DeepBLOC: A Framework for Securing CPS through Deep Reinforcement Learning on Stochastic Games

Alireza Tahsini Noah Dunstatter Mina Guirguis
Department of Computer Science
Texas State University
Email: tahsini, nfd8, msg@txstate.edu

Chuadhry Mujeeb Ahmed
Singapore University of
Technology and Design
Email: chuadhry@mymail.sutd.edu.sg

***Abstract*—One important aspect in protecting Cyber Physical System (CPS) is ensuring that the proper control and measurement signals are propagated within the control loop. The CPS research community has been developing a large set of check blocks that can be integrated within the control loop to check signals against various types of attacks (e.g., false data injection attacks). Unfortunately, it is not possible to integrate all these "checks" within the control loop as the overhead introduced when checking signals may violate the delay constraints of the control loop. Moreover, these blocks do not completely operate in isolation of each other as dependencies exist among them in terms of their effectiveness against detecting a subset of attacks. Thus, it becomes a challenging and complex problem to assign the proper checks, especially with the presence of a rational adversary who can observe the check blocks assigned and optimizes her own attack strategies accordingly. This paper tackles the inherent state-action space explosion that arises in securing CPS through developing DeepBLOC (DB) – a framework in which Deep Reinforcement Learning algorithms are utilized to provide optimal/sub-optimal assignments of check blocks to signals. The framework models stochastic games between the adversary and the CPS defender and derives mixed strategies for assigning check blocks to ensure the integrity of the propagated signals while abiding to the real-time constraints dictated by the control loop. Through extensive simulation experiments and a real implementation on a water purification system, we show that DB achieves assignment strategies that outperform other strategies and heuristics.**

I. Introduction

Major transformations are underway in a wide variety of domains owing to the integration of Cyber-Physical Systems (CPS) as critical components in their operations. From transportation to health-care and from the power-grid to manufacturing systems, CPS are becoming a necessity for efficient, intelligent, autonomous and adaptive operation. Such a wide range of adoption has also made these systems subject to new vectors of cyber-attacks whose impact would cross from the cyber realm into the physical one.

Cyber-attacks on such infrastructure do not just come with a high price tag due to the economical loss of data/equipment/operation (e.g., due to DoS or ransomware), but can be sophisticated enough to slowly drive the system into unsafe operational stages putting human life in danger. The list of such incidents keeps growing (e.g., [22], [31]) suggesting that we will soon encounter more instances of these attacks that cause autonomous vehicles to veer off the road, manipulate devices responsible for power generation and consumption and exploit robotics/drone systems for malicious and terrorism-related activities.

One class of cyber-attacks that has received a lot of attention aims to target the control and/or measurement signals that are propagated over the network in a CPS. Since the proper operation of CPS relies on integral and timely transmission of signals, it has been shown that delaying/dropping signals can push the system outside its stability margins and injecting false data can steer the system into catastrophic states (e.g., [21], [25]). These attacks have prompted a large body of research focusing on developing resilient defense through the use of stateless and state-full checks to detect attacks (e.g., [28]). These checks range from the simple threshold and CUSUM checks to more sophisticated ones that rely on state estimation, model predictors and machine learning to detect anomalies (e.g., [26]).

Problem statement: In this paper, we consider the problem of strategic assignment of checks in the control loop to detect attacks through a deep reinforcement learning approach based on a game-theoretic formulation. We consider a two-player zero-sum stochastic game model between the adversary and the defender. The adversary seeks to choose different attack methods to mount on signals while attempting to minimize their risk of detection. The defender, on the other hand, seeks a non-stagnant policy to assign stateless and stateful checks to signals to detect attacks. This involves capturing the effectiveness of every check, the warm-up periods for checks to be effective, the importance of every signal, and the delay constraints on the control loop. Every sub-game is represented by a state that captures the evolution of the assigned checks and dictates the players' available actions.

Illustrative example: Consider a stage in a multi-stage water purification system (as an example of a CPS) with 5 different types of control and measurement signals (e.g., inlet/outlet pumps and valves, and level sensors). Assume the attacker can mount different types of attacks on these signals and that the defender has 6 checks – with varying effectiveness and warm-up periods – to assign to signals. For each signal, there are 2^6 possible assignments ranging from not assigning any check to assigning all of them. With 5 types of signals, the state space of $(2^6)^5$ exceeds 1 billion. Moreover, the action spaces of the players grow exponentially with more checks and attack methods. Given the varying effectiveness of checks

978-1-7281-4760-4/20/$31.00 ©2020 IEEE

against attacks, their warm-up periods, the importance of every signal, and the delay constraints, it becomes quite complex to obtain potent assignments of checks to signals to detect attacks for practical CPS.

While Dynamic Programming-based approaches can be used to obtain optimal policies through iterating over the solutions of sub-games [16], it becomes computationally prohibitive to track the players' actions and solve each sub-game in every state when the state/action spaces explode. To this end, we develop a deep reinforcement learning approach – which we term DeepBLOC (DB) – to tackle the high dimensionality of our problem. DB provides an approximation to the quality of the players' actions in every state using a deep network. Furthermore, DB approximates Nash equilibrium in every sub-game through the use of fictitious play [4]. A unique aspect of DB stems from the asymmetry between the players (i.e., their actions spaces are different) and unlike traditional approaches that require separate (or alternate) training for each player, in DB the players are trained simultaneously. We demonstrate that our approach yields potent strategies that outperform other heuristics typically used in large-scale systems and resemble the optimal strategies in small-scale problems.

Contributions: We summarize our contributions below:

- We extend the game-theoretic framework presented in [8] to account for a more comprehensive adversary model with budget constraints as well as explicitly forcing the defender to switch his assignment through a stagnancy metric.
- We develop the DeepBLOC (DB) framework that obtains optimal/sub-optimal policies on stochastic games for the defender and the adversary. DB is capable of tackling environments with exponentially large state and action spaces in the presence of asymmetric players.
- We craft a set of features to provide effective approximation methods for solving a high-dimensional problem that is otherwise computationally prohibitive to solve.
- We assess the effectiveness of our defense through extensive simulations and compare our sub-optimal results to optimal ones whenever feasible. We show that our policies outperform other strategies (e.g., random, greedy) by 13%-30%.
- We provide a real-experimental evaluation on a water-treatment test-bed.

Paper organization: In Sec. II we discuss related works. We present a stochastic game model for protecting CPS in Sec. III. We present our DeepBLOC (DB) framework to solve the game model in Sec. IV. We evaluate the performance of DB in Sec. V and conclude the paper in Sec. VI.

II. Related Work

This work is related to three major areas of research:

1) Secure control in networked control systems: A large body of research has investigated the impact of cyber attacks on the measurement and control signals on the stability of Networked Control Systems (NCS) (e.g., [23]). The works in [25], [30], [23] consider different attack models that result in signal loss and/or delay (e.g., Bernoulli packet dropping process, TCP/UDP packet drops, etc.).

The effects of false data injection attacks on CPS have been investigated in various studies (e.g., [21], [17], [27]). All of these studies show how well-crafted attacks can completely cripple the operation of the system. To protect against cyber attacks, the works in [18], [9], [2] develop various methods for maintaining optimal control under cyber attacks and ensuring resiliency through generating fresh signals.

2) Game-theoretic threat screening models: Game theory has been utilized in a wide range of studies to analyze the security of CPS and NCS [6], [33]. Researchers model and analyze security issues as Security Games in which the attacker and the defender have conflicting objectives. Some models consider the attacks on measurement signals (e.g., [14]), control signals (e.g., [7]), or both (e.g., [32]). Another class of security games is those that screen for threats in various applications (e.g., [5], [24]). Their models, however, rely on one-shot games under stateless assumptions. When deriving strategies over time horizons, self-play can be used [11], [12], [13]. In self-play agents are trained one at a time, in the way a player approximates the best response on a Markov Decision Process (MDP), which is made by the other players' strategy profile. Interchangeably, the problem can be framed in the context of multi-agent reinforcement learning (MARL) on a Markov game as suggested in [15]. It has been shown in [16] that such systems can be proven to converge to Nash values under certain settings and provide a reliable way to derive game-theoretic and time-dependent policies.

3) Assigning stateless and stateful checks: Various defense mechanisms have been proposed to detect/mitigate cyber attacks in CPS (e.g., [17], [27], [26], [29], [3]). For example, in [26], the authors construct a safety envelope from the measurements obtained under the normal operation of the system. Attack detectors are then constructed that compare the measurements received to the ones maintained by the safety envelope. The authors in [29] assume knowledge of the state of the system and derive correlation graphs without attacks to study how that information impacts decisions. In [3] the authors prevent an adversary from finding attack vectors by identifying two sets: a set of sensors to protect and a set of state variables that can be independently verified. The work in [28] and [8] investigate the combination and configuration of various defense mechanisms using stateless and stateful checks. The work in [28] does not consider a game-theoretic approach while our previous work in [8] considers a game-theoretic approach but is only applicable to small scale problems and does not use deep reinforcement learning mechanisms. Physics-based models have also been proposed to ensure that the signals in the control loop reflect a legitimate behavior of the system (e.g., [10], [29]). These models include state predictors, estimators and threshold checks which can be integrated in our framework as checks.

To summarize, this work adopts a game-theoretic approach to derive intelligent strategies that integrate the developed checks and defenses to detect attacks coherently. The framework tackles the large state and action spaces through the use of deep reinforcement learning methodologies.

III. Stochastic Game Model

We consider a general Cyber-Physical System (CPS) composed of the plant and controller (distributed or centralized) that communicate over the network. Sensors attached to the plant generate a set of measurements that are transmitted to the controller. Based on these measurements and the goal of the controller, a set of control signals are transmitted to the actuators that change the behavior of the plant. We refer to the measurement and/or control signals as Targets, denoted by the set G. Figure 1 shows a block diagram of a general CPS.

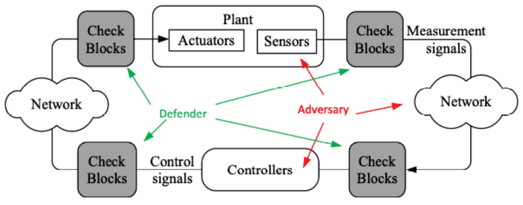

Fig. 1: Block diagram for a CPS with check blocks.

Adversary Model: We assume the adversary has a set of attack methods M (e.g., false-data injection, jamming, delay) available to mount on different targets. The attacks can be mounted directly on the network as in man-in-the-middle attacks, or through malware that compromises the sensors and actuators and/or through manipulating the physical environment. These attacks would result in different target signals than the intended ones. We let U_g denote the utility the adversary gains after a successful attack on target g, and thus U_g reflects the influence of that target signal. Furthermore, we assume the adversary can probe the system to know which checks are in place, but the checks themselves are implemented on a separate system so the adversary cannot manipulate them.

The defender has at his disposal a set of check blocks B that can check the target signals within the control loop. These checks are implemented as bump-in-the-wire (BITW) intercepting signals and checking them on a separate system along four locations as depicted in Figure 1. We let E_b^m denote the effectiveness (i.e., probability) of block b in detecting attack m. Due to the real-time constraints, it is not possible for the defender to assign all of the check blocks on all of the targets, and thus we let C_g denote the capacity in terms of the number of blocks that can be assigned on target g. We assume check blocks could be stateless or stateful (i.e., they need to observe the signal over time to detect attacks). We let W_b denote the warm-up period for block b to be fully effective. Stateless blocks have a W_b of 1. Also, we assume that a check block that is warming-up has an effectiveness that is linearly proportional to the time-steps spent warming-up.

We consider a game-theoretic formulation of a 2-player stochastic game between the defender and the adversary over an infinite horizon in which a zero-sum sub-game is played at every time-step. This stochastic game is represented by the tuple $\langle \mathcal{S}, \mathcal{A}_a, \mathcal{A}_d, \mathcal{T}, \mathcal{R}, \beta \rangle$ where:

- $\mathcal{S}$ is the finite set of system states, where each $s \in \mathcal{S}$ describes the following:
  - **Block assignment N:** represents the current assignment matrix of blocks on targets. Each entry, $n_{b,g}$, denotes the number of times steps still needed by b to be fully effective on g. An entry of 0 indicates that the block is operating at its maximum effectiveness.
  - **Attacker budget ψ:** is an integer bounded by Ψ that represents the resources available to the attacker in mounting attacks whereby each target signal attacked incurs a cost. This conceptually represents the risk the attacker is willing to take.
- $\mathcal{A}_a$ is a finite set of actions the attacker can choose from in order to attack the system. An action $a \in \mathcal{A}_a$ contains an attack method m and a subset of the targets, which we denote by the vector v of length G that specifies for every target whether it is attacked (1) or not (0).
- $\mathcal{A}_d$ is a finite set of actions the defender can choose from. An action $d \in \mathcal{A}_d$ includes adding or removing one block, or making no changes to the block assignment, subject to the *capacity constraints*.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A}_a \times \mathcal{A}_d \times \Rightarrow \Pi(\mathcal{S})$ is the state evolution function based on the players action pair, where Π is a discrete probability distribution over $\mathcal{S}$. Accordingly, we let $\mathcal{T}(s, a, d, s')$ denote the probability of transitioning into state s' from state s when the attacker and the defender take actions $a \in \mathcal{A}_a$ and $d \in \mathcal{A}_d$, respectively. Based on the adversary's action a, her budget ψ is either decremented based on the number of attacked targets, or incremented by a unit value ψ_0, under no attack. Similarly, based on the defender's action d, deploying a new block requires setting the corresponding element in N to W_b, whereas removing a block deactivates a block. Partially active blocks would have the current warm-up period decremented by 1. We also account for environmental uncertainty to capture infrequent transient instabilities on the physical part of the CPS (e.g., upgrades and failures that make the assigned check blocks impractical). Thus, with a small probability λ, the assignment matrix is altered randomly by assigning fresh blocks and unassigning effective ones.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A}_a \times \mathcal{A}_d \times \mathcal{S} \Rightarrow \mathbb{R}$ is the reward obtained by the defender. We let $\mathcal{R}(s_t, a_t, d_t, s_{t+1})$ denote the reward received when going from s_t to s_{t+1} under actions $a \in \mathcal{A}_a$ and $d \in \mathcal{A}_d$ as shown in Equation 1. The first term captures the loss due to a successful attack on the intended targets and the second term reflects the loss due to a stagnant assignment, where γ is a constant weight to punish the defender based on the level of stagnancy. Stagnancy captures the degree unto which the assignment

remains the same between two time-steps. A stagnant assignment is more predictable and thus, can be exploited.

$$\mathcal{R}(s_t, a_t, d_t, s_{t+1}) = -\sum_{g\in v} x_{m,g}U_g - \gamma p(s_t, s_{t+1}) \quad (1)$$

$$x_{m,g} = \prod_{b\in B}(1 - E_b^m \times (1 - \frac{n_{b,g}}{W_b})) \quad (2)$$

$$p(s_t, s_{t+1}) = \sum_{b\in B}\sum_{g\in G} \mathbb{1}_{n_{b,t}=0 \ in \ s_t \ and \ s_{t+1}} \quad (3)$$

Equation 2 gives the probability of a successful attack in which *all* the blocks assigned fail to detect attack m based on the effectiveness E_b^m (hence, the product). Notice that the effectiveness are proportional to the number of remaining time-steps before becoming fully effective. Equation 3 gives the stagnancy p denoting the number of blocks that remained fully active between two time-steps where $\mathbb{1}_{cond}$ is an indicator function when condition $cond$ is true. Due to the zero-sum nature of the game, one reward function suffices for both agents.

- β : is the discount factor such that $0 < \beta < 1$.

In every time-step, a sub-game is generated based on the current state and the available actions of the players. The following Linear Program (LP) optimization problem aims to maximize the defender's worst-case payoff, z, over the probability distribution $\pi(s,d)$ of the defender's actions in state s:

$$\max_{\pi(s,d)\in\Pi(\mathcal{A}_d)} z \quad (4)$$

$$z \leq \sum_{d\in\mathcal{A}_d} \pi(s,d)Q(s,a,d), \quad \forall a \in \mathcal{A}_a \quad (5)$$

$$\sum_{d\in\mathcal{A}_d(s)} \pi(s,d) = 1 \quad (6)$$

$$0 \leq \pi(s,d) \leq 1 \qquad \forall d \in \mathcal{A}_d(s) \quad (7)$$

Constraint 5 follows from the definition of Nash equilibrium where the defender is constrained by the attacker's best response. Since the actions of the players in this sub-game determine not just the reward received, but also expected future rewards based on subsequent states, $Q(s,a,d)$ gives the expected quality of the action pair a, d in s (and will be approximated in the next Section). Constraints 6 and 7 ensure a valid probability distribution over the defender's actions.

The complexity of this LP grows exponentially with the players' action spaces. Moreover, the large state space makes it prohibitive to obtain optimal solutions with dynamic programming. Next, we present our approximation framework based on deep reinforcement learning incorporating Nash equilibrium.

IV. Methodology

In this section we present DeepBLOC (DB) for deriving optimal/sub-optimal policies on multi-agent zero-sum stochastic games. We also present an approximation method for solving every sub-game in DB.

A. DeepBLOC

Deep Q-Network (DQN), is a reinforcement learning approach that has been successfully applied in environments with huge state spaces (e.g., Atari games) [19], [20]. As a variant of Q-learning, DQN can estimate Q-values using a deep neural network over a small number of observations. This method utilizes Bellman's equation to iteratively update the Q-values based on the *best* action to take in every state. In single player environments, the *best* action is the one that maximizes the current reward together with expected future ones. In multi-agent environments, however, the *best* actions have a more complicated meaning as the reward of one agent depends on the actions chosen by the other ones. Therefore, depending on the nature of the interactions, the typical max operator would need to be replaced with more comprehensive ones.

Algorithm 1 DeepBLOC with Experience Replay

```
1: Initialize $i = 0$
2: Initialize learning network with random weights $\theta_i$
3: Initialize target network weights $\theta^- = \theta_i$
4: Initialize $\tau$ to desired update cycle
5: Initialize replay memory $\mathcal{Z}$ to capacity $N$
6: for episode $= 1, E$ do
7:     Randomize starting state $s_1 \in S$
8:     for $t = 1, T$ do
9:         With probability $\epsilon$ take random action pair $\langle a_t, d_t\rangle$
10:          otherwise
11:            $\mathcal{Q}_{s_t}$ = GetQMatrix($s_t$, $\theta_i$)
12:            $\langle\pi_A,\ \pi_D\rangle$ = Nash($\mathcal{Q}_{s_t}$)
13:            Sample $a_t \sim \pi_A$ and $d_t \sim \pi_D$
14:         Execute action pair $\langle a_t, d_t\rangle$ in and observe reward
              $r_t$ and next state $s_{t+1}$
15:         Store transition $(s_t, a_t, d_t, r_t, s_{t+1})$ in $\mathcal{Z}$
16:         Sample random mini-batch of transitions
              $(s_j, a_j, d_j, r_j, s_{j+1})$ from $\mathcal{Z}$
17:         $\mathcal{Q}_{s_{j+1}}$ = GetQMatrix($s_{j+1}$, $\theta^-$)
18:         $\langle\pi_A,\ \pi_D\rangle$ = Nash($\mathcal{Q}_{s_{j+1}}$)
19:         Set $y_j = r_j + \beta(\pi_D \times \mathcal{Q}_{s_{j+1}} \times \pi_A^{'})$
20:         Take gradient descent step on
              $\left(y_j - Q(s_j, a_j, d_j; \theta_i)\right)^2$
21:         Set $i = i + 1$
22:         if $i$ mod $\tau = 0$, then
23:             $\theta^- = \theta_i$
24:         end if
25:     end for
26: end for
```

DB (presented in Algorithm 1) extends DQN for multi-agent environments with a zero-sum reward function. We let $Q(\phi(s,a,d);\theta)$ denote the approximated quality of the action pair a and d in state s based on the neural network with weights θ. We rely on a set of features $\phi(s,a,d)$ that is a function of the state and the action pair to present the input to the network. Due to the non-cooperative nature of our setup between the attacker and defender, we use the $maximin$ function to compute $Q(\phi(s,a,d);\theta)$.

The algorithm uses an experience replay memory $\mathcal{Z}$ to store single transitions (i.e., current state, players' actions, reward, and next state) that is initialized with random transitions. The algorithm stores transitions at every time step and randomly samples a mini-batch from this memory to learn from a diverse set of past experiences without being biased to recent transitions that may exhibit high data correlation. This has been shown to produce highly effective neural networks. The algorithm also uses two identical neural networks – a learning and a target network. The target network is a more stable version of the training network that gets updated after τ steps with the weights of the training network. Both networks are typically initialized with the same random weights.

The algorithm proceeds by running E episodes starting from random states, where each episode is T time steps long. In every state, a sub-game matrix Q_s gets generated according to the actions available to players. Each element in the game matrix is the estimation of the training network about the quality of the corresponding action pair, $Q(\phi(s,a,d);\theta)$. Then, we solve the game matrix using $Nash$ to compute Nash equilibrium and the associated probability distribution over the players' actions, π_A and π_D. By considering an exploration rate ϵ players sample π_A and π_D to observe the next state and the corresponding reward, which will be stored in the replay memory in the form of $< s_t, a_t, d_t, r_t, s_{t+1} >$.

In the next phase, we randomly pick a mini-batch from the replay memory in order to compute the loss (Equation 8) between the prediction of the training network and the target y that we are moving our approximation toward.

$$L_i(\theta_i) = \mathbb{E}_{s_t,a_t,d_t,s_{t+1}\sim z}\Big[\big(y - Q(\phi(s_t,a_t,d_t);\theta_i)\big)^2\Big], \quad (8)$$

where y is given by:

$$\begin{aligned} y = \mathbb{E}_{s_t+1}\big[&\mathcal{R}(s_t,a_t,d_t,s_{t+1}) \\ &+ \beta \max_{d_{t+1}\in A_D} \min_{a_{t+1}\in A_a} Q(\phi(s_{t+1},a_{t+1},d_{t+1});\theta^-)|s_t,a_t,d_t\big]. \end{aligned} \quad (9)$$

Then, we differentiate the loss function regarding the training network parameters to infer the gradient as follows:

$$\begin{aligned} \nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s_t,a_t,d_t,s_{t+1}}\Big[&\nabla_{\theta_i} Q(\phi(s_t,a_t,d_t);\theta_i) \\ \cdot \Big(&\mathcal{R}(s_t,a_t,d_t,s_{t+1}) \\ &+ \beta \max_{d_{t+1}\in A_D} \min_{a_{t+1}\in A_A} Q(\phi(s_{t+1},a_{t+1},d_{t+1});\theta^-) \\ &- Q(\phi(s_t,a_t,d_t);\theta_i)\Big)\Big]. \end{aligned} \quad (10)$$

Finally, after τ number of updates on the training network, we clone the training network into the target network.

B. Fictitious Play

In every time-step, a sub-game is generated based on the current state and the actions available to the players. This sub-game is solved optimally via the LP outlined in Equation 4, which yields Nash equilibrium mixed strategies for both players. Due to the large action spaces in our model, it is computationally prohibitive to solve this LP at every time-step. Thus, we approximate the $Nash$ function in the DB using fictitious play [4] as presented in Algorithm 2.

Algorithm 2 Iterative Fictitious Play

```
1: $R$ is an $m \times n$ matrix of rewards
2: $rowRew$ and $rowCnt$ are $m$-length arrays of zeros
3: $colRew$ and $colCnt$ are $n$-length arrays of zeros
4: Initialize $bestResponse$ to any random row action
5: for $i = 1, Iteratoins$ do
6:     $colRew = colRew + R[bestResponse, :]$
7:     $bestResponse = \text{argmin}(colRew)$
8:     $colCnt[bestResponse]++$
9:     $rowRew = rowRew + R[:, bestResponse]$
10:    $bestResponse = \text{argmax}(rowRew)$
11:    $rowCnt[bestResponse]++$
12: end for
13: $gameValue = \big(\max(rowRew) + \min(colRew)\big)/(2\times i)$
14: $rowMixedStrat = rowCnt / i$
15: $colMixedStrat = colCnt / i$
```

Fictitious play is an iterative approach for approximating Nash equilibrium in normal form games. In this algorithm, players repeatedly play the game in an iterative fashion while tracking the historical behavior of their opponents. In each iteration, a player chooses the best response based on the historical utility their opponent has accrued. This historical utility is represented by *rowRew* and *colRew*, with a best response being defined by a max or min over these vectors, respectively. By tracking each player's action counts (vectors *rowCnt* and *colCnt*), we can obtain mixed strategies for both players with the game value being represented by the average of both player's best response utilities. The accuracy of the algorithm in approximating Nash equilibrium can be tuned by adjusting the number of iterations.[1]

V. Performance Evaluation

In this section, we evaluate the performance of our DB check block assignment method in comparison to other common heuristics through extensive simulation and a realistic setup of a water purification system.

A. Environments, Features and Hyperparameters

Environments: We assessed the performance of DB on different environments with different sizes to show its applicability across them. Env-Small is a moderately simple environment, on which we could calculate the optimal policies using dynamic programming (DP) and demonstrate that the policies derived from DB are almost identical to the optimal ones. Env-Medium and Env-Large are larger environments in which

[1]This sub-optimality can be regarded as a degree of exploration in the same manner as ϵ-greedy approaches aim to explore the environment by choosing random actions (line 9 in Algorithm 1).

obtaining optimal policies is computationally prohibitive. In Env-Medium, we experimented with hand-crafted effectiveness in which some blocks are more effective against certain attack methods than others (rather than being random across all of them). This enabled us to track and interpret the defender's actions. Env-Medium also resembles our experimental setup of the water purification system. Env-Large is a massive environment with 3×10^{89} state space.

Name	Env-Small	Env-Medium	Env-Large
$\|G\|$	3	5	8
$\|B\|$	3	6	10
$\|M\|$	3	6	10
$C_{G\times 1}$	$[2,2,1]$	$[4,3,6,2,5]$	$[4,8,6,4,8,5,9,7]$
$U_{G\times 1}$	$[25,15,40]$	$[35,25,40,30,32]$	$[36,16,47,10,20,30,15,8]$
$W_{B\times 1}$	$[1,2,4]$	$[1,12,6,2,10,5]$	$[2,12,1,12,1,6,14,13,10,4]$
$E_{B\times M}$	Random$(0,1)$	Hand Crafted	Random$(0,0.4)$
Ψ	9	15	48
$\|S\|$	$\sim 3E+5$	$\sim 7E+18$	$\sim 3E+89$
λ	0	0.1	0.1

TABLE I: Parameters used in our Evaluation environments.

Features: We orchestrated a set of features, indicated by $\phi(s,a,d)$, based on the state s, and the action pair a,d as an input to the neural network. Based on the state, the assignment N is represented such that each entry describes the extent to which an assigned check block is effective (i.e., $1-\frac{n_{t,b}}{W_b}$). The attacker's budget ψ is used directly as a feature. Based on the defender's action, we designed features that capture the type of action (e.g., assignment, removal or no change) along with the utility of the affected target signal and a vector showing the effectiveness of the chosen block b against all attack methods (i.e., E_b). Based on the attacker's action, we used features capturing the chosen attack method along with the attacked targets vector v that is element-wise multiplied by the corresponding targets utilities.

Hyperparameters: One of the main challenges was identifying a *consistent* neural network architecture and hyperparameters that are effective across different environment of various sizes. Due to the complexity of DB, a pure trial-and-error process in identifying quality features and proper hyperparameters is not feasible. Therefore, we designed an initial supervised learning method to bootstrap the approximation of the reward function. We generate a data-set by letting the players interact with the environment according to a random policy, where the starting state of each episode was biased to capture special transitions (e.g., significant number of blocks are assigned and/or the attacker has enough budgets to attack). In every transition we store the state and action pair taken in the feature format $\phi(s_i,a_i,d_i)$ in set Φ and the observed reward r_i in set ρ. Then, we trained a neural network, $n:\Phi\rightarrow\rho$ to predict the associate reward with a transition $n(\phi(s_i,a_i,d_i)) \simeq \mathcal{R}(s_i,a_i,d_i,s_{i+1})$, which is very similar to what happens in DB before the first target network update.

We fine-tuned the architecture with a trial-and-error process and decided on a fully connected neural network with 6 layers with the following number of neurons in each layer: $[30,30,20,20,60,1]$. We chose the activation function, loss function, optimizer, batch size and learning factor to be rectified linear unit (ReLU), mean square error, Adam, 32 and 0.005, respectively. This architecture outperformed other ones consistently across all of our experiments and environments.

To find proper values for the hyperparameters in DB, we run many instances of this expensive algorithm. Eventually, we set the size of the replay buffer to 3200, ψ_0 to 1 and the discount factor β to 0.99. We also found that fixing the number of iterations in the fictitious play algorithm to 500 gave us a good balance between accuracy and running time. In particular, we were able to get a speed-up of 7 to 20 times with an accuracy of 94% and above compared to LP.

B. Training and Convergence

Based on our selection of the features and hyperparameters, we trained the attacker and defender on the 3 described environments. It was only feasible to obtain optimal strategies on Env-Small through the use of Dynamic Programming (DP) [16]. Figure 2 shows the convergence of the Q-values in DP while Figure 3 shows the convergence of the loss function of DB through the training process where the periodic spikes indicate the updates to the target network.

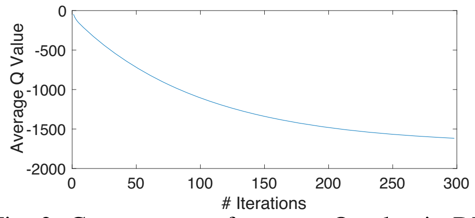

Fig. 2: Convergence of average Q value in DP.

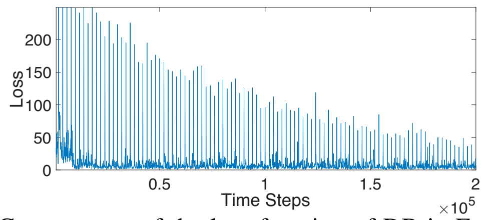

Fig. 3: Convergence of the loss function of DB in Env-Small.

C. Policy Evaluation Results

After training the players, we assess their policies by simulating their interaction with players that follow random and greedy strategies. A random strategy picks actions at random from the set of feasible actions (i.e., subject to the capacity constraints for the defender and budget constraints for the adversary). We introduce two greedy defenders; $G1$ chooses the target with the highest utility and least protection and mounts the block with the highest average effectiveness

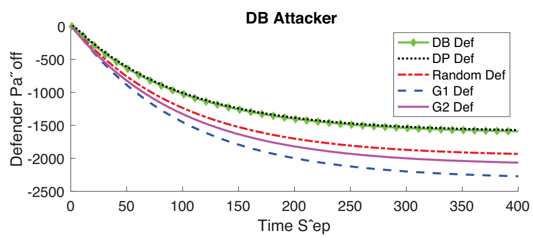

Fig. 4: Defender polices' payoff vs. DB attacker in Env-Small.

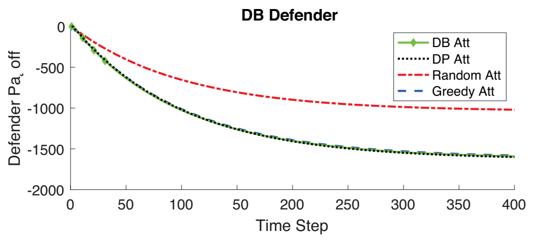

Fig. 5: DB defender's Payoff vs. attacker policies in Env-Small.

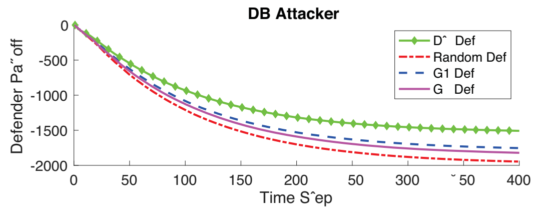

Fig. 6:Defender polices' payoff vs. DB attacker in Env-Medium.

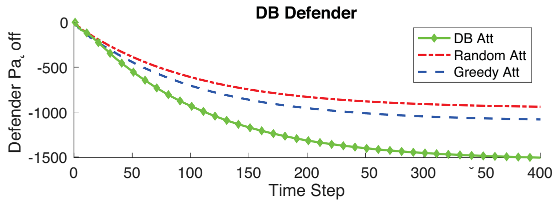

Fig. 7: DB defender's Payoff vs. attacker policies in Env-Medium.

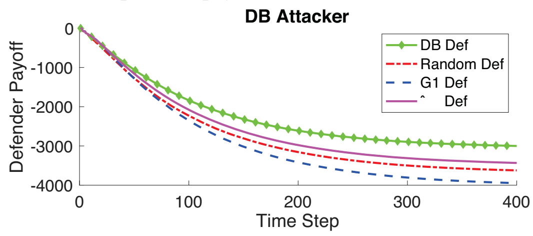

Fig. 8: Defender polices' payoff vs. DB attacker in Env-Large.

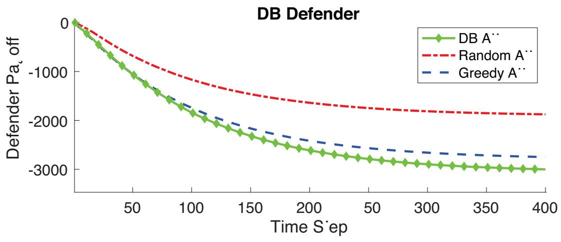

Fig. 9: DB defender's Payoff vs. attacker policies in Env-Large.

relative to its maximum warm-up period. $G2$ considers the stagnancy along with the threshold on the number of blocks that can be fully effective. $G2$ assigns blocks with the same approach as $G1$, however in the case that the limit is reached, $G2$ removes a block with the least weighted effectiveness from the lowest utility most protected target. A greedy attacker, on the other hand, chooses the action that achieves the highest immediate reward.

To compare policies, we ran 500 simulations with 400 time steps for each policy pair and averaged the achieved cumulative discounted reward (i.e., payoff) in every time step. Figures 4 and 5 show the defender's payoff on Env-Small for DB players. In Figure 5 we can observe that the DB attacker can inflict more damage (i.e., lower defender's payoff) on the system when the defender deviates from DB. This implies that our framework protects the system the most in comparison to other methods. In Figure 5, we show that the DB defender would gain if the attacker deviates from a DB policy. Since Env-Small is a simple environment, the attacker's task is straightforward and thus can also achieve a payoff similar to DP. Notice that DB players have almost similar behaviors as DP ones ensuring that DB is a good approximator for DP.

As shown in Figures 6 and 7, both DB players in Env-Medium have a significantly better performance than other heuristics. We noticed that the DB defender intelligently does not allow the number of assigned check blocks to reach the capacity constraints (as opposed to the greedy defender). This keeps the DB defender with more open options and allows him to hedge against the need to remove blocks as the capacity is reached. This also allows him to recover easier when uncertainty happens. On the other hand, the DB attacker typically (and intelligently) accumulates her budget and strikes the system right after the occurrence of the uncertainty in consecutive time steps. Recall that the uncertainty is introduced to account for system upgrades and instabilities which present an opportune moment to attack the system from the perspective of the adversary.

Similar intelligent behaviors are observed on Env-Large and likewise, Figures 8 and 9 show that DB achieves the highest payoff in all cases. This environment is a fairly challenging one, especially for the defender as he needs to consider the uncertainty while keeping his behavior unpredictable on a very large state space. These results demonstrate the scalability of our approach. Overall, DB achieved between 13% - 30% better protection than other approaches.

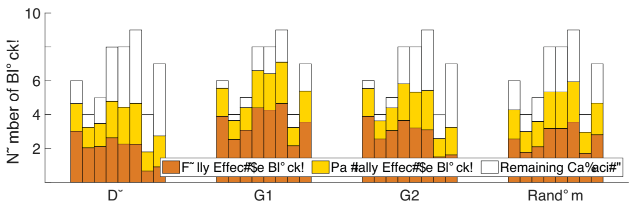

Fig. 10: Average number of assigned blocks.

Our next set of results delve deeper into the way DB assigns checks. Figure 10 represents the average number of fully assigned, partially assigned and unassigned blocks across defense policies on Env-Large. Each bar represents a target with the leftmost bar being the target with the highest utility. We observe that the DB defender strategically utilizes a smaller portion of targets' capacity compared to other polices. This allows for faster recovery when uncertainty occurs. Figure 11 shows the detection probability of the most frequent attack method across various policies over time on Env-Large. The top, middle, and bottom plots show the detection probability for high utility, medium utility and low utility target, respectively. One can observe that DB strategically assigns check blocks that provide the highest detection probability on the most important target (top figure) and provides less protection as the target utility decreases. The vertical lines indicate the point in time when uncertainty events happen and we can see that the DB defender is resilient against those events and in most cases can recover faster than other strategies.

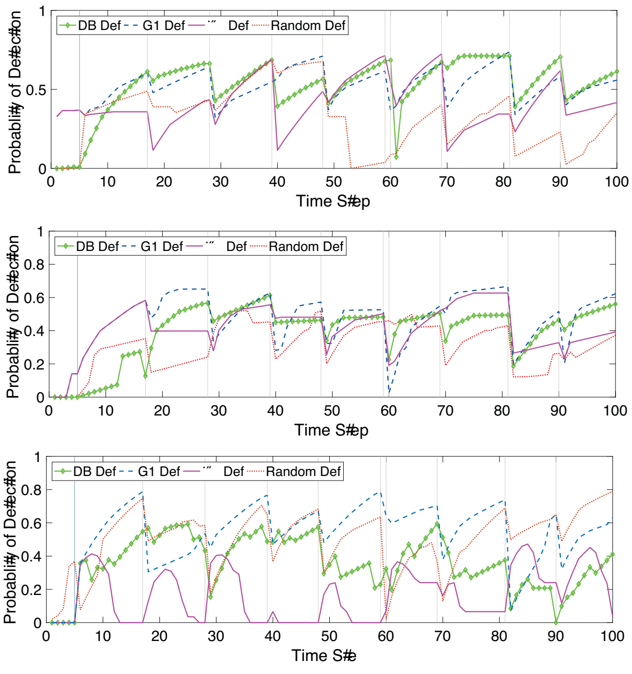

Fig. 11: Probability of detecting attacks across policies on the target with a highest utility (top), medium utility (middle), and lowest utility (bottom).

D. Evaluation on a Water Treatment System

As a case study, we evaluated DB on a six stage water treatment system called SWaT [1]. We focused on stage 1 of SWaT, which is called "Raw Water", and it is responsible for holding the water to be treated in the upcoming stages.

As shown in Figure 12, stage 1 is constituted of a water tank, level (LIT101) and flow (FIT101) sensors. Besides these components, it has an inlet motorized valve (MV101) and two outlet pumps (P101 and P102), which are controlled by a Programmable Logic Controllers (PLC) regarding the sensor measurements. Attacks on stage 1 would cause Tank-3 to overflow; Tank-3 is in stage 3, which handles ultra-filtration (UF).

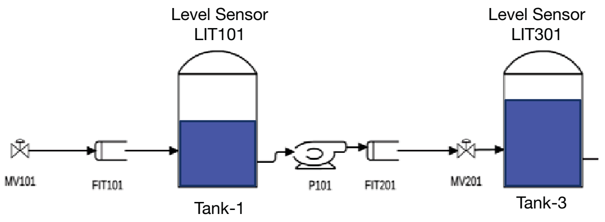

Fig. 12: Stage-1 of the water treatment system.

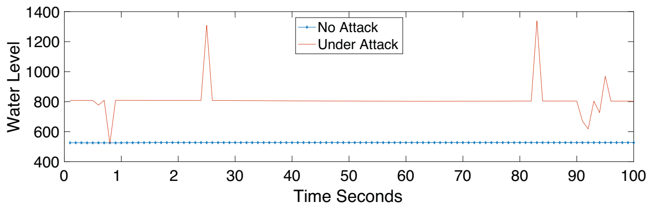

Fig. 13: Impact of cyber attacks on level sensor (LIT101).

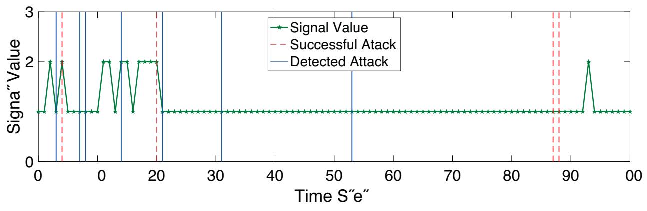

Fig. 14: Detection events over time for the DB defender.

We consider 6 attack methods: (1) random spoofing attack, (2) clamp attack that would fix the signal on a particular level beyond a particular value, (3) offset attack that would offset a signal by a positive or a negative value, (4) spoofing a signal with its log value (i.e., diminishing its value), (5) inverting the signal and (6) flipping a constrained set of bits. We also utilize a set of 6 check blocks: (1) cumulative sum (stateful), (2) checksum, (3) parity checks with previous signals (stateful), (4) maximum delta change over a window of values (stateful), (5) count of on/off changes (stateful) and (6) minimum delta change over a window of values (stateful).

We trained the DB players and conducted our experiments over 100 seconds (a time-step is 1 second). Since DB only detects attacks (i.e., an alarm is raised), their impact would continue on the physical plant. Figure 13 shows the impact of attacks (the attacker selects an attack method or none at every time step) on the water level sensor LIT101. The difference between the steady water levels (i.e., 527 and 800) was not due to the attack, but just the initial conditions. Since the water was being pumped from the first stage, we observed that the tank was being filled in stage 3 and would ultimately overflow.

We assess the success of DB in detecting attacks comparing it to the greedy approach (G1). Figure 14 shows the P101 signal value along with the attack detection events over 100 time steps. The vertical solid lines indicate attack detection events and the dashed ones indicate attacks that are not detected. DB defender was able to detect 55% of the attacks whereas the greedy approach was able to detect 39%. It is worth mentioning that these detection percentages depend on the effectiveness matrix which had an overall average effectiveness of 46% across all of its elements.

VI. Conclusions

Protecting CPS against cyber attacks requires the careful selection and integration of checks within the control loop. Despite the recent development of a wide selection of stateless and stateful checks to detect anomalies, a coherent approach to integrate such checks in a preventative manner is lacking. In this work, we develop a stochastic game model that captures the interaction between the defender and the adversary and present a deep reinforcement learning framework that provides optimal and sub-optimal policies that we coin DeepBLOC (DB). DB derives strategies that randomizes the assignment of checks based on their effectiveness, statefulness, warm-up periods, the importance of the target signals and the capabilities of the adversary. Through engineering efforts and trial-and-error approaches, we identify an architecture (along with its hyperparameters) that consistently captures the quality of the actions of the defender and adversary based on the current state. Through DB we are able to obtain policies that are Nash equilibrium conformant in environments with large state and action spaces. Through our extensive evaluation we show that the policies obtained with DB are very close to optimal polices that are obtained using dynamic programming on small environments and outperform other policies – with improvements reaching 30% – on larger environments.

Acknowledgment

This work was supported in part by NSF award #1814064.

References

[1] C. M. Ahmed, J. Zhou, and A. P. Mathur. Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in cps. In *ACSAC*, NY, USA, 2018. ACM.

[2] S. Amin, A. Cárdenas, and S. Sastry. Safe and Secure Networked Control Systems under Denial-of-Service Attacks. *Hybrid Systems: Computation and Control*, pages 31–45, 2009.

[3] R. Bobba, K. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T. Overbye. Detecting False Data Injection Attacks on DC State Estimation. In *The First Workshop on Secure Control Systems, CPS Week*, 2010.

[4] G. W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.

[5] M. Brown, A. Sinha, A. Schlenker, and M. Tambe. One Size Does Not Fit All: A Game-Theoretic Approach for Dynamically and Effectively Screening for Threats. In *AAAI conference*, 2016.

[6] C. Do, N. Tran, C. Hong, C. Kamhoua, K. Kwiat, E. Blasch, S. Ren, N. Pissinou, and S. Iyengar. Game theory for cyber security and privacy. *ACM Computing Surveys (CSUR)*, 50(2):30, 2017.

[7] A. Farraj, E. Hammad, A. Al Daoud, and D. Kundur. A game-theoretic analysis of cyber switching attacks and mitigation in smart grid systems. *IEEE Transactions on Smart Grid*, 7(4):1846–1855, 2016.

[8] M. Guirguis, A. Tahsini, K. Siddique, C. Novoa, J. Moore, C. Julien, and N. Dunstatter. Bloc: A game-theoretic approach to orchestrate cps against cyber attacks. In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2018.

[9] C. N. Hadjicostis and R. Touri. Feedback control utilizing packet dropping network links. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 2, pages 1205–1210. IEEE, 2002.

[10] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. Through the eye of the plc: semantic security monitoring for industrial processes. In *ACSAC*, pages 126–135. ACM, 2014.

[11] J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*, 2015.

[12] J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[13] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4190–4203, 2017.

[14] Y. Li, L. Shi, P. Cheng, J. Chen, and D. E. Quevedo. Jamming attacks on remote state estimation in cyber-physical systems: A game-theoretic approach. *IEEE Transactions on Automatic Control*, 60(10), 2015.

[15] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, volume 157, pages 157–163, 1994.

[16] M. L. Littman. Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2(1):55–66, 2001.

[17] Y. Liu, P. Ning, and M. Reiter. False Data Injection Attacks against State Estimation in Electric Power Grids. In *ACM CCS*, 2009.

[18] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[21] Y. Mo and B. Sinopoli. False Data Injection Attacks in Control Systems. In *Proceedings of the 1st workshop on Secure Control Systems*, 2010.

[22] N. Perlroth, M. Scott, and S. Frenkel. Cyberattack Hits Ukraine Then Spreads Internationally. https://www.nytimes.com/2017/06/27/technology/ransomware-hackers.html, June 2017.

[23] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. Sastry. Foundations of Control and Estimation Over Lossy Networks. *IEEE*, 95(1):163, 2007.

[24] A. Sinha, T. H. Nguyen, D. Kar, M. Brown, M. Tambe, and A. X. Jiang. From Physical Security to Cybersecurity. *Journal of Cybersecurity*, 1(1):19–35, 2015.

[25] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry. Kalman Filtering with Intermittent Observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, 2004.

[26] A. Tiwari, B. Dutertre, D. Jovanović, T. de Candia, P. Lincoln, J. Rushby, D. Sadigh, and S. Seshia. Safety Envelope for Security. In *HiCoNS*, pages 85–94. ACM, 2014.

[27] N. Trcka, M. Moulin, S. Bopardikar, and A. Speranzon. A Formal Verification Approach to Revealing Stealth Attacks on Networked Control Systems. In *HiCoNS*, Chicago, IL, April 2014.

[28] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *ACM CCS*, 2016.

[29] Y. Wang, Z. Xu, J. Zhang, L. Xu, H. Wang, and G. Gu. SRID: State Relation Based Intrusion Detection for False Data Injection Attacks in SCADA. In *ESORICS*, pages 401–418. Springer, 2014.

[30] M. Yu, L. Wang, T. Chu, and G. Xie. Stabilization of Networked Control Systems with Data Packet Dropout and Network Delays via Switching System Approach. In *IEEE CDC*, 2004.

[31] K. Zetter. A Cyberattack Has Caused Confirmed Physical Damage for the Second Time Ever. ttp://www.wired.com/2015/01/german-steel-mill-ack-destruction/, 2015.

[32] M. Zhu and S. Martinez. Stackelberg-game analysis of correlated attacks in cyber-physical systems. In *American Control Conference*, 2011.

[33] Q. Zhu and T. Basar. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems. *IEEE control systems*, 35(1):46–65, 2015.