BADControl: Backdoor Attacks Against Control Systems

Luis Burbano[†], Hampei Sasahara[§], Ruoyu Song[‡], Z. Berkay Celik[‡], and Alvaro A. Cardenas[†]

[†] *University of California, Santa Cruz, {lburbano, alacarde}@ucsc.edu*
[§] *The University of Tokyo, hsasahara@g.ecc.u-tokyo.ac.jp*
[‡] *Purdue University, {song464, zcelik}@purdue.edu*

Abstract

We introduce BADControl, the first backdoor attack against low-level controllers that uses physical triggers. The attack poisons operational data to implant a vulnerability that can be activated by an exogenous signal from the environment, such as a specific driving maneuver or adversarial road patches within autonomous driving applications. BADControl solves a constrained optimization problem by using a projected gradient ascent to modify the data, maximizing the frequency response of the controlled system at a target frequency. This method differs from backdoor attacks against Deep Learning (DL) and Reinforcement Learning (RL) models, which manipulate high-dimensional model inputs or reward functions. We additionally propose two defenses: one based on regularization and one based on robust optimization, to limit the worst-case amplification of trigger signals. This is achieved by converting infinite poisoning scenarios into a single, tractable optimization problem via a specialized mathematical transformation. We evaluate BADControl on Proportional-Integral-Derivative (PID) and Linear-Quadratic-Regulator (LQR) controllers through simulations and physical experiments. In the adaptive cruise control scenario, we achieve a 100% crash rate, while in lane-keeping control, the backdoor causes the victim vehicle to steer 62% into the opposing lane, compared to 0% in both cases without a backdoor. By contrast, a state-of-the-art falsification framework for autonomous vehicles identifies only a single crash instance over 30 trials, underscoring its stealthiness.

1 Introduction

Machine learning and data-driven methods have become central to the design and deployment of autonomous vehicle systems. From relatively simple driver assistance features such as lane-keeping assist and adaptive cruise control, to advanced perception, planning, and decision-making stacks in fully autonomous vehicles, data-driven approaches enable controllers to adapt to diverse operating conditions. These techniques have significantly improved robustness, performance, and adaptability, fueling rapid innovation in both commercial driver assistance systems and self-driving platforms.

However, with the increasing adoption of data-driven methods comes the risk of adversarial machine learning [12]. A large body of work has shown that learned models, including those in autonomous driving, can be compromised by training-time data poisoning or test-time physical triggers.

In the **perception** domain, researchers have shown that deep learning models for lane detection, traffic sign recognition, and other tasks can be misled by physical world perturbations, such as adversarial patches or environmental modifications [19, 26, 41, 60–62]. These attacks can cause a vehicle to drift from its lane, misclassify signs, or take unsafe actions, even in realistic driving scenarios.

In the **prediction and planning** domain, *Pourkeshavarz et al.* [38] create a backdoor against the planning module to cause a misprediction of the actors' trajectories. *TrojDRL* [32] evaluates backdoor attacks against deep reinforcement learning (DRL) agents, implanting triggers that cause targeted misbehavior while maintaining benign performance. *BackdooRL* [51] shows how competitive multi-agent DRL systems can be compromised by embedding malicious policies that activate only under specific environmental conditions. *Stop-and-Go* [53] introduces backdoors in DRL-based traffic congestion control systems to cause congestion while appearing benign under normal operation. Finally, a broader study [52] analyzes the general vulnerability of DRL in applications of autonomous vehicles to backdoor attacks, highlighting the potential safety risks of such compromises.

Despite this growing body of work, none of these efforts have focused on the **control** domain, particularly low-level classical feedback algorithms such as Proportional–Integral–Derivative (PID) [5, 49] and Linear–Quadratic–Regulator (LQR) [2, 3] control. PID and LQR controllers are the workhorses of modern control engineering, forming the backbone of virtually every safety-critical control stack (from automotive and aerospace to industrial automation) due to their simplicity, reliability, and decades of

proven deployment [4, 29]. We believe that studying adversarial machine learning in this unexplored but widely used area is needed to fully understand the risks to cyber-physical systems.

In this paper, we introduce BADCONTROL, a backdoor attack against control systems that injects a physical trigger to disrupt the functionality of a controller, while preserving safe behavior otherwise. BADCONTROL consists of two phases: (1) an offline phase in which the attacker poisons the dataset to embed a backdoor on the controller, and (2) an online phase where the attacker triggers the backdoor. Unlike previous work on backdoor attacks in DL and DRL, BADCONTROL uses dynamic physical attacks as a trigger mechanism, such as an adversarial driving maneuver by manipulating a vehicle's acceleration and deceleration or placing patches on the road to affect the victim's perception. This necessitates a novel approach to backdoor attacks and defenses in the context of low-level controllers.

To propose and evaluate this attack, we overcome several technical challenges: (1) formally defining the problem of backdooring low-level controllers, (2) proposing, to the best of our knowledge, the first backdoor trigger mechanism based on time series inputs, and (3) solving a new infinite-dimensional constraint problem to develop a robust defense that can provably prevent such attacks. Our results demonstrate that even the most fundamental control algorithms can harbor hidden backdoors that threaten safety in autonomous driving.

We demonstrate the effectiveness of our approach in two vehicle applications: adaptive cruise control (ACC) and lane keeping control (LKC). In ACC, the attacker causes a crash by accelerating and decelerating in front of the victim's vehicle. We achieve an attack success rate of 100%, while the attacker cannot create crashes against a controller without a backdoor. In LKC, the attacker causes the vehicle to invade another lane by placing patches on the road, thereby affecting the victim's perception. In this case, the attacker can create a safety violation by making 62% of the vehicle invade another lane.

To defend against these attacks, we introduce a robust optimization algorithm that improves the resilience of controllers against data poisoning attacks. Because an adversary can arbitrarily inject attacks, the formulated problem has an infinite number of optimization constraints, and to solve this problem, we prove a new theorem showing how to adapt this problem into a computationally tractable optimization algorithm. Theoretical analysis and simulation results demonstrate its effectiveness in limiting backdoor injection into controllers. Even if the attacker poisons the dataset, the attacker cannot cause crashes through several simulations with this defense.

In this paper, we make the following contributions:

- We introduce BADCONTROL, the first backdoor attack against low-level controllers. In particular, we first formulate the problem of backdooring PID and LQR controllers, and then propose a framework to inject backdoors during the parameter tuning of a controller design.

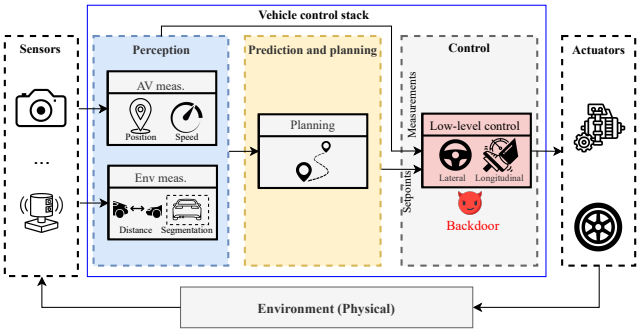


Figure 1: Autonomous vehicle control stack with backdoor injection in the low-level controller. The backdoor is triggered by an adversary in the physical environment, for instance, through subtle manipulations affecting the vehicle (e.g., slight acceleration of a nearby vehicle).

- We propose a new trigger format: the use of dynamic physical attacks as triggers, which expands the scope of backdoor attacks beyond conventional triggers (e.g., a specific pattern of pixels added to samples) in traditional ML algorithms.

- We evaluate BADCONTROL on two automotive systems that integrate data-driven adaptive cruise and lane-keeping control. We demonstrate that an attacker can cause a crash by accelerating and decelerating in front of a victim's vehicle and placing patches on the road that affect its perception.

- To counter backdoor attacks, we introduce two defenses, a regularization approach useful for PID controllers and a robust optimization algorithm for LQR controllers, which optimizes the controller's performance for the nominal (clean) data while constraining the closed-loop characteristic (e.g., H_∞ norm) for all perturbed data. This ensures that the controller maintains a certain level of robustness in the presence of adversarial perturbations.

2 Background and Related Work

2.1 Control Systems

Control systems are a foundational component of modern infrastructure. They operate in applications such as aircraft autopilot systems, electrical power grid regulation, and industrial control systems. Given their broad scope and critical requirements, they are often structured into layers of abstraction [37]. These layers define system objectives and coordination by interpreting sensor data, setting targets, and regulating system states under disturbances, noise, or uncertainties.

One key example of these layers is the autonomous vehicle (AV) driving stack as shown in Figure 1. This stack has perception, prediction & planning, and control [55]. These three distinct yet tightly coupled phases transform raw sensor data into safe vehicle motion. The *perception* phase processes

input from cameras, LiDAR, radar, GPS, and other sensors. This builds a structured understanding of the environment, e.g., detecting and classifying objects such as vehicles and pedestrians. It also involves identifying lane boundaries and traffic signals, and estimating the ego vehicle's state. Following this, the *prediction & planning* phase reasons about the future. It predicts likely trajectories and behaviors of surrounding agents under uncertainty. It then generates a feasible plan for the ego vehicle. This plan adheres to traffic rules, avoids collisions. The plan may be a high-level route or a fine-grained trajectory as a sequence of target positions and velocities.

The *control* phase translates these setpoints into concrete actuator commands that include throttle, braking, and steering through control algorithms. PID [5, 49], a feedback controller that adjusts control output based on error, its integral, and its derivative, and LQR [2, 3], an optimal control method that determines control inputs to minimize a quadratic cost function, are widely used controllers in AVs. Here, while perception and planning operate in high-dimensional, semantic spaces, control algorithms ensure precise and stable execution of the trajectory in the physical world by rejecting disturbances and maintaining control fidelity.

PID and LQR are critical for AV functions such as adaptive cruise control (ACC) and lane-keeping control (LKC), corresponding to SAE Levels 1 and 2 of driving automation. Table 1 summarizes their specific applications, required measurements, and control actions for ACC and LKC [3, 5]. For ACC, both PID and LQR adjust vehicle acceleration based on inter-vehicle distance. LQR also considers relative velocity, vehicle acceleration, and the integral of the inter-vehicle distance error. For LKC, PID uses position with respect to the lane center to adjust steering, while LQR also incorporates the vehicle's angle with respect to the lane center.

Table 1: Measurements and control actions for LKC and ACC applications with PID and LQR controllers.

System	Controller	Measurements	Action
ACC	LQR	Inter-vehicle distance Relative velocity Vehicle acceleration Integral of distance error	Acceleration\ Deceleration
	PID	Inter-vehicle distance	
LKC	LQR	Position wrt the lane center Angle wrt the lane center	Steering
	PID	Position wrt the lane center	

2.2 Backdoor Attacks

Backdoor attacks are a type of security threat that introduces hidden vulnerabilities into a system, which are then activated by a specific trigger to cause targeted misbehavior [6]. These vulnerabilities are typically introduced during the model training phase through data poisoning.

In the context of AVs, these attacks are investigated in DL models for perception and prediction through various sensors. For perception, triggers can involve physical-world manipulations. For example, specific adversarial patches on roads activate backdoors in camera-based systems [26, 60, 62]. Environmental changes can also affect LiDAR [61] and temperature sensors [57], which may trigger backdoors.

More recently, RL agents, responsible for planning and decision-making of an AV, have been shown to be vulnerable to backdoor attacks. An RL agent with a backdoor maximizes long-term reward under normal conditions, but its performance decreases when the trigger is presented. In these attacks, an attacker modifies the agent's observed data [15, 24, 32, 56]. For example, an attacker can place a patch on an image observed by the RL agent, or steer the environment to a specific state through physical actions, rather than modifying the observed data [51, 53]. These trigger the backdoor and cause the agent to take a harmful action that reduces future reward.

The previous efforts focus on different layers of an AV stack. DL backdoors are most effective in the perception layer, and DRL backdoors are most effective in the prediction and planning layer. However, as far as we are aware, there are no backdoor attacks at the low-level control layer. In this paper, as illustrated in Figure 1, we formalize the problem and show how attackers can embed backdoors in popular controllers.

Moreover, we propose new defenses against backdoor attacks in control systems. Different classifications of defenses against backdoor attacks for ML exist [23, 50, 59]. Based on those, we classify the defenses into three groups: (1) training-time defenses, (2) model correction and backdoor identification, and (3) input detection. Training-time defenses modify the learning process to improve robustness against poisoning attacks, for example, by altering the training objective or by pruning contaminated samples to ensure training on clean data. Model correction and backdoor identification defenses determine whether a model has a backdoor and remove it. Finally, input detection defenses aim to identify and reject inputs containing backdoor triggers at inference time. In this paper, we explore defenses based on the first approach.

3 Problem Statement and Motivation

We consider an AV as a physical process described by a state vector $x_k \in \mathbb{R}^n$ at each sampling time step k with n system states with sampling period $T_s > 0$. Here, the perception module transforms the sensor measurements into an observation vector $y_k \in \mathbb{R}^p$. The controller computes control actions $u_k \in \mathbb{R}^m$. These actions modify system dynamics. In addition, the system is disturbed by an exogenous signal $d_k \in \mathbb{R}^\ell$ originating from the environment, which may be exploited by an attacker to trigger an embedded backdoor. The state-update

and observation equations are expressed as:

$$x_{k+1} = f(x_k, u_k, d_k), \quad y_k = g(x_k, u_k, d_k), \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \to \mathbb{R}^n$ describes the dynamics of the physical process, and $g : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell \to \mathbb{R}^p$ models how observations depend on state and control inputs.

The controller design problem involves constructing a controller κ. This controller computes action u_k such that the vehicle follows the plans from the planning module. Formally, κ receives processed sensor measurements y_k from the perception module and setpoints $r_k \in \mathbb{R}^p$ from the planning module. The objective is to drive y_k to r_k.

PID and LQR controllers depend critically on a small set of parameters. For PID, control action u_k depends on error $e_k = y_k - r_k$, its sum, and its rate of change:

$$\kappa : \begin{cases} u_k = K_p e_k + K_i \sum_{i=1}^{k} e_i T_s + K_d \frac{e_k - e_{k-1}}{T_s}, \\ e_k = y_k - r_k, \end{cases} \tag{2}$$

where K_p, K_i, and K_d are design parameters. For LQR, where the full state is observed ($y_k = x_k$ and $p = n$), the control action with K as the design parameter is:

$$\kappa : \begin{cases} u_k = K e_k, \\ e_k = y_k - r_k, \end{cases} \tag{3}$$

Thus, in both cases, control design reduces to estimating appropriate parameter values. For this, designers often rely on data to obtain the parameters from a sequence of trajectories that comprises inputs U and outputs Y:

$$U = [u_0, \ldots, u_{T-1}] \in \mathbb{R}^{m \times T}, \; Y = [y_0, \ldots, y_T] \in \mathbb{R}^{p \times (T+1)}. \tag{4}$$

Designers typically formulate the parameter tuning algorithm $\mathcal{K}$ through an optimization problem dependent on this data (See Appendix A for details).

The critical insight and motivation for this work is that *PID and LQR parameters are data-derived. This makes them vulnerable to data poisoning. An adversary corrupting* (U, Y) *can bias the design of the algorithm* $\mathcal{K}$. This can produce a controller with a backdoor that can be triggered by manipulating the controller inputs in the physical world.

Motivating Example. Figure 2 illustrates a backdoor in an ACC scenario on the road. A yellow vehicle, operating with a backdoored controller, follows a benign blue vehicle on the highway. Normally, the system maintains a safe inter-vehicle distance. However, an attacker, aware of the embedded backdoor and driving a red vehicle, cuts in front of the blue car. The attacker then performs a precisely designed adversarial maneuver, such as accelerating and decelerating in a fixed pattern. With a clean controller, the yellow car responds correctly and avoids collisions, as indicated by the blue line to the right of Figure 2. In contrast, with the backdoored controller, this same maneuver activates hidden dynamics. This causes the yellow vehicle to misbehave and collide with the blue car.

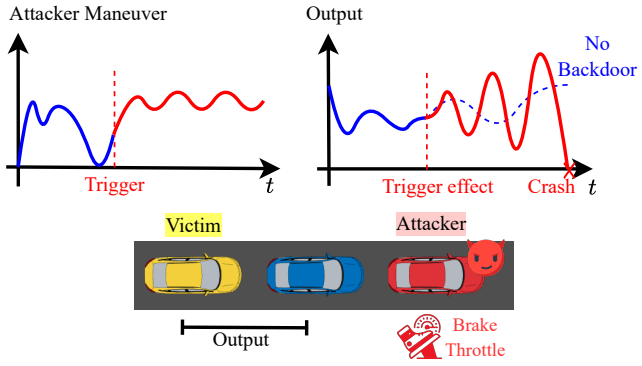


Figure 2: Example of a backdoor attack against a controller.

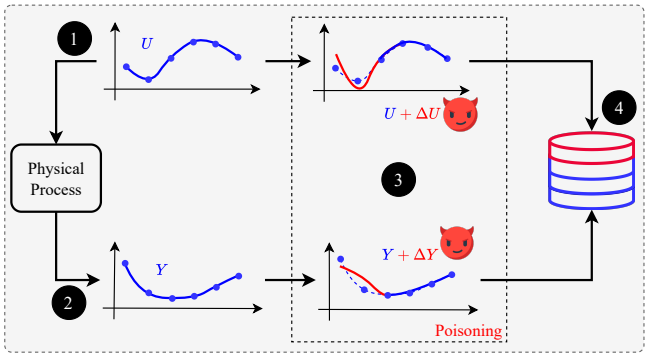
4 Design Requirements and Challenges

The problem of injecting backdoors into controllers presents several distinct challenges for stealth and exploitation in the physical world. These challenges distinguish this type of attack from those targeting DL and DRL models.

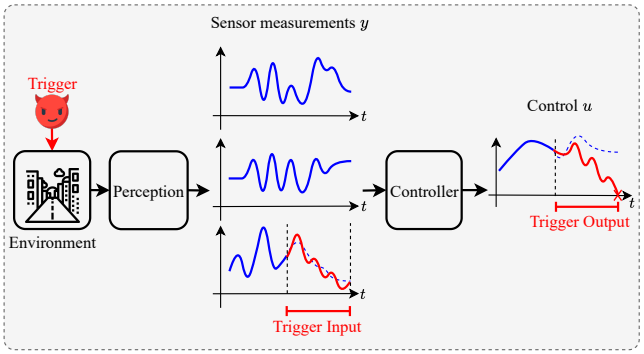
Absence of a Clear Attack Objective (C1). Because we are the first to study backdoor attacks against controllers, we begin by formalizing what a *backdoor* even means in this setting. Unlike DL, which has classification accuracy, or DRL, which relies on a reward function, data-driven controllers do not have these metrics. Simply inverting the controller's optimization objective (e.g., maximizing the tracking error) would cause poor performance, making the attack obvious and preventing the controller from ever being used. Therefore, the attacker cannot simply target a standard metric. Instead, the attacker must manipulate a deeper property of the physical system, such as its frequency response or controllability. This property would not be noticeable until the attack is triggered.

Maintaining Attack Stealth (C2). Low-level controllers have fewer parameters than neural networks, so modifying these parameters is more easily detected. This means the attacker must perform a subtle poisoning of the dataset to avoid detection while still embedding a functional backdoor. The poisoned controller must also perform normally and maintain nominal performance without the trigger. For instance, in the ACC scenario, the backdoored controller must keep a safe distance between vehicles under normal conditions, and it must not show any signs of instability or poor performance that would alert a designer. This requires a fine balance between embedding an effective trigger and remaining completely hidden during normal operation.

Designing a Dynamic and Physically Realizable Trigger (C3). Unlike DL attacks that use simple artifacts such as specific pixel patterns, a backdoor against a data-driven controller requires a dynamic and physically realistic trigger. The attacker must create a time-series input to activate the hidden vulnerability. This requires a new approach to trigger mechanisms. For example, in an ACC scenario, the attacker's



(a) Poisoning the Dataset.



(b) Triggering the Backdoor.

Figure 3: Threat model. The blue lines represent the original data, while the red lines represent the attacker's actions.

vehicle must perform a precisely designed adversarial maneuver, such as accelerating and decelerating in a fixed pattern. This generates a sinusoidal signal at a target frequency that the backdoored controller will amplify. The complex, time-dependent nature of this trigger is fundamentally different from trigger implementations in a typical DL attack.

5 Threat Model

Our work follows a threat model similar to other backdoor attacks in DL and RL, consisting of two phases: an offline phase and an online phase, as shown in Figure 3. In the offline phase, an attacker poisons part of the trajectories, while in the online phase, the attacker triggers the backdoor.

The adversary has two main objectives for embedding a backdoor. The first is *stealth*: the compromised controller must behave normally and avoid unsafe conditions without the trigger, ensuring the victim trusts and continues to use it. The second is *exploitation*: once the trigger is activated, the backdoored controller must cause the system to enter an unsafe state, potentially leading to a crash.

During the offline phase, the attacker modifies the trajectories by adding a disturbance, $\Delta = (\Delta U, \Delta Y)$. This results in a poisoned dataset of $(U + \Delta U, Y + \Delta Y)$. A realistic method for this is a supply chain attack. Because system manufacturers rely on large amounts of data collected across fleets or obtained from third-party sources, this attack can occur in two ways: (1) an attacker gains access to the data collection hardware (e.g., compromising data loggers of test fleet vehicles) and injects malicious measurement errors; or (2) an attacker corrupts external data suppliers or shared repositories that provide the training data. The victim can then retrieve the compromised data to tune their controller, which will then contain a backdoor. While the attacker must have some level of access, data poisoning follows a similar threat model to compromises involving machine-learning datasets [24, 27, 42].

In the online phase, we consider that the attacker modifies the physical environment of the target vehicle. This involves altering lane markings or controlling a nearby vehicle to perform a maneuver that triggers the backdoor. The specific method depends on the AV function the controller manages, such as ACC or LKC (detailed in Section 6).

6 BADControl

We introduce BADCONTROL, the first backdoor attack against controllers. The attack has an offline phase where the attacker solves an optimization problem to poison the trajectories and an online phase where the attacker triggers the backdoor.

To address C1, we evaluate physical properties that attackers can target. Properties such as *controllability* [36] or *attractivity and stability* [25] are inherent features of the closed-loop system and, once the controller is deployed, cannot be selectively activated by an adversary. What we require instead is a property that remains dormant under normal conditions but can be deliberately excited by an attacker during operation. We therefore propose an attack that targets the *frequency response* of the controlled system, which quantifies how the system reacts to an input at a given frequency [21]. Unlike structural properties such as stability or controllability, the frequency response can be manipulated online by injecting adversarial signals, allowing an attacker to selectively trigger the backdoor at will while leaving the system's nominal performance otherwise intact.

To address C2, we design a carefully crafted objective function that embeds the backdoor while preserving the controller's apparent normalcy. Specifically, it ensures that the backdoored controller behaves close to a clean controller under standard operating conditions, so that it passes testing and validation without raising alarms. Only when the adversary introduces the trigger (or a signal sufficiently close to it, for robustness) does the malicious behavior emerge. This design makes the attack stealthy: the backdoor remains hidden during routine operation, yet can be reliably and selectively activated on demand by the attacker.

During the online phase, BADCONTROL uses a dynamic

physical trigger that corresponds to the manipulated property. By targeting the frequency response, the attacker creates a sinusoidal input signal at a specific frequency, ω_{attack} (addressing C3). This contrasts with traditional backdoor attacks that rely on static or pixel-based triggers. The physical trigger, such as an adversarial driving maneuver or patches on the road, translates into the targeted sinusoidal signal that activates the hidden vulnerability and drives the system to an unsafe state.

6.1 Backdoor Embedding (Offline Phase)

We create a subtle yet effective attack by selecting an objective function that manipulates the controller's underlying physical properties rather than its surface performance. Our goal is to embed a trigger that is only activated by a specific external physical stimulus. We therefore require a property that (i) is fundamental to closed-loop behavior, (ii) admits precise mathematical characterization, and (iii) can remain latent until the attacker deliberately excites it.

The key clue comes from considering not whether the system can be controlled, but *how it responds to signals of different frequencies*. In particular, we are interested in the worst-case input pattern (in terms of frequency components) that this system amplifies the most. With these requirements in mind, we turn to the H_∞ norm [21], which quantifies the worst-case amplification of signals across frequencies. At first glance, using H_∞ as an attack vector is counterintuitive, as it is traditionally a tool for robust controller design rather than to introduce vulnerabilities. However, we will show that by optimizing it in a targeted way and with the appropriate constraints, the attacker can embed a frequency-selective resonance into the controller: under normal conditions, the controller behaves indistinguishably from a clean design, but when the adversary injects input at the chosen frequency, the system amplifies it disproportionately.

The H_∞ norm is analyzed through the linearized system of (1) around an operating point as

$$x_{k+1} = Ax_k + B_u u_k + B_d d_k, \quad y_k = Cx_k + D_u u_k + D_d d_k, \quad (5)$$

where A, B_u, B_d are the Jacobian matrices of f with respect to x, u, d, respectively, while C, D_u, D_d are those of g. Consequently, the closed-loop system with controller $\kappa(y_k)$ becomes $x_{k+1} = Ax_k + B_u\kappa(y_k) + B_d d_k$ with $y_k = Cx_k + B_u\kappa(y_k) + D_d d_k$. Applying the discrete-time Fourier transform to the signals, we obtain *the frequency-domain representation* $y(e^{j\omega}) = G_\kappa(e^{j\omega})d(e^{j\omega})$. The H_∞ norm ($\|\cdot\|_\infty$) of the controlled system, which measures the worst-case gain across all possible frequencies, is given by:

$$\|G_\kappa\|_\infty = \sup_{\omega\in[0,2\pi)} \sigma_{max}(G_\kappa(e^{j\omega}))$$

where $\sigma_{max}(G_\kappa(e^{j\omega}))$ is the maximum singular value of the transfer function at the frequency ω. In words, $\|G\|_\infty$ is the largest possible gain that the system applies to an input of any frequency, characterizing the relation between the system inputs and outputs. Accordingly, we set the attacker's objective function to $F(\kappa) = \|G_\kappa\|_\infty$. By poisoning the training data to maximize it, we secretly increase the system's sensitivity at a chosen frequency ω_{attack}, creating a trigger that the attacker can reliably exploit in the online phase (See Section 6.2).

We also ensure that the poisoning is subtle by constraining the disturbance to be small and by poisoning only a portion of the dataset. This makes the attack difficult to discover during the design and testing phases, and also ensures the controller remains stable after poisoning, as an unstable controller would be immediately discarded by algorithm designers. We detail these steps below.

6.1.1 Attack Constraints

Constraining Poisoning Size. To preserve stealthiness, the attacker does not want to introduce a large modification to the data (Y, U). This creates a constrained optimization problem. Thus, we impose the constraint:

$$||\Delta||_{max} \leq \varepsilon. \quad (6)$$

where $||\cdot||_{max}$ is the maximum entry norm.

The use of this norm implies that the maximum disturbance for each data point in Y and U is between $[-\varepsilon, \varepsilon]$. This constraint creates two sets of possible disturbances, $\mathcal{U}$ and $\mathcal{Y}$, which encode the set of possible disturbances for U and Y.

Ensuring Stability. A challenge to create backdoors against classical controllers is that they may become unstable. That means that the system is unable to accomplish the task using the controller, making the attack easily detectable. Consequently, we need to ensure that the backdoored controller preserves the system stability.

Given a system in Eq. (5) and a controller κ, we can get a matrix A_{cl}, which models the closed-loop system; for instance, for the LQR controller, we can use $A_{cl} = A + B_u K$. Now, we can study the system stability by analyzing the spectral radius of this matrix, denoted by $\rho(A_{cl})$. A system is stable if and only if the spectral radius of A_{cl} is less than one, i.e.,

$$\rho(A_{cl}) < 1, \quad (7)$$

which is known as Schur stability.

Poisoning Subset of Trajectories. The attacker would prefer to poison only a portion of the dataset, as if a large percentage of the data points were modified, the statistical properties of the trajectories could change dramatically, which can alert a developer. Thus, we impose a trajectory subset constraint:

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{0,\dots,T-1\}}} \frac{I_0(\Delta U_{i,j})}{2pT} + \sum_{\substack{i\in\{1,\dots,p\}\\ j\in\{0,\dots,T\}}} \frac{I_0(\Delta Y_{i,j})}{2m(T+1)} = \alpha \quad (8)$$

Algorithm 1 Backdoor generation.

Input: Trajectory (Y,U), Objective Function J, Maximum Perturbation Size ε, Target Control Algorithm $\mathcal{K}$, Step Size η, Maximum Iterations N, Poisoning Fraction α.
Output: Optimal Perturbation $(\Delta Y,\Delta U)$, Controller κ

1: $\Delta Y \leftarrow 0_{p\times T+1}$, $\Delta U \leftarrow 0_{m\times T}$, $M_Y \leftarrow 0_{p\times T+1}$, $M_U \leftarrow 0_{m\times T}$
2: **for** Each iteration until termination **do**
3: $(Df_Y, Df_U) \leftarrow (\nabla_Y J(Y+\Delta Y, U+\Delta U), \nabla_U J(Y+\Delta Y, U+\Delta U))$ # Gradient Computation
4: $(\Delta Y,\Delta U), Flag \leftarrow \tilde{\Pi}_{\mathcal{D}}((Df_Y,Df_U);(Y,U),(\Delta Y,\Delta U),\varepsilon,\mathcal{K},\eta,\alpha,M_Y,M_U)$ # Approximate Projection (Algorithm 2)
5: **if** Iteration number reaches N or $Flag$ **then**
6: Terminate
7: **end if**
8: **end for**
9: $\kappa \leftarrow \mathcal{K}(Y+\Delta Y, U+\Delta U)$
10: **return** $(\Delta Y,\Delta U)$, κ

where I_0 is an indicator function such that $I_0(a)=0$ if $a=0$ and $I_0(a)=1$ otherwise, for any $a\in\mathbb{R}$. α is the portion of the dataset that the attacker wants to compromise, and the subindex indicates the element i,j of a matrix $\Delta U_{i,j}$.

6.1.2 Optimization Objective to Create the Backdoor

We present an optimization objective $J(Y,U)=F(\mathcal{K}(Y,U))$ to create the backdoor:

$$\Delta = \underset{\Delta=(\Delta Y,\Delta U)}{\arg\max} \quad F(\mathcal{K}(Y+\Delta Y, U+\Delta U)) \tag{9a}$$

$$\text{s.t.} \quad \text{Equations (6)-(8).} \tag{9b}$$

We hereinafter denote the feasible set by $\mathcal{D}$. This is a constrained optimization problem, for which a natural solution approach would be the projected gradient method [9]. However, applying it here faces two major obstacles. (i) Gradient computation: The objective function in Eq. (9a) depends on the controller design algorithm $\mathcal{K}$, which is determined by solving an inner optimization problem. Thus, the attack design constitutes a *bi-level optimization problem*, making gradient evaluation nontrivial. (ii) Projection onto $\mathcal{D}$: Enforcing feasibility is difficult because the stability constraint in Eq. (7) is nonlinear and additionally the cardinality constraint in Eq. (8) introduces a combinatorial structure. Our proposed algorithm, summarized in Algorithm 1, addresses these issues by introducing an efficient gradient computation procedure (Line 3) and an approximate projection operator $\tilde{\Pi}_{\mathcal{D}}$ (Line 4). We now describe these techniques in detail.

Gradient Computation. BADCONTROL relies on computing the gradients $\nabla_Y J$ and $\nabla_U J$. A simple strategy is to approximate the gradient using numerical differentiation and finite differences. Although this approach is easy to implement, it is computationally inefficient and often inaccurate. In particular, it requires calling $\mathcal{K}$, the controller design algorithm, $O(T(m+n))$ times, which is highly expensive.

Algorithm 2 Approximate projection.

Input: Gradients (Df_Y, Df_U), Trajectory (Y,U), Perturbation $(\Delta Y,\Delta U)$ Maximum Perturbation Size ε, Target Control Algorithm $\mathcal{K}$, Step Size η, Poisoning fraction α, `mask` M_Y, M_U.
Output: Updated Perturbation $(\Delta Y,\Delta U)$, Termination Flag $Flag$

1: **if** Initial Iteration **then**
2: $Df_{cat} \leftarrow concat(|Df_Y|,|Df_U|)$
3: $b \leftarrow (1-\alpha)-th$ quantile of Df_{cat}
4: $M_Y, M_U \leftarrow \mathbb{I}_{\geq b}(|Df_Y|), \mathbb{I}_{\geq b}(|Df_U|)$ # Update **`mask`** where $|Df_Y|,|Df_U|$ exceeds threshold b
5: **end if**
6: $(\Delta\bar{Y},\Delta\bar{U}) \leftarrow (\Delta Y,\Delta U)+\eta(Df_Y\odot M_Y, Df_U\odot M_U)$ # Update perturbation
7: $\Delta\bar{Y},\Delta\bar{U} \leftarrow \text{proj}_{\varepsilon}(\Delta\bar{Y}), \text{proj}_{\varepsilon}(\Delta\bar{U})$ # Project to the ε-box
8: $\bar{\kappa} \leftarrow \mathcal{K}(Y+\Delta\bar{Y}, U+\Delta\bar{U})$
9: **if** System with controller κ is stable **then**
10: $\Delta U \leftarrow \Delta\bar{U}$, $\Delta Y \leftarrow \Delta\bar{Y}$ # Update perturbation
11: **else**
12: $Flag \leftarrow True$
13: **return** $(\Delta Y,\Delta U)$, $Flag$ # Return last stable perturbation
14: **end if**
15: $Flag \leftarrow False$
16: **return** $(\Delta Y,\Delta U)$, $Flag$

The PID control allows the direct computation of the gradient. Since the tuning problem is formulated as a regularized least-squares problem (see Appendix A.1), an explicit relationship exists between the controller parameters K_p, K_i, K_d and the perturbation Δ through its closed-form solution. Consequently, the gradients can be computed directly, either analytically or using automatic differentiation techniques.

However, the optimal solution cannot be obtained analytically for the LQR because the tuning problem is formulated as a semidefinite program (SDP) in Eq. (13) of Appendix A.2. To address this, we introduce an efficient gradient computation method based on *implicit differentiation* [40, 54]. Specifically, we treat the *KKT (Karush–Kuhn–Tucker) conditions* of the SDP [11, Section 5] as an implicit function linking the controller parameter K and the perturbation Δ. Differentiating these conditions yields a linear system for $\nabla_Y K$ and $\nabla_U K$, which can be solved analytically once the coefficients are derived. This approach significantly reduces computational overhead by requiring only one call to the data-driven control algorithm. Detailed mathematical background and derivation of this approach are provided in Appendix B.

Approximate Projection. Algorithm 2 summarizes the subroutine of Algorithm 1 for the approximate projection operator $\tilde{\Pi}_{\mathcal{D}}$, which incorporates two key insights. Our first insight addresses the cardinality constraint in Eq. (8). In the initial iteration (Lines 1–5), we select the data points to be perturbed by identifying those with the largest gradient magnitudes. This selection is motivated by the observation that the points with the steepest gradients contribute the most to increasing the objective function. By fixing the selected points for the

entire procedure, we avoid the combinatorial complexity of dynamically choosing them at each iteration. The perturbation update is then restricted to these points by applying a mask (Line 6, where $\odot$ denotes the element-wise Hadamard product). Our second insight enforces the stability constraint in Eq. (7). After each update, if the resulting closed-loop system becomes unstable (Lines 8–14), the update is rejected, and the algorithm terminates, returning the last stable perturbation. Finally, the projection onto the ε-box (Line 7), corresponding to (6), is carried out by simple truncation. In summary, Algorithm 2 modifies the update based on the raw gradients to ensure feasibility with respect to the set $\mathcal{D}$.

6.2 Backdoor Trigger (Online Phase)

The core of the online attack is to introduce a sinusoidal signal with a frequency of ω_{attack}, where the maximum singular value of $G_\kappa(e^{j\omega})$ is attained. The attacker does this by modifying the physical environment, causing the system to perceive an altered sensor measurement. The *attack amplitude*, a, is the magnitude of this malicious sinusoidal signal, determining the strength of the perturbation to the original sensor reading. We express this process with y_k (sensor measurements) without the attack and a as:

$$\tilde{y}_k = y_k + a\sin\left(\omega_{attack} \cdot T_s \cdot k\right). \tag{10}$$

In physical realization, this translates to different triggers depending on the AV functions that the controller is used for. In the ACC scenario, the attacker's vehicle accelerates and decelerates in a fixed pattern to introduce a sinusoidal signal into the victim's sensor measurements. In the LKC scenario, the attacker places adversarial patches on the road, which cause erroneous sensor readings and translate into the required sinusoidal signal. Those settings are explained in Section 7 in detail.

7 Experimental Setup

To show the effectiveness and generalizability of the system, we target two popular controllers: PID and LQR, which are widely adopted in AVs:

- **PID Control:** We target the Virtual Reference Feedback Tuning (VRFT) [14], which obtains the controller gains (K_p, K_i, and K_d) directly from trajectories. The details are in Appendix A.1.
- **LQR Control:** We target a data-driven LQR controller [16], which obtains the gain K directly from trajectories. The details are in Appendix A.2.

We demonstrate the system in two AV applications, ACC and LKC, that integrate PID and LQR controllers. BADCONTROL can be applied to several other systems that leverage

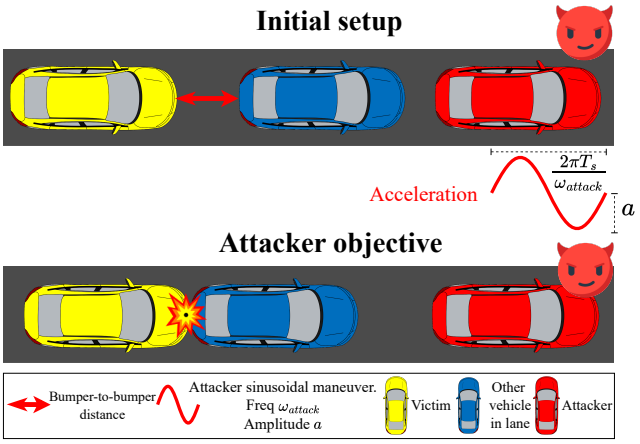


Figure 4: Backdoor attack on an ACC: The victim vehicle, in yellow, has a backdoored controller. The attacker's vehicle, in red, executes the maneuver that triggers the backdoor. The vehicle in blue is not part of the attack and uses a standard ACC controller.

such controllers in different applications. We deploy BADCONTROL in simulations and physical experiments.

All simulation experiments are performed in CARLA 0.9.15 [17], a high-fidelity simulator for autonomous driving, as it is actively supported. For real-world experiments, we employ robotic vehicles based on the BARC project [7]. Each vehicle includes a LiDAR, a camera, and wheel encoders and utilizes an Odroid XU4. The robots operate on Ubuntu Mate 20.04 and Python 3.

Developing the experimental setup required overcoming several technical challenges, both in the CARLA simulations and in the deployment on real-world robotic vehicles. These included (1) implementing and tuning the baseline controllers (Appendix A), (2) modeling the physical dynamics of the vehicles with sufficient fidelity to support both control design and poisoning (Appendix B), and (3) designing the attack pipeline, including the algorithms for injecting poisoned data into the training process (Appendix C).

7.1 AV Applications

Adaptive Cruise Control (ACC). ACC assists with highway driving by using a controller to maintain a safe distance from the vehicle ahead with a desired velocity. The attacker's objective is to trigger the victim's backdoor to cause a crash without being involved in the collision. This is illustrated in Figure 4, where the attacker, controlling the red vehicle, wants to cause a collision between the victim (the yellow car) and a third vehicle (the blue car).

Here, the attacker uses an adversarial maneuver to trigger the backdoor. By accelerating and decelerating in a sinusoidal pattern, the attacker introduces a signal with a specific frequency, ω_{attack}, into the victim's sensor measurements. To

avoid the crash, the attacker positions their vehicle in front of the blue car, which is followed by the victim's yellow car. The blue vehicle, also using ACC (without the backdoor), mimics the attacker's behavior, thereby transmitting the sinusoidal signal to the victim and activating the backdoor.

Lane-keeping Control (LKC). LKC systems use lateral control to keep a vehicle in the center of its lane. It uses sensors to measure the vehicle's position relative to the lane center and adjust steering accordingly, as shown in Figure 5. Here, the goal of the attacker is to force the victim's vehicle into a different lane. This can be accomplished by placing patches on the road [41]. The trigger has two steps: (1) first, the attacker places periodic, static patches on the road to cause the lane detection error to oscillate as the vehicle moves. (2) The oscillation then translates the spatial period of the patches into the required sinusoidal signal (frequency ω_{attack}) in sensor readings, triggering the backdoor. The altered sensor measurements cause the victim to deviate from the center of its lane and into a different lane.

7.2 Evaluation Metrics

Attack Success Rate (ASR). To measure the attack success against ACC, we compute the frequency of successful crashes between two victim vehicles. We define a crash as the bumper-to-bumper distance between the victim vehicle and another highway vehicle (not the attacker) reaching zero meters. The bumper-to-bumper distance is the space between the front-most point of the victim vehicle's chassis and the rear-most point of the vehicle ahead.

Encroachment Ratio (ER). To measure the success of attacks against LKC we use ER, which quantifies a vehicle's invasion beyond its lane boundary, which is calculated as

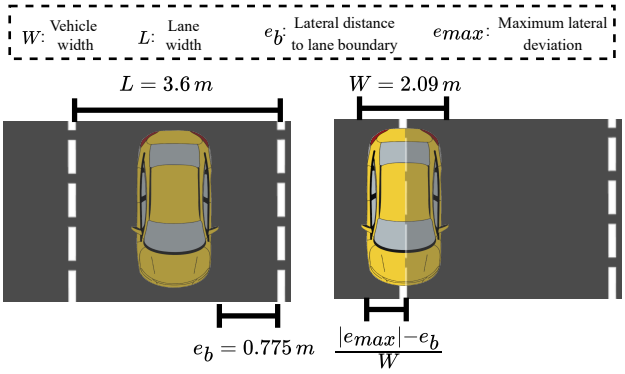
$$ER = \max\left(\frac{|e_{max}| - e_b}{W}, 0\right) \cdot 100$$

where e_{max} represents the maximum lateral deviation from the lane center during the simulation, W is the vehicle width, and e_b is the lateral deviation at which the vehicle first touches the lane marking.

Following the road patch attack work [41], a vehicle touches the marking when its lateral deviation reaches $\frac{L-W}{2}$, where L is the lane width. Using standard lane width defined by U.S. Federal Highway Administration [47] of $L = 3.6\,m$ $(12\,ft)$ and a vehicle with width of $W = 2.09\,m$, a car touches the lane if its lateral movement exceeds $e_b = 0.755\,m$ from the lane center.

8 Experimental Results

We present our BADCONTROL analysis results by focusing on the following research questions:



(a) $ER = 0.00\%$ (b) $ER = 50\%$

Figure 5: Car encroachment rate.

Table 2: Attack Success Rate (ASR) for the ACC experiments.

Controller	Max. Acc. Range $\pm[m/s^2]$	ASR [%]	
		No Backdoor	BADCONTROL
PID	[0.50, 3.00]	0.00	**33.33**
	[3.50, 5.00]	0.00	**100.00**
LQR	[0.50, 3.00]	0.00	**77.78**
	[3.50, 5.00]	0.00	**100.00**

RQ1 How effective is BADCONTROL against ACC measured in ASR and LKC measured in ER? (Section 8.1)

RQ2 How robust is BADCONTROL backdoor trigger to perturbations, allowing for activation even with noisy trigger injections? (Section 8.2)

RQ3 How stealthy is BADCONTROL backdoor to discovery by fuzzing techniques? (Section 8.3)

RQ4 Can BADCONTROL be effectively deployed in real-world systems? (Section 8.4)

8.1 Attack Effectiveness

We evaluate the system's effectiveness along two dimensions. First, we test whether the attacker can reliably induce unsafe behavior using the proposed triggering mechanism. Second, we verify that the same trigger fails when the vehicles are equipped with clean controllers, i.e., without the backdoor. This comparison is essential, as it isolates the effect of the backdoor and confirms that any observed safety violations arise directly from our backdoor rather than from inherent system vulnerabilities.

ACC. For the ACC application, we consider vehicles traveling at speeds of $72\,km/h$, $88\,km/h$, and $104\,km/h$, which include common speed limits in the US [13, 28]. To create attacker maneuvers, we use two levels of acceleration, ranging from $0.5\,m/s^2$ to $6\,m/s^2$, values that fall within the maximum acceleration capabilities of a Tesla Model 3 (our attacking vehicle

Table 3: Encroachment Rate (ER) for the LKC experiments.

Controller	ER [%]	
	No backdoor	BADCONTROL
PID	0.00	**61.99**
LQR	0.00	**43.03**

in CARLA). For each speed–acceleration pair, we evaluate the following two distances: 15 m and 20 m. This makes a total of 140 simulations. With these setups, we aim to demonstrate that an attacker can induce crashes under various conditions.

The results are summarized in Table 2. We first evaluate low accelerations and assume that the attacker can only accelerate up to a maximum of 3 m/s^2. With this maximum acceleration, the attacker has a success rate of 33% with a PID controller and 78% with an LQR controller. If the attacker accelerates at a higher rate $5 m/s^2$, we observe that the success rate becomes 100% for both controllers. These results demonstrate the effectiveness of our embedded trigger and the trade-offs that the attacker needs to consider; higher attack amplitudes are more likely to cause accidents. Furthermore, if the controller is not poisoned, the attack success rate drops to 0%.

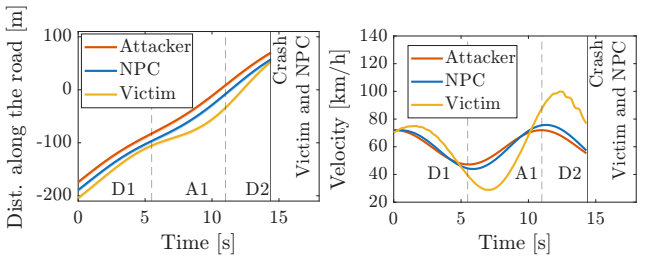
LKC. For the LKC scenario, we assume the vehicle travels at a constant speed. We first determine the minimum attack amplitude required for the vehicle to touch the lane boundary, assuming no backdoor is present. In this case, the ER is equal to zero. We then implement an attack with the same amplitude against the vehicle with a backdoor. Table 3 presents the ER. While the clean case yielded 0%, the backdoored PID leads to 61.99% lane invasion, and the backdoored LQR to 43.03%.

Takeaway. The backdoor can create unsafe situations, such as crashes in the ACC scenario and lane invasions in the LKC. In contrast, when both systems are free of backdoors, the same malicious trigger fails to create the unsafe situations. This confirms that unsafe situations arise specifically due to the presence of the backdoor.

Case Studies. To explain how the controller fails due to the backdoor, we provide examples of how the attack works in both applications. For simplicity, we present the results for the LQR controller, as the results for the PID control are similar.

Case Study for ACC. We show the time-space diagram in Figure 6a, the velocity of the vehicles in Figure 6b, and images from CARLA in Figure 6c. The ACC distance between vehicles is 15 m, the maximum acceleration of the attacker is $a = 2.5\,\text{m/s}^2$, and the vehicles travel at 72 km/h. We also consider that the vehicles arrived at a steady state before the attack began.

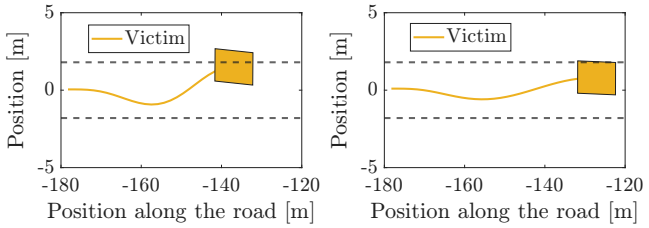
Figure 6b shows an example of a successful attack. The attacker oscillates with the frequency $\omega_{attack} = 0.55$ rad/s, obtained after running the offline section of BADCONTROL. The attacker begins the attack at $t = 0$ s by decelerating (D1). The



(a) Time-space diagram. (b) Velocity.

(c) CARLA example.

Figure 6: Adaptive cruise control time-series; a) time-space diagram, b) the vehicles' velocity, and c) photos from CARLA.



(a) Backdoored (b) Non-backdoored

Figure 7: LKC time response using the same road patch. Horizontal lines are the lane division.

blue vehicle, a non-player character (NPC), follows a similar behavior, as it uses an ACC controller that maintains the distance from the attacker at the same velocity. Later, when the attacker accelerates at $t = 5.5$ s (A1), the victim fails to accelerate and continues to brake for some instants due to the backdoor. At $t = 7$ s, the victim finally begins to accelerate, but the attacker decelerates again at $t = 11$ s (D2). The victim fails to decelerate and speeds up to 100 km/h. At $t = 12$ s, the victim finally slows down, but the remaining distance and time are insufficient to avoid a collision, and the victim collides with the NPC. The speed difference between the vehicles before the crash is 19.48 km/h. Meanwhile, the attacker is not involved in the crash, thus they can continue on the road without damage.

Case study for LKC. Figure 7a shows an attack example in which the victim uses the backdoored LKC. At some point, the attacker compromises the vehicle sensor by, for example, placing dirty patches on the road. This attack modifies the readings received by the controller regarding the center of the

Table 4: Computation time with different methods to compute the gradient for the LQR controller.

System	Implicit diff. [s]	Numerical diff. [s] Non-parallel	Numerical diff. [s] Parallel
ACC	**9.75**	448.11	150.87
LKC	**5.14**	292.35	65.00

lane. Due to the attack, the victim begins to move on the road until, eventually, they oscillate with such magnitude that they leave their lane. Figure 7b shows the vehicle response to the same attack when the controller does not have a backdoor. In this case, the victim's car moves within the lane, even if the attacker implements the same attack. Consequently, we can conclude that the attacker can create unsafe situations by deploying the backdoored controller.

Algorithm Efficiency. The offline phase of the BADCONTROL relies on gradient computation. We use implicit differentiation to compute the gradient, choosing it over numerical methods based on Euler approximation.

To compare these alternatives, we run Algorithm 1 using both numerical and implicit differentiation for our case studies. Table 4 presents the computation times for these two methods. We found that implicit differentiation is significantly faster. Even with concurrent and parallel computation using 24 threads for numerical differentiation, implicit differentiation was almost 15 times faster for the ACC system and 12 times faster for the LKC.

8.2 Attack Robustness

In real-world scenarios, it might be difficult to create a precise acceleration and deceleration maneuver that perfectly matches the trigger frequency. We therefore study whether an attacker can still trigger the backdoor without an exact trigger design.

To simulate an attacker's inability to execute the exact trigger maneuver in the ACC system, we add random noise to the attacker's control action, so that: $\tilde{u}_k = u_k + w_k$, where $w_k \sim U(-\delta, \delta)$ is uniformly distributed random noise with parameter δ. This noise affects the attacker's ability to produce the ideal trigger.

We simulate the attack using this noisy trigger, with a maximum acceleration of $a = 5$ m/s^2. Figure 8 shows the number of crashes for each noise intensity, normalized by the number of crashes from the noise-free maneuver. Our results demonstrate that for $\delta = 0.6$, the number of crashes after the adversary triggers the backdoor does not change. This indicates that with this level of noise, the attacker is still able to trigger the backdoor and make the victim's vehicle crash. Even with a larger noise ($\delta = 1$), we see that the attacker is able to produce 60% of the crashes than using the clean maneuver.

We note that an attacker can trigger the backdoor with a distorted maneuver, but only if the maneuver matches a specific

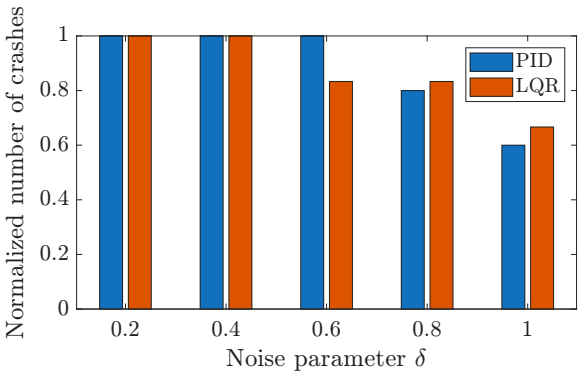
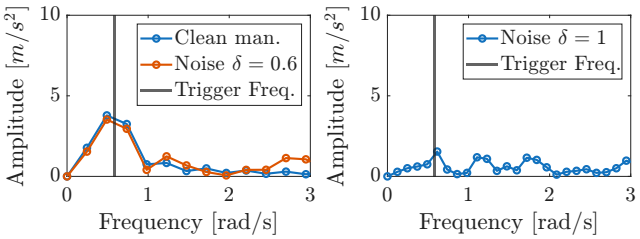


Figure 8: Number of crashes when the attacker uses the noisy maneuver to trigger the backdoor, normalized by the number when the attacker uses the clean maneuver in the ACC system.



(a) Successful attacks (b) Unsuccessful attack

Figure 9: Fast Fourier transform of the attacker's maneuver velocity for attacks with different noise levels.

pattern. Specifically, we study the frequency of the attacker's maneuver velocity using the fast Fourier transform (FFT). Figure 9a shows the frequency spectra of the maneuvers that resulted in a crash between the attacker and the NPC. Those maneuvers have the most power on the trigger frequency. In contrast, Figure 9b shows that these maneuvers have a lower power on the trigger frequency when noise increases. Consequently, the attacker cannot cause a crash. We conclude that our attack is robust; the adversary does not need to inject exactly the signal in Equation (10) and still be able to produce a crash, as long as the maneuver has enough power at the intended frequency.

8.3 Attack Stealthiness

A backdoored controller should preserve vehicle safety in the absence of a trigger, making its vulnerability difficult to detect under normal conditions. To evaluate this stealth property, we test whether existing analysis tools can uncover the hidden weakness in the poisoned controller. Specifically, we employ Acero [45], a state-of-the-art falsification framework for autonomous vehicles that systematically searches for adversarial driving maneuvers that create dangerous conditions.

We use a scenario as in Figure 4 where Acero controls the attacker's vehicle (the red vehicle) and attempts to cause

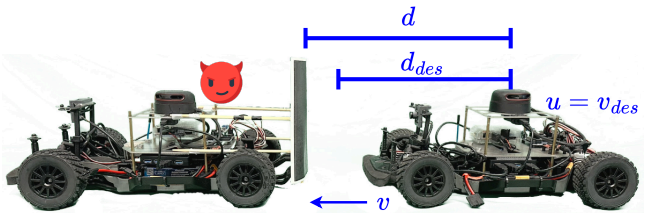


Figure 10: Experimental setup. The car on the left is the attacker that will follow a maneuver to trigger the backdoor.

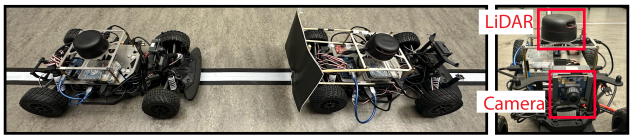


Figure 11: Robots following a straight lane and sensors.

a crash between the victim (the yellow vehicle) and a non-player character (NPC) vehicle (the blue vehicle). To achieve this, Acero defines a robustness metric based on the time to collision (TTC), which measures how close the victim is to colliding with other vehicles. In this specific scenario, it uses the TTC between the blue and yellow cars.

We denote the victim vehicle by v and the NPC by n. The position and velocity of the victim at time instant k are $p_k^{(v)}$ and $v_k^{(v)}$, respectively, and those of the NPC are $p_k^{(n)}$ and $v_k^{(n)}$. The TTC is defined as:

$$\mathrm{TTC}_k(v,n) = \frac{p_k^{(v)} - p_k^{(n)}}{v_k^{(v)} - v_k^{(n)}}.$$

At each time step k, Acero searches for the attacker's action that minimizes the TTC over the space of possible inputs. It does this by selecting pairs of throttle and braking actions, simulating the next step $(k+1)$, and then greedily choosing the action pair producing the minimum TTC. This process repeats until a crash is found or the simulation ends.

We executed Acero for over 15 hours per controller with various initial conditions, for a total of 30 hours. In 30 separate runs per controller, it found no crashes for the LQR controller and only one crash for the PID controller. This failure occurs because Acero's greedy approach always tries to minimize the TTC, while our attack strategy requires increasing the TTC for a period before decreasing it rapidly to cause a crash.

8.4 Robotic Vehicle Implementation

In the previous sections, we demonstrated the effectiveness of BADCONTROL in a high-fidelity simulator. We now show that the attack can be effective in real-world conditions by testing BADCONTROL on the robotic vehicles shown in Figure 10. Our evaluation examines whether the backdoored controller meets the two core objectives: (1) maintaining safety in the absence of the trigger, and (2) inducing a crash when the trigger is activated.

Table 5: Attack Success Rate (ASR) for the robotic vehicles.

Attacker maximum velocity change $[m/s]$	ASR [%]	
	No-backdoor	BADCONTROL
0.25	0.00	50.00
0.50	0.00	100.00
0.75	0.00	100.00

Experiment Setup. As a simplified ACC scenario in CARLA, we implement a two-vehicle setup in our lab, with one robotic vehicle acting as the attacker and the other as the victim. The robots implement a lateral control using a camera to follow a straight path, as shown in Figure 11. Another controller modulates the desired velocity v_{des} for the vehicle to keep the desired distance d_{des} with the vehicle in front. This controller receives the distance between the robots using a LiDAR.

Model identification. To create the backdoor, the attacker needs a model of the vehicle dynamics. We model the vehicle with two variables, distance d and velocity v, while the input is the desired velocity v_d. We run the vehicle at different velocities and obtain the distance with respect to the vehicle in front. We then run system identification, following Appendix C.

Backdoor creation and trigger. We attack the controller that modulates the velocity v_des to keep the desired distance d_des. With the model, we use BADCONTROL to obtain a controller with the backdoor and install it in the victim's vehicle. We find that the attacker has to use a frequency of $\omega_{attack} = 0.70\,\mathrm{rad/s}$.

To trigger the backdoor, the attacker can perform a maneuver in front of the vehicle's victim. The attacker will change their velocity to trigger the backdoor and generate a crash.

ASR. We evaluate the attack success rate (ASR), defining an attack as successful whenever the attacker induces the victim to collide. We perform experiments with the robots operating at 1 m/s and desired distance d_{des} as 0.7 m, 0.9 m, 1.2 m and 1.5 m. The attacker attempts to trigger the backdoor using maneuvers at three different velocities: $a = 0.25\,\mathrm{m/s}$, $0.5\,\mathrm{m/s}$, and $0.75\,\mathrm{m/s}$. In total, we run three experiments per attack.

Table 5 presents the ASR for each maximum velocity a. The ASR increases when the attacker performs a more aggressive maneuver. This again shows that the attacker can use the backdoor to create unsafe situations.

Crucially, the attack remains stealthy: no collisions occur in the absence of the trigger, while crashes can be reliably induced once the trigger is applied. These results demonstrate that BADCONTROL not only succeeds in simulation, but also transfers to physical systems, satisfying both attack objectives in a real-world setting.

Summary. Our experiments demonstrate that BADCONTROL

Table 6: Maximum frequency gain (H_∞ norm) [dB].

		BADCONTROL	
		w/o regularization	w/ Regularization
ACC	PID	3.3	0.93
	LQR	14.31	14.31
LKC	PID	7.96	3.67
	LQR	5.44	0

is a high-impact (up to 100% ASR), reliable (robust to noise), and difficult-to-detect vulnerability, confirmed by its successful transfer to physical robotic vehicles and its ability to evade the state-of-the-art robustness-guided vulnerability discovery framework, Acero [45].

9 Countermeasures

Sensitivity Reduction via Regularization. Regularization is a standard method to mitigate the sensitivity to adversarial perturbations in training data [58]. The central idea is to penalize large parameter values, thereby encouraging smoother solutions that are less responsive to fine-grained variations. In this way, regularization functions as a baseline defense strategy that aims to suppress data manipulations.

We apply existing regularization techniques for both PID and LQR settings [16, 20]. For PID, the controller is identified directly from input-output data through adjustment of the PID gains K_p, K_i, K_d in Eq. (2). Imposing penalties on these parameters restricts their magnitude [20] and thereby reduces the likelihood that adversarial perturbations induce sharp resonances. For LQR, we incorporate regularization during the parameter tuning stage [16], with the expectation that such smoothing limits the effect of adversarial interference.

Table 6 presents the maximum frequency gain of different controllers under BADCONTROL, both without and with regularization. We vary the regularization weight λ over the range 0.01–100 and report the minimum gain achieved. For PID controllers, regularization substantially reduces the peak gain and suppresses amplification at the resonance frequency. A similar effect is observed for the LQR controller in the LKC. In contrast, for the LQR controller in the ACC, regularization provides little improvement and does not effectively mitigate the attack. These results indicate that regularization may be ineffective for the LQR controller in certain scenarios. This result can be explained by the following property.

Property 1 *Let λ denote the regularization weight. For PID control, $K_p, K_i, K_d \to 0$ as $\lambda \to \infty$ for any perturbation $\Delta \in \mathcal{D}$. For LQR control, we have $K \to K_{\mathrm{LS}}$ as $\lambda \to \infty$ where K_{LS} is the least-squares LQR controller, given in Appendix A.2.*

Property 1 claims that, in PID design, the controller parameters are directly penalized, i.e., the sensitivity to the adversarial perturbation becomes zero in the extreme case. On the other hand, in LQR design, regularization influences intermediate quantities in the learning process rather than the final controller κ in Eq. (3) directly (See Appendix A). Although these intermediate values are penalized, the subsequent optimization step can still construct controllers with sharp amplification at specific frequencies. Consequently, adversarial perturbations can remain harmful despite regularization.

Robust Optimization for LQR. The limited effectiveness of standard regularization highlights the need for defenses that directly constrain the closed-loop behavior of the controller. Our key insight is that the clean trajectory lies within a neighborhood of the collected data, even if the latter is slightly poisoned. By ensuring that the controller maintains bounded closed-loop characteristics for all systems consistent with such perturbations, we make it insensitive to backdoor embeddings. We formalize this defense using a robust optimization framework [8], where the synthesis problem is reformulated to minimize nominal cost while simultaneously enforcing constraints that hold under every admissible perturbation:

$$\begin{aligned} \min_K \quad & L(K) \\ \text{s.t.} \quad & F(K(Y+\Delta Y, U+\Delta U)) \le \gamma, \ \forall \Delta = (\Delta Y, \Delta U) \in \mathcal{D} \end{aligned} \tag{11}$$

where $L(K)$ denotes an objective function that evaluates control performance for the nominal data, $F(K(Y+\Delta Y, U+\Delta U))$ represents the closed-loop characteristic given by the H_∞ norm as in (9a), $\gamma > 0$ determines a guaranteed level of the closed-loop characteristic.

The main challenge in the proposed robust optimization framework lies in the nature of its constraints. Since the defender must account for all possible adversarial perturbations within a prescribed set $\mathcal{D}$, the resulting formulation becomes a semidefinite program (SDP) with infinitely many constraints. Thus, the direct formulation is computationally intractable and cannot be addressed with standard optimization tools.

To address this, we employ a recent result from control theory known as the matrix S-lemma [30, 31, 48]. This technique provides an exact reformulation of the infinitely many constraints into a single linear matrix inequality, resulting in the following optimization problem

$$\begin{aligned} \min_{Z,V,\nu,\mu} \quad & \hat{L}(Z) \\ \text{s.t.} \quad & \begin{bmatrix} Z-\mu\gamma I & 0 & 0 & 0 \\ 0 & 0 & \begin{bmatrix} Z \\ V \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} Z^\top & V^\top \end{bmatrix} & Z & Z^\top \\ 0 & 0 & Z & I \end{bmatrix} - \begin{bmatrix} \nu N & 0 \\ 0 & 0 \end{bmatrix} \succeq 0, \\ & \begin{bmatrix} Z & Z^\top \\ Z & I \end{bmatrix} \succ 0, \quad \nu \ge 0, \quad \mu > 0 \end{aligned} \tag{12}$$

where the parameters are given in Appendix D, and $\succ$ ($\succeq$) means that the matrix is positive definite (semidefinite).

In this way, the matrix S-lemma bridges the gap between theoretical robustness guarantees and practical implementa-

tion for optimization over symmetric cones [46]. The controller is designed as $K = VZ^{-1}$. We can give a formal guarantee of the maximum frequency gain against adversarial perturbation as long as the size of Δ is less than ε.

Property 2 *By designing a robust controller with the parameter K through Eq. (12), we can guarantee that* $\|G_K\|_\infty < \gamma$ *for any* $\|\Delta\|_{max} \leq \varepsilon$.

See Appendix D for technical details and experiments showing the effectiveness of this defense.

10 Limitations and Future Work

Evasion Maneuver of Victims. We consider that the victims deploy the controllers without an evasive technique. A victim could implement additional strategies to the control system, such as anomaly detection [22,35,39,44], to partially mitigate the effects of the attack. We leave this possible defense study as a path for extending the work.

Extension to Multiple Victims. While our work focuses on single-agent scenarios, BADCONTROL can extend to systems with multiple victims. The extension involves simultaneously maximizing the H_∞ norm across victims while considering the interactions between their dynamics and objective functions. We leave the exact formulation as future work.

Extension to other Domains. BADCONTROL generalizes to any system relying on classical data-derived controllers (robotics, industrial automation). The poisoning threat model remains consistent, but trigger activation differs; closed environments require triggers targeting internal sensor measurements, whereas vehicles allow external physical manipulation.

11 Conclusion

We introduce BADCONTROL, a backdoor attack against controllers that uses physical triggers to cause safety violations. We demonstrate the effectiveness of BADCONTROL in two systems: ACC and LKC. In both applications, an attacker can trigger safety violations, including vehicle crashes. To counter such attacks, we study two defense mechanisms: one based on regularization and another based on robust optimization. These defenses operate during the controller's parameter tuning phase to create a controller that is resilient to data poisoning. BADCONTROL is a significant step forward in the analysis of AV control systems, but it requires further investigation. Future work will explore the mathematical properties of the optimization problems, such as convexity, to establish guarantees of convergence to an optimal solution. We will also explore the attack's potential on different controllers and investigate other triggers, such as sensor manipulation, and more advanced defense mechanisms that can offer robust guarantees against a broader range of backdoor attacks.

Acknowledgments

This work was supported in part by JSPS KAKENHI Grant Number JP24K17296. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-24-1-0015, by the National Science Foundation (NSF) under grant numbers CNS-2144645, IIS-2229876 (in part by funds provided by the Department of Homeland Security and IBM), and by the National Center for Transportation Cybersecurity and Resiliency (TraCR), a U.S. Department of Transportation National University Transportation Center headquartered at Clemson University, Clemson, South Carolina, USA (USDOT Grant #69A3552344812). Any opinions, findings, conclusions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of TraCR, and the U.S. Government assumes no liability for the contents or use thereof.

Ethical Considerations

We investigate a theoretical class of data-poisoning attacks in control systems analogous to widely studied ML backdoors. Our research aligns with vulnerability discovery and proposes countermeasures to address them. Our goal is to inform the control and security communities about this unaddressed security gap in widely deployed systems across both the automotive and general control domains.

Although this work primarily addresses theoretical attacks, practical applications and extension of this work have the potential to impact a range of stakeholders, including: (1) control stack developers, (2) control systems and security researchers, (3) control systems end-users, and (4) policymakers. Below, we detail how we addressed ethical risks at each stage of the research and our reflections on the broader impact of the publication.

Beginning and Conducting the Research: During the active research phase, our primary ethical focus was the safety of the research team and the physical testbed. To protect them, we initially conducted experiments in purely simulated environments. Once we understood the effects of the backdoor, we performed the physical tests in a closed, controlled environment. We operated the robots at low speeds in order to reduce the risk of harm to researchers or equipment in the event of unexpected behavior.

Decision to Publish and Stakeholder Impact: We recognize that by publishing this research, we introduce potential risks to control stack developers, automotive manufacturers, and control-system end users by revealing a mechanism for backdoor attacks. However, we believe that because this will allow control developers to verify their datasets and create new defenses, the potential benefits of publishing this work outweigh the risks it introduces.

To mitigate the attack risks, we identified two suggestions for stakeholders. **For control stack developers**: we empha-

size that, although our work is theoretical, developers should implement additional measures to audit data sources because they are at risk when tuning controllers with unverified data. **For policymakers**: we suggest that policymakers use these findings to develop guidelines that encourage stronger data provenance checks.

Mitigation and Responsible Disclosure: We prioritized designing a defense against the attack before submitting this work. After completing the study, we adhered to principles of responsible vulnerability disclosure. We prepared a technical report summarizing our findings and have started sharing it with the stakeholders across the automotive sector[1]. This ensures the relevant stakeholders have advanced notice to develop and deploy mitigations.

Open Science

We commit to releasing BADCONTROL as an open-source project under the MIT license. The link to the code is

https://zenodo.org/records/17932676

for public use. We provide three main parts: 1) the files to create the controller with a backdoor using Matlab, 2) the implementation on the Carla simulator, and 3) code to deploy the attack on the physical robots (requires the real hardware).

References

[1] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 2017.

[2] Brian DO Anderson and John B Moore. *Optimal Control: Linear Quadratic Methods*. Courier Corporation, 2007.

[3] Apollo auto: How to tune control parameters. https://daobook.github.io/apollo/docs/howto/how_to_tune_control_parameters.html, 2022. Accessed: 2025-08-27.

[4] Karl Johan Åström and Tore Hägglund. The future of PID control. *Control Engineering Practice*, 2001.

[5] Autoware universe: PID longitudinal controller. https://autowarefoundation.github.io/autoware_universe/main/control/autoware_pid_longitudinal_controller/, 2025. Accessed: 2025-08-27.

[1] Companies and Developers (e.g., Waymo, Zoox, Ford AV and Mobility, Nuro, May Mobility, Daimler Truck, Motional, Cruise, and Waabi, Autoware, OpenPilot); Organizations (e.g., National Association of City Transportation Officials, The Autonomous Vehicle Computing Consortium, and Autonomous Vehicle Industry Association); and Policymakers (e.g., NHTSA and SAE-ITC).

[6] Yang Bai, Gaojie Xing, Hongyan Wu, Zhihong Rao, Chuan Ma, Shiping Wang, Xiaolei Liu, Yimin Zhou, Jiajia Tang, Kaijun Huang, et al. Backdoor attack and defense on deep learning: A survey. *IEEE Transactions on Computational Social Systems*, 2024.

[7] BARC: Berkeley Autonomous Race Car. https://github.com/MPC-Berkeley/barc, 2015. Accessed: 2025-08-27.

[8] Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization–methodology and applications. *Mathematical Programming*, 2002.

[9] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 3rd edition, 2016.

[10] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. *Advances in Neural Information Processing Systems*, 2022.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] William N. Caballero, Mark Friend, and Erik Blasch. Adversarial machine learning and adversarial risk analysis in multi-source command and control. In *Proceedings of SPIE 11756, Signal Processing, Sensor/Information Fusion, and Target Recognition*, volume 11756, page 117560L. SPIE, 2021.

[13] California Department of Motor Vehicles. California driver's handbook – safe driving. https://www.dmv.ca.gov/portal/handbook/california-driver-handbook/safe-driving-cont2/, 2024. Accessed: 2025-08-27.

[14] Marco C Campi, Andrea Lecchini, and Sergio M Savaresi. Virtual reference feedback tuning: A direct method for the design of feedback controllers. *Automatica*, 2002.

[15] Jing Cui, Yufei Han, Yuzhe Ma, Jianbin Jiao, and Junge Zhang. Badrl: Sparse targeted backdoor attack against reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[16] Florian Dörfler, Pietro Tesi, and Claudio De Persis. On the certainty-equivalence approach to direct data-driven LQR design. *IEEE Transactions on Automatic Control*, 2023.

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Annual Conference on Robot Learning*, 2017.

[18] Michael Everett, Björn Lütjens, and Jonathan P How. Certifiable robustness to adversarial state uncertainty in deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[19] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] Simone Formentin and Alireza Karimi. Enhancing statistical performance of data-driven controller tuning via L2-regularization. *Automatica*, 2014.

[21] Bruce A Francis. *A Course in H_∞ Control Theory*. Springer, 1987.

[22] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

[23] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[24] Chen Gong, Zhou Yang, Yunpeng Bai, Jieke Shi, Junda He, Kecen Li, Bowen Xu, Sinha Arunesh, Xinwen Hou, David Lo, et al. Baffle: Hiding backdoors in offline reinforcement learning datasets. In *IEEE Symposium on Security and Privacy (S&P)*, 2024.

[25] Wassim M Haddad and VijaySekhar Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-based Approach*. Princeton University Press, 2008.

[26] Xingshuo Han, Guowen Xu, Yuan Zhou, Xuehuan Yang, Jiwei Li, and Tianwei Zhang. Physical backdoor attacks to lane detection systems in autonomous driving. In *ACM International Conference on Multimedia*, 2022.

[27] Tran Huynh, Anh Tran, Khoa D Doan, and Tung Pham. Data poisoning quantization backdoor attack. In *European Conference on Computer Vision*, pages 38–54. Springer, 2024.

[28] Insurance Institute for Highway Safety. Speed limit laws. https://www.iihs.org/research-areas/speed/speed-limit-laws, August 2025. Accessed: 2025-08-27.

[29] Stephen Bassi Joseph, Emmanuel Gbenga Dada, Afeez Abidemi, David Opeoluwa Oyewola, and Ban Mohammed Khammas. Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems. *Heliyon*, 2022.

[30] Taira Kaminaga and Hampei Sasahara. Data informativity for quadratic stabilization under data perturbation. In *American Control Conference*, 2025.

[31] Taira Kaminaga and Hampei Sasahara. Data informativity under data perturbation. *arXiv preprint arXiv:2505.01641*, 2025.

[32] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. TrojDRL: Evaluation of backdoor attacks on deep reinforcement learning. In *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[33] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[34] Jan R Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, 2019.

[35] Raymond Muller, Yanmao Man, Ming Li, Ryan Gerdes, Jonathan Petit, and Z. Berkay Celik. VOGUES: Validation of object guise using estimated components. In *USENIX Security Symposium*, 2023.

[36] Fabio Pasqualetti, Sandro Zampieri, and Francesco Bullo. Controllability metrics, limitations and algorithms for complex networks. *IEEE Transactions on Control of Network Systems*, 2014.

[37] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 2017.

[38] Mozhgan Pourkeshavarz, Mohammad Sabokrou, and Amir Rasouli. Adversarial backdoor attack by naturalistic data poisoning on trajectory prediction in autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[39] Raul Quinonez, Jairo Giraldo, Luis Salazar, Erick Bauman, Alvaro Cardenas, and Zhiqiang Lin. SAVIOR: Securing autonomous vehicles with robust physical invariants. In *29th USENIX security symposium (USENIX Security 20)*, pages 895–912, 2020.

[40] Hampei Sasahara. Adversarial destabilization attacks to direct data-driven control. *arXiv preprint arXiv:2507.14863*, 2025.

[41] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Dirty road can attack: Security of deep learning based automated lane centering under Physical-World attack. In *USENIX Security Symposium*, 2021.

[42] Shawn Shan, Arjun Nitin Bhagoji, Haitao Zheng, and Ben Y Zhao. Poison forensics: Traceback of data poisoning attacks in neural networks. In *USENIX Security Symposium*, 2022.

[43] Robert E Skelton, Tetsuya Iwasaki, and Dimitri E Grigoriadis. *A Unified Algebraic Approach to Control Design*. Routledge, 2017.

[44] Ruoyu Song, Muslum Ozgur Ozmen, Hyungsub Kim, Antonio Bianchi, and Z Berkay Celik. Enhancing llm-based autonomous driving agents to mitigate perception attacks. *arXiv preprint arXiv:2409.14488*, 2024.

[45] Ruoyu Song, Muslum Ozgur Ozmen, Hyungsub Kim, Raymond Muller, Z Berkay Celik, and Antonio Bianchi. Discovering adversarial driving maneuvers against autonomous vehicles. In *USENIX Security Symposium*, 2023.

[46] Jos F Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 1999.

[47] U.S. Department of Transportation, Federal Highway Administration. Highway statistics 2023: Arterial and collectors length by lane width (table hm-53). Technical report, Office of Highway Policy Information, 2023. Accessed: 2025-08-27.

[48] Henk J van Waarde, M Kanat Camlibel, and Mehran Mesbahi. From noisy data to feedback controllers: Nonconservative design via a matrix S-lemma. *IEEE Transactions on Automatic Control*, 2020.

[49] Antonio Visioli. *Practical PID control*. Springer, 2006.

[50] Hang Wang, Zhen Xiang, David J Miller, and George Kesidis. MM-BD: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic. In *IEEE Symposium on Security and Privacy (S&P)*, 2024.

[51] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. Backdoorl: Backdoor attack against competitive reinforcement learning. *arXiv preprint arXiv:2105.00579*, 2021.

[52] Yue Wang, Esha Sarkar, Saif Eddin Jabari, and Michail Maniatakos. On the vulnerability of deep reinforcement learning to backdoor attacks in autonomous vehicles. In *Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Use Cases and Emerging Challenges*. Springer, 2023.

[53] Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *IEEE Transactions on Information Forensics and Security*, 2021.

[54] Ming Xu, Timothy L Molloy, and Stephen Gould. Revisiting implicit differentiation for learning problems in optimal control. *Advances in Neural Information Processing Systems*, 2024.

[55] Yuan Xu, Xingshuo Han, Gelei Deng, Jiwei Li, Yang Liu, and Tianwei Zhang. SoK: rethinking sensor spoofing attacks against robotic vehicles from a systematic view. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2023.

[56] Zhaoyuan Yang, Naresh Iyer, Johan Reimann, and Nurali Virani. Design of intentional backdoors in sequential models. *arXiv preprint arXiv:1902.09972*, 2019.

[57] Wen Yin, Jian Lou, Pan Zhou, Yulai Xie, Dan Feng, Yuhua Sun, Tailai Zhang, and Lichao Sun. Physical backdoor: Towards temperature-based backdoor attacks in the physical world. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[58] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[59] Kaiyuan Zhang, Siyuan Cheng, Guangyu Shen, Guanhong Tao, Shengwei An, Anuran Makur, Shiqing Ma, and Xiangyu Zhang. Exploring the orthogonality and linearity of backdoor attacks. In *IEEE Symposium on Security and Privacy (S&P)*, 2024.

[60] Xinwei Zhang, Aishan Liu, Tianyuan Zhang, Siyuan Liang, and Xianglong Liu. Towards robust physical-world backdoor attacks on lane detection. In *ACM International Conference on Multimedia*, 2024.

[61] Yan Zhang, Yi Zhu, Zihao Liu, Chenglin Miao, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Towards backdoor attacks against LiDAR object detection in autonomous driving. In *ACM Conference on Embedded Networked Sensor Systems*, 2022.

[62] Junhao Zheng, Chenhao Lin, Jiahao Sun, Zhengyu Zhao, Qian Li, and Chao Shen. Physical 3D adversarial attacks against monocular depth estimation in autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

A Controller Design Algorithms

A.1 PID Controller

The PID controller remains one of the most widely employed control strategies in practice. Among various approaches for parameter tuning, the Virtual Reference Feedback Tuning (VRFT) method [14] has established itself as a standard. In its regularized version [20], the controller gains K_p, K_i, K_d are obtained by solving a regularized quadratic optimization

$$\min_{K_p,K_i,K_d} \|F_{\text{pre}}(T_{\text{des}} - T_{K_p,K_i,K_d})r\|_2^2 + \lambda(K_p^2 + K_i^2 + K_d^2)$$

where $\|\cdot\|_2$ denotes the Euclidean norm, $\lambda \geq 0$ denotes a regularization parameter, r denotes the reference signal, F_{pre} denotes a prefilter, T_{des} denotes the desired transfer function and T_{K_p,K_i,K_d} denotes the actual closed-loop transfer function associated with the controller parameters. The second term explicitly penalizes the magnitude of the controller parameters, thereby preventing overly aggressive tuning. In the limiting case $\lambda \to \infty$, the optimization suppresses all nonzero parameters, effectively nullifying the controller sensitivity to perturbations as claimed by Property 1.

A.2 LQR Controller

The LQR controller is the benchmark controller in the control community [2]. It supposes state feedback, i.e., $y_k = x_k$ and minimizes the cost $c = \sum_{k=0}^{\infty} x_k Q x_k^\top + u_k R u_k^\top$, where Q and R are positive definite tuning matrices of appropriate size. To design the controller, let us denote the trajectories $Y_0 = [y_0, y_1, ..., y_{T-1}]$, $Y_1 = [y_1, y_2, ..., y_T]$, and $U_0 = [u_0, u_1, ..., u_T]$.

Dörfler *et al.* [16] have developed an LQR controller design method through the following optimization problem using the training data:

$$\begin{aligned} \min_{P \succeq I, K, Z} \quad & \text{tr}(QP) + \text{tr}(K^\top R K P) + \lambda ||\Pi Z|| \\ \text{s.t.} \quad & Y_1 Z P Z^\top Y_1^\top - P + I \preceq 0, \\ & \begin{bmatrix} K^\top & I \end{bmatrix}^\top = W_0 Z, \end{aligned} \tag{13}$$

where $\text{tr}(\cdot)$ is the matrix trace, $\lambda \geq 0$ is a regularization parameter, $W_0 = \begin{bmatrix} U_0^\top & Y_o^\top \end{bmatrix}^\top$, $\Pi = I - W_0^\dagger W_0$, $W_0^\dagger$ is the right inverse of W_0, and $\succeq$ ($\preceq$) means that the matrix is positive (negative) semi-definite. The study has shown that K converges to the least-squares LQR controller designed through the optimization problem

$$\begin{aligned} \min_K \quad & c \\ \text{s.t.} \quad & x_{k+1} = (A^* + B^* K) x_k, \\ & [A^* \ B^*] = \text{argmin}_{A,B} \|Y_1 - A Y_0 - B U_0\|_{\text{F}}, \end{aligned}$$

where $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm. In other words, at the limit $\lambda \to \infty$, the LQR controller (13) is equivalent to the model-based LQR controller where the model is identified using the least square method. Hence, the controller parameter K does not necessarily converge to zero as claimed by Property 1.

B Gradient Computation via Implicit Differentiation

We first review basic matrix calculus [34] necessary for implicit differentiation techniques. We define the derivative of a matrix-valued function $F(X) = [f_1 \ \cdots \ f_m] \in \mathbb{R}^{n \times m}$ with respect to the matrix variable $X = [x_1 \ \cdots \ x_q] \in \mathbb{R}^{p \times q}$ by

$$\frac{dF}{dX} := \frac{d\text{vec}(F)}{d\text{vec}(X)} = \begin{bmatrix} \frac{df_1}{dx_1} & \cdots & \frac{df_1}{dx_q} \\ \vdots & & \vdots \\ \frac{df_m}{dx_1} & \cdots & \frac{df_m}{dx_q} \end{bmatrix} \in \mathbb{R}^{nm \times pq}.$$

The definition preserves fundamental properties of differentiation for vector-valued functions including the chain rule, which claims that for $H(X) := G(F(X))$ we have

$$\frac{dH}{dX}(X) = \frac{dG}{dY}(Y)\Big|_{Y=F(X)} \frac{dF}{dX}(X).$$

Implicit differentiation is a widely used technique in the field of machine learning and control theory [1, 10, 54] to compute a derivative of a function whose relationship with its variable is represented through an implicit function. Suppose that we have an implicit function $G(F,X) = 0$ and we want to compute the derivative dF/dX. The chain rule leads to

$$\frac{\partial G}{\partial F}\frac{dF}{dX} + \frac{\partial G}{\partial X} = 0,$$

where the partial derivative with the symbol ∂ is defined in a similar manner. Thus, we can obtain dF/dX by solving the linear equation if we can easily compute $\partial G/\partial F$ and $\partial G/\partial X$. We utilize this technique for computing the gradients necessary for the backdoor embedding.

Consider the objective function (9a) for the LQR controller design. Our objective here is to compute the gradients $\nabla_Y J$ and $\nabla_U J$, which are derived from $\partial J/\partial Y$ and $\partial J/\partial U$. Note that the chain rule leads to

$$\begin{aligned} \frac{\partial J}{\partial Y}(Y,U) &= \frac{dF}{dK}\frac{\partial K}{\partial Y}(Y,U), \\ \frac{\partial J}{\partial U}(Y,U) &= \frac{dF}{dK}\frac{\partial K}{\partial U}(Y,U). \end{aligned}$$

The term dF/dK is easy to compute by numerical differentiation, because it is independent of outcomes of the optimization problem (13). Hence, we consider computing $\partial K/\partial Y$ and $\partial K/\partial U$ using implicit differentiation.

The key observation is that $K(Y,U)$ is given through an optimization problem (13). We adopt the KKT

(Karush–Kuhn–Tucker) condition for semidefinite programming (SDP) [11, Section 5] as an implicit function between the designed controller K and the collected data (Y,U). For demonstration, we consider an abstract SDP described by

$$\begin{aligned} \min_K \quad & L(K,Y) \\ \text{s.t.} \quad & G(K,Y) \succeq 0, \end{aligned}$$

whose Lagrangian is given by

$$\mathcal{L}(K,\Lambda,Y) = L(K,Y) - \operatorname{tr}(G(K,Y)\Lambda)$$

with the Lagrange multiplier $\Lambda \succeq 0$. The optimal solution (K,Λ) satisfies the KKT condition given by

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial K}(K,\Lambda,Y) = 0, \quad \operatorname{tr}(G(K,Y)\Lambda) = 0, \quad \Lambda - \Lambda^{\mathsf{T}} = 0, \\ &G(K,Y) \succeq 0, \quad \Lambda \succeq 0. \end{aligned}$$

We employ only the first three equations as an implicit function by ignoring the primal and dual feasibility conditions [10], and let us denote the aggregated equations by $H(K,\Lambda,Y) = 0$. Then, by the chain rule, we have

$$\frac{\partial H}{\partial K}\frac{dK}{dY} + \frac{\partial H}{\partial \Lambda}\frac{d\Lambda}{dY} + \frac{\partial H}{\partial Y} = 0. \tag{14}$$

We can analytically calculate the terms $\partial H/\partial K$, $\partial H/\partial \Lambda$, and $\partial H/\partial Y$. Therefore, the linear equation (14) can readily be solved with respect to dK/dY and $d\Lambda/dY$. For the specific SDP (13), we can effectively compute $\partial K/\partial Y$ and $\partial K/\partial U$ by following the procedure above.

C System Identification

For our experiments, the attacker needs to create a model of the vehicle physical dynamics. That is, we need to find the parameters A, B, and C in Eq. (5).

To achieve this, we had to solve the system identification problem to create models for the vehicles in CARLA and for the robotic vehicle. Using best practices for selecting a driving control input U, we then collect the sensor values Y corresponding to the physical behavior to the given input.

After we collect enough trajectories Y, U for the CARLA vehicle and the real-world robotic vehicle, we can use system identification to create such a model. We use the following optimization problem to find the parameters,

$$\begin{aligned} \underset{A,B,C}{\arg\min} \quad & \sum_{k=0}^{T} ||y_k - \hat{y}_k||^2 \\ \text{s.t.} \quad & \hat{x}_{k+1} = A\hat{x}_k + Bu_k, \\ & \hat{y}_k = C\hat{x}_k, \\ & \hat{x}_0 = x_0, \end{aligned}$$

and we can solve this problem using gradient-based optimizations. As we do not impose constraints on the decision variables, we can use standard optimization algorithms. In particular, we use `fminsearch` function from Matlab.

D Robust Optimization

Countermeasures for adversarial attacks can be classified into two categories: reactive and proactive strategies [58]. While reactive strategies mainly include the detection of attacks, proactive strategies include robustness enhancement of the designed architecture. Robust optimization is a mathematical framework for proactive defense that can achieve certified performance [18, 33]. Originally, it is a branch of optimization theory that focuses on finding solutions capable of withstanding a degree of uncertainty in the data [8]. This uncertainty is modeled using specific uncertainty sets that characterize the nature and extent of the data perturbations. These sets are typically defined based on the type of uncertainty and a parameter that determines their size. The goal of robust optimization is to obtain solutions that are feasible and well-behaved under any realization in the uncertainty set. For example, consider the standard constrained optimization problem

$$\begin{aligned} \min_x \quad & f(x;d) \\ \text{s.t.} \quad & g(x;d) < 0, \end{aligned}$$

where d is the given data. The corresponding robust optimization formulation is

$$\begin{aligned} \min_x \quad & f(x;d) \\ \text{s.t.} \quad & g(x;d+\delta) < 0, \quad \forall \delta \in \Delta \end{aligned}$$

where Δ is the uncertainty set. The objective of this problem is to select a solution x that minimizes the objective function for the given data and satisfies the constraint across all possible variations of the problem parameters contained within the uncertainty set. Our defense method leverages the robust optimization framework for designing a controller that eliminates possible backdoors.

The works [30, 31] develop an efficient computation method to solve the robust controller design problem (11). We first transform the perturbation set into a form easy to handle. Define

$$\hat{\mathcal{D}} := \{\Delta : \Delta\Delta^{\mathsf{T}} \preceq (2n+m)\varepsilon^2(T+1)I\}.$$

Then $\mathcal{D} \subset \hat{\mathcal{D}}$. This claim can be proven as follows: Denote $\Delta = [\delta_0 \ \cdots \ \delta_T]$. Then $\|\delta_t\|_\infty \le \varepsilon$. Since

$$\frac{\|\delta_t\|}{\sqrt{2n+m}} \le \|\delta_t\|,$$

we have $\|\delta_t\|_2^2 \le (2n+m)\varepsilon^2$. By summing up with respect to t, we have

$$\sum_{t=0}^{T} \|\delta_t\|_2^2 \le (2n+m)T\varepsilon^2,$$

which is equivalent to the inequality.

This inclusion relationship implies that it suffices to consider the perturbation set $\hat{\mathcal{D}}$ instead of $\mathcal{D}$. The works [30, 31]

have shown that the set of admissible system matrices for some $\Delta \in \hat{\mathcal{D}}$ can be characterized by

$$\hat{\Sigma} = \left\{ (A,B) : \begin{bmatrix} I \\ A^\top \\ B^\top \end{bmatrix}^\top N \begin{bmatrix} I \\ A^\top \\ B^\top \end{bmatrix} \succeq 0 \right\}$$

with

$$N = \begin{bmatrix} I \\ \mathbf{X}^\top \end{bmatrix}^\top \Phi \begin{bmatrix} I \\ \mathbf{X}^\top \end{bmatrix}, \quad \Phi = \begin{bmatrix} \bar{\varepsilon} I & 0 \\ 0 & -I \end{bmatrix}$$

where

$$\bar{\varepsilon} = (2n+m)\varepsilon^2(T+1), \quad \mathbf{X} := [Y_1^\top \ -Y_0^\top \ -U_0^\top]^\top.$$

Thus, the original robust optimization problem is reduced to

$$\begin{aligned} \min_K \ & L(K) \\ \text{s.t.} \ & F(K;A,B) < \gamma \quad \forall (A,B) \in \hat{\Sigma}. \end{aligned} \tag{15}$$

An advantage of the representation (15) is that the constraint can be transformed into a tractable form.

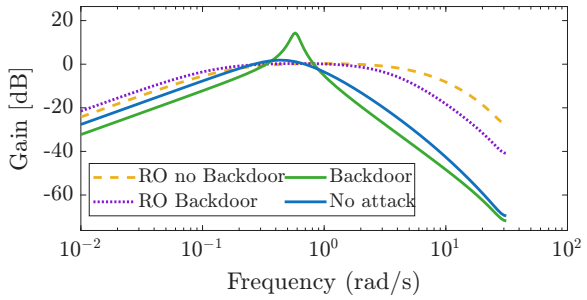


Figure 12: Effectiveness of the Robust Optimization (RO) protection for LQR controllers in the ACC system.

In control theory, it is known that $F(K;A,B) < \gamma$ is satisfied if there exists $P \succ 0$ such that

$$\begin{cases} P - A_K^\top (P^{-1} - \frac{1}{\gamma^2} I)^{-1} A_K - I \succ 0, \\ P^{-1} - \frac{1}{\gamma^2} I \succ 0 \end{cases} \tag{16}$$

where $A_K = A + BK$ as shown in [43, 48]. The matrix S-lemma [30, 31] is given as follows.

Lemma 1 *Let*

$$M^S := \begin{bmatrix} M_{11}^S & M_{12}^S \\ * & M_{22}^S \end{bmatrix}, \quad N^S := \begin{bmatrix} N_{11}^S & N_{12}^S \\ * & N_{22}^S \end{bmatrix}$$

be two symmetric matrices. Assume that

$$M_{22}^S \preceq 0, \quad \ker M_{22}^S \subset \ker M_{12}^S,$$

and there exists Z_0 such that

$$\begin{bmatrix} I & Z_0^\top \end{bmatrix} N^S \begin{bmatrix} I \\ Z_0 \end{bmatrix} \succeq 0.$$

Then the condition

$$\begin{bmatrix} I & Z^\top \end{bmatrix} N^S \begin{bmatrix} I \\ Z \end{bmatrix} \succeq 0 \Rightarrow \begin{bmatrix} I & Z^\top \end{bmatrix} M^S \begin{bmatrix} I \\ Z \end{bmatrix} \succeq 0$$

holds if and only if there exist $\nu^S \geq 0$ and $\mu^S > 0$ such that

$$M^S - \nu^S N^S \succeq \begin{bmatrix} \mu^S I & 0 \\ 0 & 0 \end{bmatrix}.$$

Note that both the sensitivity requirement (16) and $(A,B) \in \hat{\Sigma}$ can be represented as the form in the lemma with $Z^\top = [A \ B]$ by appropriately choosing N^S and M^S. It transforms the infinitely many constraints in (15) into a tractable linear matrix inequality. The resulting optimization problem is represented by (12) where $\mu_\gamma := \mu + 1/\gamma^2$ and $\hat{L}(Z)$ is the cost function with respect to the variable Z transformed from the original cost function $L(K)$. As stated, the transformed problem becomes an SDP with a finite constraint and can readily be solved using existing solvers.

The effectiveness of this defense is illustrated in Figure 12, where we can see that our Robust Optimization (RO) approach mitigates the attempted resonance attack. In addition, we can observe that the frequency response of the new RO LQR controller follows the original response of the LQR controller without the attack very closely.

Table 7: H_∞ norm [dB] for the LKC system with an LQR controller changing the poisoning portion (α) and size (ε).

Poisoning size (ε)	Poisoning portion (α) [%]			
	0	10	50	100
$4 \times 10^{-4} \|\|(X,U)\|\|_{max}$	0	1.03	9.43	10.92
$1 \times 10^{-3} \|\|(X,U)\|\|_{max}$	0	7.96	11.79	14.00

E Poisoning Size vs. Portion of Dataset

Finally, we analyze the effect of the maximum poisoning size (ε in Eq. (6)) and the portion of the dataset poisoned (α in Eq. (8)). Table 7 presents the H_∞ of the LKC system with a backdoored controller as a function of α and ε. In the table, the operator $||\cdot||_{\max}$ denotes the maximum magnitude of a value in the dataset.

Recall that a larger H_∞ indicates a larger effect of the backdoor. We conclude two points. (1) The attacker can create the backdoor by compromising a small portion of the dataset; by poisoning only 10% of the dataset, the attacker achieves an H_∞ norm of 7.96 dB same as the controller in Sec. 8. (2) As the poisoning portion decreases, the attacker needs to increase the maximum poisoning size. For example, if the attacker compromises 10% of the dataset, they can achieve an H_∞ norm of 1.03 dB when $\varepsilon = 4 \times 10^{-4} ||(X,U)||_{max}$. For a stronger attack, the adversary needs to increase the poisoning magnitude to $\varepsilon = 1 \times 10^{-3} ||(X,U)||_{max}$.