” IEEE Transactions on Automatic Control, vol. 63, no. 1, pp. 145–157, 2017.

[8] A. Jones, K. Leahy, and M. Hale, “Towards differential privacy for symbolic systems,” in 2019 American Control Conference (ACC). IEEE, 2019, pp. 372–377.

[9] Z. Huang, S. Mitra, and G. Dullerud, “Differentially private iterative synchronous consensus,” in Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society, ser. WPES ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 81–90. [Online]. Available: https://doi.org/10.1145/2381966.2381978

[10] Y. Wang, Z. Huang, S. Mitra, and G. E. Dullerud, “Differential privacy in linear distributed control systems: Entropy minimizing mechanisms and performance tradeoffs,” IEEE Transactions on Control of Network Systems, vol. 4, no. 1, pp. 118–130, 2017.

[11] Z. Xu, K. Yazdani, M. T. Hale, and U. Topcu, “Differentially private controller synthesis with metric temporal logic specifications,” in 2020 American Control Conference (ACC). IEEE,
Fig. 6: The numerical values of privacy parameters as designed by privacy/network co-design with fixed parameters specified in Example 2 and $\epsilon_{\text{max}}^i \in \{0.03, 0.04, 0.05\}$. When $\epsilon_{\text{max}}^i = 0.03$, the agents are using the weakest privacy possible, which occurs because $\epsilon_R = 2$ requires relatively strong performance, and agents must share more information to facilitate this. However, when $\epsilon_{\text{max}}^i \in \{0.04, 0.05\}$ the agents are using stronger privacy than the weakest level they specified, as the resulting $\epsilon_i$’s are not at their constrained maximum.

[20] B. Hajek, Random processes for engineers. Cambridge university press, 2015.
[21] A. Jadbabaie and A. Olshervsly, “Scaling laws for consensus protocols subject to noise,” 2015.
[22] L. Krick, M. E. Broucke, and B. A. Francis. “Stabilisation of infinitesimally rigid formations of multi-robot networks,” International Journal of control, vol. 82, no. 3, pp. 423–439, 2009.
[23] W. Ren, R. W. Beard, and E. M. Atkins, “Information consensus in multivehicle cooperative control,” IEEE Control systems magazine, vol. 27, no. 2, pp. 71–82, 2007.
[24] W. Ren, “Consensus strategies for cooperative control of vehicle formations,” IET Control Theory & Applications, vol. 1, no. 2, pp. 505–512, 2007.
[25] J. A. Fax and R. M. Murray, “Information flow and cooperative control of vehicle formations,” IEEE transactions on automatic control, vol. 59, no. 9, pp. 1465–1476, 2004.
[26] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” Proceedings of the IEEE, vol. 95, no. 1, pp. 215–233, 2007.
[27] M. Yang, L. Lyu, J. Zhao, T. Zhu, and K.-Y. Lam, “Local differential privacy and its applications: A comprehensive survey,” arXiv preprint arXiv:2008.03686, 2020.
[28] R. Hall, A. Rinaldo, and L. Wasserman, “Differential privacy for functions and functional data,” The Journal of Machine Learning Research, vol. 14, no. 1, pp. 703–727, 2013.
[29] Z. Huang, S. Mitra, and N. Vaidya, “Differentially private distributed optimization,” in Proceedings of the 2015 international conference on distributed computing and networking, 2015, pp. 1–10.
[30] P. M. Gahinet, A. J. Laub, C. S. Kenney, and G. Hewer, “Sensitivity of the stable discrete-time lyapunov equation,” IEEE Transactions on Automatic Control, vol. 35, no. 1, pp. 1209–1217, 1990.
[31] W. H. Kwon, Y. S. Moon, and S. C. Ahn, “Bounds in algebraic riccati and lyapunov equations: a survey and some new results,” International Journal of Control, vol. 64, no. 3, pp. 377–389, 1996.
[32] C. Hawkins, “Differentially-private-formation-control-privacy-network-co-design,” https://github.com/fx0641/Differentially-Private-Formation-Control-Privacy-Network-Co-Design.
[33] T. Mori, N. Fukuma, and M. Kuwahara, “On the discrete lyapunov matrix equation,” IEEE Transactions on Automatic Control, vol. 27, no. 2, pp. 463–464, 1982.
Fig. 7: The outputs of privacy/network co-design with fixed parameters specified in Example 3 for different values of $\lambda_2L \in \{0.1, 0.5, 1\}$. The smaller a node is drawn, the more private it is, i.e., its value of $\epsilon_i$ is smaller. The thicker an edge is drawn, the larger the edge weight, i.e., $w_{ij}$ is larger. As we require the output network to be more connected, more edge weight is used and each agent uses weaker privacy. When $\lambda_2L = 1$, the output $\epsilon_i$ is close to $\epsilon_i^{\text{max}}$ for each $i$. Here privacy/network co-design is adding edge weight to meet the connectivity constraint, and then weakens privacy to meet the performance constraint, thus trading off privacy and performance as desired.

**Fig. 8:** The numerical values of privacy parameters as designed by privacy/network co-design with fixed parameters specified in Example 3 and $\lambda_2L \in \{0.1, 0.5, 1\}$. As the output is required to be more connected, increasing $\lambda_2L$, each agent uses weaker privacy. When $\lambda_2L = 1$, the largest value, the agents are using near their weakest required level of privacy in order to meet the performance requirement.

**APPENDIX**

**A. Proof of Lemma 2**

From (3) we have the node level protocol

$$\bar{x}_i(k+1) = \bar{x}_i(k) + \gamma \sum_{j \in N_i} w_{ij} (\bar{x}_j(k) + v_j(k) - \bar{x}_i(k)) + n_i(k),$$

which we factor as

$$\bar{x}_i(k+1) = \bar{x}_i(k) + \gamma \sum_{j \in N_i} w_{ij} (\bar{x}_j(k) - \bar{x}_i(k)) + \gamma \sum_{j \in N_i} w_{ij} v_j(k) + n_i(k).$$

Let $z_i(k) = \gamma \sum_{j \in N_i} w_{ij} v_j(k) + n_i(k)$ and note that

$$z_i(k) = \gamma ([A(G)]_{row \ i} \otimes I_d) v(k) + n_i(k),$$

where $[A(G)]_{row \ i} \in \mathbb{R}^{1 \times n}$ is the $i^{th}$ row of $A(G)$ and $\otimes$ is the Kronecker product. Next we define the vector $z_{[i]}(k) = [z_{1[i]}(k)^T, \ldots, z_{N[i]}(k)^T]^T \in \mathbb{R}^N$ and we have $z_{[i]}(k) = \gamma A(G)v_{[i]}(k) + n_{[i]}(k)$ at the network level. Since $v_{[i]}(k)$ and $n_{[i]}(k)$ are zero mean, $z_{[i]}(k)$ is
Applying this fact, the linearity of expectation, and the zero mean. The covariance of $z_{[l]}(k)$ is calculated as

$$\Sigma_z = E[z_{[l]}(k)z_{[l]}(k)^T] = E[(\gamma A(G)v_{[l]}(k) + n_{[l]}(k)) (\gamma A(G)v_{[l]}(k) + n_{[l]}(k))]$$

Then, since $v_{[l]}(k)$ and $n_{[l]}(k)$ are statistically independent, $E[v_{[l]}(k)n_{[l]}(k)^T] = E[n_{[l]}(k)v_{[l]}(k)^T] = 0$. Applying this fact, the linearity of expectation, and the symmetry of $A(G)$ gives

$$\Sigma_z = \gamma^2 A(G)E[v_{[l]}(k)v_{[l]}(k)^T] A(G) + E[n_{[l]}(k)n_{[l]}(k)^T]$$

$$= \gamma^2 A(G)\Sigma_v A(G) + \Sigma_n.$$ 

Then $z_{[l]}(k) \sim \mathcal{N}(0, \gamma^2 A(G)\Sigma_v A(G) + \Sigma_n)).$ 

---

**B. Proof of Lemma 3**

First, we expand $e_{[l]}(k+1)$,

$$e_{[l]}(k+1) = (I_N - \gamma L(G) - \frac{1}{N}11^T) \bar{x}_{[l]}(k)$$

Expanding further gives

$$e_{[l]}(k+1) = \left( I_N - \gamma L(G) - \frac{1}{N}11^T \right) x_{[l]}(k) + \left( I_N - \frac{1}{N}11^T \right) \beta_{[l]}(k),$$

where we have used the fact that $L(G)1 = 0$. Let

$$\mathcal{M} = I_N - \gamma L(G) - \frac{1}{N}11^T$$

and note that

$$\mathcal{M} \beta_{[l]}(k) = \left( I_N - \frac{1}{N}11^T \right) \beta_{[l]}(k) = 0.$$
As $\vartheta$ is increased, the output privacy parameters are much lower for each agent. When $\vartheta = 1$, the privacy parameters $\epsilon_i$ are near their constrained maxima, $\epsilon_i^{\max}$. When $\vartheta = 1000$, we prioritize minimizing $\sum_i \epsilon_i^2$ rather than $\text{Tr}(L(G))$, and thus the output privacy parameters are much smaller. This further illustrates the flexibility of privacy/network co-design.

Then we have

$$e_{\vartheta}(k + 1) = \mathcal{M} x_{\vartheta}(k) + \left( I_N - \frac{1}{N} 11^T \right) z_{\vartheta}(k)$$

$$= \mathcal{M} (x_{\vartheta}(k) - \beta_{\vartheta}(k)) + \left( I_N - \frac{1}{N} 11^T \right) z_{\vartheta}(k)$$

$$= \mathcal{M} e_{\vartheta}(k) + \left( I_N - \frac{1}{N} 11^T \right) z_{\vartheta}(k),$$

which proves (3). Plugging this into the definition of $\Sigma e_{\vartheta}(k + 1)$ gives

$$\Sigma e_{\vartheta}(k + 1) = E \left[ e(k + 1) e(k + 1)^T \right]$$

$$= \mathcal{M} E \left[ e_{\vartheta}(k) e_{\vartheta}(k)^T \right] \mathcal{M}$$

$$+ \left( I_N - \frac{1}{N} 11^T \right) \Sigma z \left( I_N - \frac{1}{N} 11^T \right)$$

$$= \mathcal{M} \Sigma e_{\vartheta}(k) \mathcal{M} + \left( I_N - \frac{1}{N} 11^T \right) \Sigma z \left( I_N - \frac{1}{N} 11^T \right),$$

which proves (6). Then (7) follows by applying (6) recursively.

C. Proof of Lemma 4

We begin by analyzing the eigenvalues of $\mathcal{M}$. Note that $\mathcal{M} 1 = 0$, and thus $\mathcal{M}$ has eigenvalue 0 with eigenvector $1$. Now let $(\lambda, v)$ be an eigenpair of $\mathcal{M}$ with $\lambda \neq 0$. We now show that $\lambda \neq 0$ implies that $(\lambda, v)$ is also an eigenpair of $I - \gamma L(G)$. Since $G$ is connected and undirected, $I - \gamma L(G)$ is a doubly stochastic matrix, it has one eigenvalue of modulus 1 which is $\lambda (I - \gamma L(G)) = 1$ with an eigenvector of 1, and the rest of the eigenvalues are positive and lie strictly in the unit disk [26, Lemma 3].

Using $1^T (I_N - \gamma L(G)) = 1^T$, we have

$$1^T v = 1^T (I_N - \gamma L(G)) v.$$  

Adding and subtracting $1^T (\frac{1}{N} 11^T) v$ gives

$$1^T v = 1^T (I_N - \gamma L(G)) v - 1^T (\frac{1}{N} 11^T) v + 1^T (\frac{1}{N} 11^T) v$$

$$= 1^T \mathcal{M} v + 1^T (\frac{1}{N} 11^T) v,$$

then plugging in $\mathcal{M} v = \lambda v$ and $1^T (\frac{1}{N} 11^T) v = 1^T$ gives

$$1^T v = \lambda 1^T v + 1^T$$

$$= (\lambda + 1) 1^T v.$$  

Since $\lambda \neq 0$, we must have $1^T v = 0$ for the above to be true. This implies that $v$ is orthogonal to 1. Furthermore we have that $(\frac{1}{N} 11^T) v = 0$, so

$$\lambda v = \mathcal{M} v = (I_N - \gamma L(G)) - \frac{1}{N} 11^T v$$

$$= (I_N - \gamma L(G)) v.$$

This means that any non-zero eigenvalue of $\mathcal{M}$ is also an eigenvalue of $I - \gamma L(G)$ and the associated eigenvector $v$ is orthogonal to 1.
Furthermore, $L(G)$ and $I_N - \gamma L(G)$ have the same eigenvectors and we have that
\[
\lambda_i(I_N - \gamma L(G)) = 1 - \gamma \lambda_i(L(G)).
\]
Thus, $\lambda_i(M) = 1 - \gamma \lambda_i(L(G))$ for $i \in \{2, \ldots, N\}$. With the sorting $\lambda_2(L(G)) \leq \cdots \leq \lambda_N(L(G))$, we have that $\lambda_{\max}(M) = 1 - \gamma \lambda_2(L(G)) < 1$ This implies that all eigenvalues of $M$ lie strictly in the unit disk.

Now $\sigma_{\max}(M) = (1 - \gamma \lambda_2(L))$ follows from the fact that
\[
[\sigma_{\max}(M)]^2 = \left[(\lambda_{\max}(M))^{1/2}\right]^2
\]
\[
= \lambda_{\max}(M)^2
\]
\[
= \lambda_{\max}(M)^2
\]
\[
= (1 - \gamma \lambda_2(L(G)))^2.
\]

\section*{D. Proof of Theorem 2}
First, with Theorem 1 and [31, Equation 151] [33] we can bound $Tr(\Sigma_{\infty})$ as
\[
Tr(\Sigma_{\infty}) \leq \frac{Tr(Q)}{1 - (\sigma_{\max}(M))^2},
\]
where $\sigma_{\max}(M)$ denotes the maximum singular value of $M$.

We start by expanding $Tr(Q)$ in (10), which gives
\[
Tr(Q) = Tr\left((I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)\Sigma_z(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)\right).
\]
Applying cyclic permutation of the trace gives
\[
Tr(Q) = Tr\left((I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)\Sigma_z\right).
\]
Note that $(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T) = I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T$.
Plugging this and $\Sigma_z = \gamma^2 A(G)\Sigma_v A(G) + \Sigma_n$ into (11)
\[
\lambda_1(L(G)) = 0, \text{ we still have } \lambda_{\max}(M) = 1 - \gamma \lambda_2(L(G)) \text{ since } \lambda_1(L(G)) \text{ has a corresponding eigenvector of } 1 \text{ and the non-zero eigenvalues of } M \text{ must have eigenvectors orthogonal to } 1.
\]

To simplify $\gamma^2 Tr((I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G)\Sigma_v A(G))$, cyclic permutation of the trace gives that
\[
\gamma^2 Tr\left((I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G)\Sigma_v A(G)\right)
\]
\[
= \gamma^2 Tr\left(A(G)(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G)\Sigma_v\right).
\]
Now, $(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G)$ has the form
\[
(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G) = A(G) - \frac{1}{N} \mathbb{1} \mathbb{1}^T A(G),
\]
and note that
\[
\frac{1}{N} \mathbb{1} \mathbb{1}^T A(G) = \frac{1}{N} \begin{bmatrix} d_1 & \cdots & d_N \\ \vdots & \ddots & \vdots \\ d_1 & \cdots & d_N \end{bmatrix}.
\]
Thus
\[
(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G) = \begin{bmatrix} -\frac{1}{N} d_1 & w_{12} - \frac{1}{N} d_2 & \cdots & w_{1N} - \frac{1}{N} d_N \\ w_{21} - \frac{1}{N} d_1 & -\frac{1}{N} d_2 & \cdots & w_{2N} - \frac{1}{N} d_N \\ \vdots & \ddots & \ddots & \vdots \\ w_{N1} - \frac{1}{N} d_1 & \cdots & -\frac{1}{N} d_N \end{bmatrix}.
\]
Now, it follows that the $i^{th}$ diagonal term of $A(G)(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G)$ is
\[
\begin{align*}
\left[ A(G)(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G) \right]_{ii} &= \sum_{j=1}^{N} w_{ij}^2 - \frac{1}{N} d_i \sum_{j=1}^{N} w_{ij} \\
&= \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N},
\end{align*}
\]
and
\[
\begin{align*}
\gamma^2 Tr \left( A(G)(I_N - \frac{1}{N} \mathbb{1} \mathbb{1}^T)A(G) \Sigma_v \right) &= \gamma^2 \sum_{i=1}^{N} \left( \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N} \right) \sigma_i^2. \tag{14}
\end{align*}
\]
Plugging (13) and (14) into (12) gives
\[
Tr(Q) = \gamma^2 \sum_{i=1}^{N} \left( \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N} \right) \sigma_i^2 + \frac{N - 1}{N} \sum_{i=1}^{N} s_i^2,
\]
and plugging this into (10) gives
\[
Tr(\Sigma_{\infty}) \leq \frac{\gamma^2 \sum_{i=1}^{N} \left( \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N} \right) \sigma_i^2 + \frac{N - 1}{N} \sum_{i=1}^{N} s_i^2}{1 - (\sigma_{\text{max}}(\mathcal{M}))^2}.
\]
We now simplify $(\sigma_{\text{max}}(\mathcal{M}))^2$, plugging in (8) from Lemma 4 gives
\[
Tr(\Sigma_{\infty}) \leq \frac{\gamma^2 \sum_{i=1}^{N} \left( \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N} \right) \sigma_i^2 + \frac{N - 1}{N} \sum_{i=1}^{N} s_i^2}{1 - (1 - 2\gamma \lambda_2(L)) \lambda_2(L)^2}.
\]
Expanding the denominator and simplifying gives
\[
Tr(\Sigma_{\infty}) = \frac{\gamma^2 \sum_{i=1}^{N} \left( \sum_{j=1}^{N} w_{ij}^2 - \frac{d_i^2}{N} \right) \sigma_i^2 + \frac{N - 1}{N} \sum_{i=1}^{N} s_i^2}{1 - (1 - 2\gamma \lambda_2(L) + \gamma^2 \lambda_2(L)^2)}.
\]
Then using $e_{ss} = \frac{d}{N} Tr(\Sigma_{\infty})$ we arrive at the expression of interest.