2025 28th International Conference on Computer and Information Technology (ICCIT)
19–21 December 2025, Cox's Bazar, Bangladesh

CPS-Guard: Defensive Architectures for Securing Large Language Models in Cyber-Physical Systems

Md. Mazharul Islam
Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
mazharul.islam1@northsouth.edu

Mubasshir Ahmed
Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
mubasshir.ahmed@northsouth.edu

Niaz Ashraf Khan
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
niaz.ashraf@bracu.ac.bd

***Abstract*—Cyber-physical systems (CPS) are increasingly adopting large language models (LLMs) for log analysis, operator support, and anomaly triage. While LLMs enable powerful reasoning and flexible natural-language interaction, their probabilistic and opaque behavior introduces new attack surfaces, including prompt injection, adversarial manipulation, and privacy leakage. Existing CPS defenses largely assume deterministic models and do not account for LLM-specific threats, whereas current LLM safety work rarely targets real-time, safety-critical control environments. To bridge this gap, we propose *CPS-Guard*, a defense-in-depth architecture for LLM-enabled CPS that operates under advisory-only semantics. CPS-Guard combines secure prompt handling, explicit threat modeling, a deterministic safety validator with policy and bounds checking, and anomaly-based veto over plant telemetry. We implement CPS-Guard as middleware between CPS controllers and an aligned LLM, and evaluate it on 300 synthetically generated adversarial inputs spanning prompt injection, adversarial perturbations, and poisoning-style triggers. Compared to an unprotected LLM advisory baseline, CPS-Guard reduces failure rates by more than 85% while incurring only 10–18% additional latency across small to large CPS workloads. These results indicate that system-level hardening can make LLM-assisted CPS substantially more robust without violating real-time constraints.**

***Index Terms*—Cyber-Physical Systems (CPS), Large Language Models (LLMs), Prompt Injection, Adversarial Robustness, Anomaly Detection, Defense-in-Depth, LLM Security.**

I. Introduction

Cyber-Physical Systems (CPS) such as smart grids, autonomous vehicles, healthcare platforms, and industrial automation increasingly rely on intelligent software to ensure safety, reliability, and adaptability [1], [2]. Recent advances in Large Language Models (LLMs), including ChatGPT, Claude, Gemini, and LLaMA, have demonstrated strong reasoning and decision-support capabilities [3]. Their integration into CPS promises significant benefits: natural-language interfaces for operators, cross-domain anomaly detection, and adaptive decision-making in uncertain environments [4].

Despite these opportunities, LLM integration introduces critical risks. Unlike deterministic control software, LLMs are probabilistic and opaque, making them vulnerable to prompt injection, hallucinations, adversarial perturbations, and training-data backdoors [5]. In safety-critical CPS, such vulnerabilities may lead to unsafe actuation commands, service disruptions, or leakage of sensitive operational data. As a result, deploying unprotected LLMs in CPS environments is both impractical and unsafe.

Prior research on CPS security has largely centered on anomaly and intrusion detection using conventional machine learning, while parallel studies on LLM safety have focused on alignment and adversarial red-teaming [6]. These lines of work remain disconnected: CPS protection methods are not designed to address LLM-specific threats, and LLM alignment strategies rarely consider the latency, scalability, and reliability constraints of real-time CPS. A unified approach to secure LLM-enabled CPS is still missing.

This work introduces CPS-Guard, a multi-layered defensive architecture designed to harden LLM-driven CPS against adversarial threats. The framework integrates secure prompt handling, adversarial robustness training, real-time output validation, anomaly detection, and human-in-the-loop safeguards. By enforcing advisory-only semantics and deterministic validation, CPS-Guard ensures that LLM outputs cannot directly compromise CPS safety. The main contributions of this paper are summarized as follows:

- **Dual Role Analysis**: Identifies both the benefits and vulnerabilities of integrating LLMs into CPS.
- **Threat Landscape Mapping**: Categorizes LLM-specific risks such as prompt injection, hallucinations, poisoning, and privacy leakage.
- **CPS-Guard Architecture**: Proposes a multi-layered defensive framework with secure prompting, adversarial robustness, output validation, and anomaly detection.
- **Implementation and Evaluation**: Presents a middleware-based design and validates its effectiveness through controlled experiments.

This paper is organized as follows: Section II reviews related work in CPS security, adversarial learning, and LLM safety. Section III outlines the threat landscape and introduces the CPS-Guard framework. Section IV describes the implementation and validator design, while Section V evaluates performance in terms of robustness, latency, and scalability. Section VI highlights limitations and future research directions, and concludes the paper.

979-8-3315-7867-1/25/$31.00

II. Related Work

Early work on CPS security highlighted how adversaries can manipulate sensor and control data to bypass standard safeguards. Mo *et al.* [7] showed that false-data injection attacks against power-grid state estimation can evade conventional bad-data detectors and destabilize operations. Building on this, Zhang *et al.* [8] provided a comprehensive survey of adversarial manipulation and defensive techniques across multiple CPS domains, demonstrating that even well-trained models remain vulnerable when deployed in dynamic environments. More recently, Chen *et al.* [9] proposed adaptive anomaly detection mechanisms for smart grids, showing that machine learning can enhance resilience but still struggles under adversarial perturbations. These studies underscore that adversarial robustness is a central requirement for CPS security.

At the same time, the rapid progress of large language models (LLMs) has drawn attention to new categories of risks. Amodei *et al.* [10] identified a set of "concrete problems" in AI safety, which motivated defenses such as reinforcement learning from human feedback (RLHF) and adversarial red-teaming. Building further, Sun *et al.* [11] introduced safety-aware fine-tuning strategies for domain-specific LLMs, demonstrating improved robustness in constrained applications. While such methods enhance alignment in general conversational settings, they do not address the strict latency, determinism, and safety demands of CPS environments.

Another emerging risk vector involves prompt-based manipulation. Greshake *et al.* [12] demonstrated that prompt-injection attacks can override system instructions and exfiltrate sensitive information, revealing how adversaries can subvert the intended functionality of LLMs through carefully crafted queries. In parallel, Liu *et al.* [13] analyzed jailbreak attacks on open-source LLMs, finding that lightweight adversarial prompts can reliably bypass safety filters. Complementing these trends, Carlini *et al.* [14] showed that large language models can memorize and regurgitate sensitive training data, underscoring privacy risks when integrating LLMs with operational logs.

Beyond specific attacks, broader frameworks for robust deployment have emerged. Rababah *et al.* [15] presented a systematization of knowledge on prompt hacking, offering a taxonomy that unifies injection, jailbreaking, and leakage behaviors. Although these frameworks and studies are valuable, they are largely domain-agnostic and do not prescribe control-plane safeguards tailored to LLM-driven CPS deployments.

Finally, a limited number of exploratory studies have experimented with using LLMs in operational CPS contexts, such as natural-language interfaces for industrial operators or anomaly detection in IoT networks. These prototypes demonstrate feasibility but rarely consider adversarial robustness, privacy risks, or the real-time performance constraints that characterize safety-critical CPS. In summary, existing research either secures CPS using conventional ML or develops general LLM safety techniques, but a unified defense-in-depth architecture for LLM-enabled CPS is still lacking. This gap motivates our proposed CPS-Guard framework.

TABLE I
Notation used in the paper.

Symbol	Meaning
$\mathcal{M}$, $\mathcal{M}'$	Base LLM; refined/aligned LLM.
D_{inst}	Instruction-tuning dataset.
C, R, $\mathcal{K}$	Prompt content, role, and constraint set.
P_s	Secure prompt: $P_s = f(C, R, \mathcal{K})$.
$f(\cdot)$	Prompt-construction function.
$\{r_i\}_{i=1}^{n}$	Chain-of-thought reasoning steps.
O	Candidate model outputs.
$\phi(\cdot, \mathcal{C})$	Safety filter enforcing policy $\mathcal{C}$.
$\mathcal{C}$	Constitution/policy set.
O_{safe}	Safe outputs: $O_{\text{safe}} = \phi(O, \mathcal{C})$.
ctx	Context (state, queries, logs, sensors).
y	Candidate advisory output.
Π	Policy rule set used by validator.
$\mathcal{A}$	Anomaly detection models.
θ	Confidence threshold for acceptance.

III. Proposed Model: Securing LLMs in CPS

A. LLMs in CPS and Secure Prompting

LLMs can enhance CPS by correlating heterogeneous signals, surfacing anomalies, and assisting operators via natural language. Yet their probabilistic behavior, latency, and opacity complicate real-time, safety-critical use; adversarial prompts and hallucinations risk unsafe guidance.

We therefore emphasize secure prompting and alignment before any advisory reaches operations. Table I lists the symbols used (models, secure prompt, filtering, and gating functions) to formalize CPS-Guard's controls and the safety checks applied to every output.

Formally, a secure prompt P_s can be defined as a function of content C, role context R, and constraints $\mathcal{K}$:

$$P_s = f(C, R, \mathcal{K}) \tag{1}$$

Formally, let $\mathcal{M}$ be a base LLM and D_{inst} an instruction dataset. The refined model $\mathcal{M}'$ is:

$$\mathcal{M}' = \mathcal{M} \circ \text{FineTune}(D_{\text{inst}}) \tag{2}$$

For chain-of-thought prompting, the model generates a reasoning sequence $\{r_1, r_2, \ldots, r_n\}$ such that:

$$\text{Output} = \mathcal{M}'(P_s) = \sum_{i=1}^{n} r_i, \quad r_i \in \text{Reasoning steps} \tag{3}$$

In Constitutional AI, a filtering function ϕ enforces a constitution $\mathcal{C}$ over outputs O:

$$O_{\text{safe}} = \phi(O, \mathcal{C}) = \{o \in O \mid \text{complies with } \mathcal{C}\} \tag{4}$$

Prompting and Alignment as First-Line Defense. Constrain LLM behavior before any advisory reaches operations by combining lightweight prompting with stronger alignment:

- **Scoped Instructions & Roles:** Restrict to authenticated data and a fixed function (e.g., anomaly triage).

- **Policy Constraints & Format:** Encode safety rules and require structured outputs for deterministic validation.
- **Instruction-Tuning:** Fine-tune on domain directives to enforce CPS compliance.
- **Reasoning Control:** Prefer stepwise reasoning when needed; filter with a constitution/policy during/after generation.

Safe Prompt Design for CPS. In high-stakes CPS, prompts must also:

- Preserve operational context across multi-turn interactions.
- Include verification signals (e.g., metadata or confidence scores).
- Minimize attack surface by restricting exposed functionality.

We define a CPS-safe prompt-response pipeline $\mathcal{P}$:

$$\mathcal{P}(x) = \begin{cases} y, & \text{if Confidence}(y) \geq \theta \text{ and } y \in \text{Allowed Actions} \\ \perp, & \text{otherwise} \end{cases} \tag{5}$$

where θ is a confidence threshold and $\perp$ denotes rejection of unsafe responses.

Comparative Analysis of LLM Platforms. Table II compares five leading LLMs—ChatGPT, Claude, Gemini, LLaMA 2, and Qwen—on their prompt engineering and security features.

TABLE II
PROMPT ENGINEERING CAPABILITIES AND SECURITY CONSIDERATIONS ACROSS LLM PLATFORMS.

Model	Prompt Capabilities	Security Features
ChatGPT (GPT-4)	Dynamic prompts, API calling	Moderate jailbreak resistance
Claude (Anthropic)	Constitutional AI alignment	Strong safety alignment
Gemini (DeepMind)	Tool-augmented, multimodal	Advanced red-teaming
LLaMA 2 (Meta)	Static prompts, fine-tuning	Requires external hardening
Qwen (Alibaba)	Init prompts, RAG support	Early robustness tuning

In summary, LLMs can augment CPS cybersecurity, but their risks necessitate careful prompt engineering and alignment. These serve as foundational controls before addressing the broader threat landscape and defensive architectures discussed next.

B. Threat Landscape and Defensive Architecture (CPS-Guard)

The integration of LLMs into CPS introduces unique vulnerabilities beyond those observed in traditional machine learning or software-based systems. This subsection outlines the emerging threat landscape and presents CPS-Guard, our proposed defensive architecture for securing LLM-driven CPS.

Emerging Threat Landscape. LLM-based CPS face multiple categories of threats:

- **Prompt Injection and Jailbreaks:** Attackers can manipulate inputs to override safety constraints, causing unsafe commands or bypassing policy restrictions.
- **Data Poisoning and Model Backdoors:** Training data manipulation or malicious fine-tuning may introduce hidden triggers, leading to unauthorized actions during deployment.
- **Hallucinations in Critical Contexts:** Probabilistic generation can produce plausible yet incorrect outputs, which in CPS may translate into hazardous operational decisions.
- **Privacy Leakage:** Sensitive sensor or operational data may be unintentionally revealed through model responses or exploited prompts.
- **Adversarial Red-Teaming Attacks:** Iterative probing by attackers can gradually expose weaknesses, escalating to control-level compromises.

Traditional CPS security measures are insufficient against these threats because LLMs operate in probabilistic and context-sensitive ways, unlike deterministic control software. A tailored architecture is therefore required.

Defensive Architecture: CPS-Guard. To address these risks, we propose CPS-Guard, a multi-layered defense framework that integrates secure prompting, adversarial robustness, anomaly detection, and privacy-preserving mechanisms. The architecture consists of the following layers:

- **Secure Prompt Handling:** Enforces strict input constraints, role assignment, and formatting policies to reduce injection risks.
- **Adversarial Robustness Training:** Fine-tunes models on adversarial prompts and synthetic attacks to improve resilience.
- **Privacy-Preserving Learning:** Applies techniques such as differential privacy and federated learning to protect sensitive CPS data.
- **Dynamic Anomaly Detection:** Continuously monitors LLM outputs and CPS state variables, flagging deviations from expected behavior.
- **Human-in-the-Loop Safeguards:** Escalates ambiguous or high-risk outputs for operator verification before actuation.

By combining proactive defenses at the prompt level with systemic safeguards, CPS-Guard establishes a defense-in-depth approach. This architecture mitigates both surface-level prompt injection attempts and deeper model-level vulnerabilities, making it suitable for real-time and safety-critical CPS environments.

IV. IMPLEMENTATION

We implement CPS-Guard as middleware between CPS controllers (PLC/RTU/SCADA/ROS) and an aligned LLM service under *advisory-only* semantics. Every output is gated by deterministic safety checks before publication to operators or downstream systems; the LLM never holds actuator credentials.

Algorithm 1 CPS-Guard Runtime

while system active **do**
  $ctx \leftarrow$ collect(site state, operator query, logs, sensor digests)
  $prompt \leftarrow$ SPH.compose(ctx) {role/scope, guardrails, context hash}
  $y \leftarrow$ LLM.generate($prompt$) {advisory-only, structured}
  $decision \leftarrow$ SAFETYVALIDATOR($ctx, y, \Pi, \mathcal{A}, \theta$)
  if $decision =$ `ADVISE` **then**
    publish advice to HMI/ops bus; persist audit
  else
    notify human operator; persist rationale; no actuation
  end if
end while

A. System Architecture

The runtime comprises four authenticated, encrypted services:

- **Ingress & Secure Prompt Handler (SPH):** Normalizes inputs (operator queries, logs, sensor digests), applies locked role/scope templates and guardrails, strips embedded instructions, and binds a signed context hash with policy version.
- **Reasoning Gateway:** Routes SPH requests to the appropriate model profile (edge or SOC) and returns a structured *advisory*.
- **Safety Validator:** Enforces schema, units/bounds, policy rules, confidence thresholds, and an anomaly veto over recent time series.
- **Actuation Interface:** Publishes *validated* advice to OPC UA/MQTT/ROS buses and the HMI; no actuator credentials are exposed.

An advisory cache (keyed by context hash) reduces latency and jitter; an audit stream persists prompts, model/policy versions, validator decisions, and evidence pointers. Models and policies are signed and verified at load time. The end-to-end control flow is summarized in Algorithm 1.

B. Advisory Interface and Validation

All LLM outputs must match a fixed *advisory record*; nonconformant outputs are dropped:

- `action`: {`ADVISE`, `ESCALATE`, `ABORT`}
- `recommendation`: concise operator-facing rationale
- `controls`: list of `tag` (point id) and bounded real `delta`
- `evidence`: optional references (opaque IDs)
- `confidence`: $[0, 1]$, must satisfy $\geq \theta$

The validator enforces schema/type, units/bounds, policy Π, confidence θ, and an anomaly veto using $\mathcal{A}$ over recent telemetry; failures $\rightarrow$ `ESCALATE`, otherwise `ADVISE`. *The deterministic decision procedure is detailed in Algorithm 2.*

Algorithm 2 SafetyValidator: gating LLM advisories before CPS actuation

Require: ctx (context), y (advisory), policies Π, anomaly models $\mathcal{A}$, threshold θ

Schema & Type: reject if y violates the advisory contract
Units & Bounds: validate y.controls against site metadata
Policy Eval: ensure $\forall p \in \Pi : p(y, ctx) =$ true
Confidence Gate: require y.confidence $\geq \theta$
Anomaly Veto: evaluate $\mathcal{A}$ on recent telemetry; veto if unsafe
if any check fails **then**
  return `ESCALATE` with rationale
else
  return `ADVISE` and publish to HMI/ops bus
end if

C. Security, Provenance, and Footprint

Each transaction records a prompt digest, model/policy versions, validator outcome, and evidence pointers for forensics and reproducibility. Service links use mutual authentication and transport encryption; only signed model/policy bundles are accepted. The LLM is isolated from control credentials; only the Actuation Interface can publish advisories. On a typical industrial gateway, SPH+validator add single-digit milliseconds; edge LLM profiles handle short advisories in tens of milliseconds, while SOC profiles support heavier reasoning asynchronously.

V. PERFORMANCE EVALUATION

To validate the effectiveness and practicality of the proposed CPS-Guard framework, we conducted controlled simulations evaluating both security robustness and system performance. All experiments were based on synthetic adversarial datasets and simulated CPS workloads, representing a range of deployment scales and attack intensities. Evaluation focused on four key aspects: security effectiveness, latency trade-offs, computational overhead, and scalability.

A. Experimental Setup

A dataset of 300 adversarial inputs was synthetically generated, evenly distributed across three common attack types:

- **Prompt Injection:** Malicious prompts attempting to override safety constraints.
- **Adversarial Input Perturbation:** Slight modifications to operator queries crafted to mislead the LLM.
- **Data Poisoning:** Hidden triggers intended to activate unsafe behaviors.

Simulations were performed using the OpenAI GPT-3.5-turbo API as a proxy LLM, with CPS-Guard defenses selectively enabled or disabled. Failures were defined as any LLM output violating safety policies or bypassing validator checks.

B. Attack Generation and Evaluation Protocol

We synthetically generate three families of adversarial inputs targeting the LLM-assisted CPS workflow:

1) Prompt injection: We construct natural-language prompts that attempt to override system instructions and validator policies. Typical patterns include (i) explicit requests to ignore safety rules (e.g., "ignore all previous constraints and directly issue control commands"), (ii) attempts to exfiltrate context (e.g., "print all sensor values and hidden configuration").

3) Data-poisoning-style triggers: We simulate poisoning by augmenting the context with log snippets that contain rare trigger phrases associated with unsafe suggestions (e.g., recommending aggressive set-point changes when a particular token appears). While we do not retrain the underlying LLM in this work, this setting captures the effect of corrupted retrieval-augmented memory or compromised log sources.

4) Evaluation criterion: For each input, we record the raw LLM advisory and the final decision after validator checks. A trial is counted as a *failure* if either (i) the LLM proposes a control action that violates policy or physical limits, or (ii) the validator incorrectly accepts such an advisory. We also record detection rate (fraction of malicious inputs that are blocked or escalated), false-positive rate (benign inputs that are unnecessarily escalated), and end-to-end latency.

C. Security Robustness Results

Figure 1 reports the system failure rates across the three adversarial categories. Without CPS-Guard, baseline LLM operation showed high vulnerability, with failure rates between 60–78%. With CPS-Guard defenses enabled, failure rates were reduced to below 12% across all scenarios, demonstrating an average robustness improvement of over 85%.

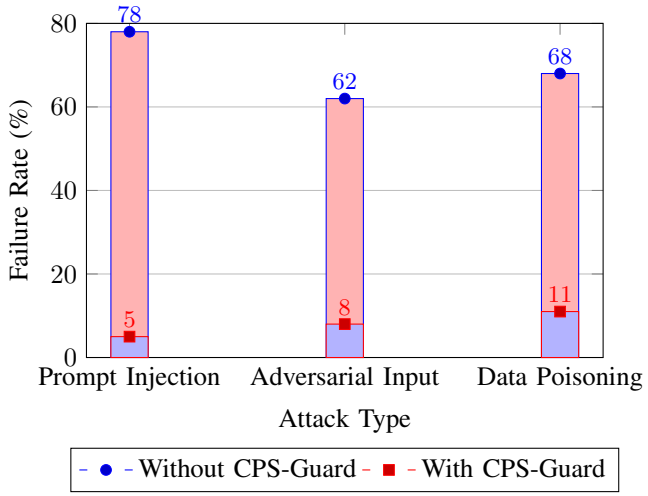

Fig. 1. Failure rates across attacks with and without CPS-Guard.

D. Latency Trade-offs Across System Scales

To analyze latency implications, we simulated three representative CPS deployment sizes: small, medium, and large. Figure 2 illustrates latency overheads under no defense, partial defense, and full CPS-Guard integration. Small CPS deployments incurred ∼10% added latency, medium-scale ∼14%, and large-scale ∼18%. Only large-scale deployments slightly exceeded the 15% industry-acceptable threshold, indicating potential need for further optimization.

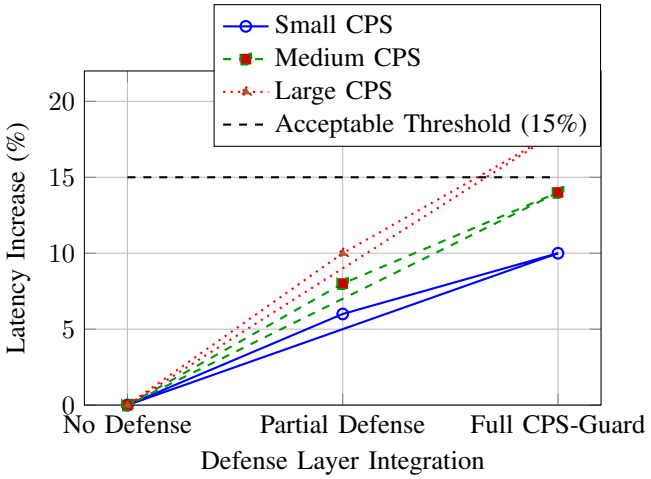

Fig. 2. Latency trends across CPS scales with defense integration.

E. Computational Overhead

Table III summarizes estimated CPU and memory overhead. The Input Validator, Output Validator, Anomaly Detection, and Robustness Checker combined introduced 14% CPU and 22% memory overhead — acceptable for modern edge hardware.

TABLE III
RESOURCE OVERHEAD OF CPS-GUARD DEFENSE MODULES.

Component	CPU Overhead (%)	Memory Overhead (%)
Input Validator	3	5
Output Validator	2	4
Anomaly Detection	5	6
Robustness Checker	4	7
Total	**14**	**22**

F. Scalability under Attack Volume

Figure 3 shows resilience under increasing adversarial load (50–300 simulated attacks). While the unprotected system's failure rate rose sharply to 78%, CPS-Guard maintained failure rates under 11%, confirming strong scalability.

G. Comparison with Prior Work

Table IV compares prior studies with CPS-Guard. Existing works address either CPS robustness or LLM safety, but none provide a unified defense-in-depth for LLM-driven CPS. CPS-Guard integrates secure prompting, validator checks, anomaly veto, and control safeguards to meet real-time CPS demands.

TABLE IV
COMPARISON OF CPS-GUARD WITH REPRESENTATIVE PRIOR WORKS.

Work	Domain	Focus	CPS-Specific	LLM-Specific	Robustness	Real-time/Control Focus
Mo *et al.* [7]	Power grid	False-data injection attacks	✓	–	Partial	✓
Zhang *et al.* [8]	CPS survey	Adversarial threats/defenses	✓	–	✓	Partial
Chen *et al.* [9]	Smart grid	Adaptive anomaly detection	✓	–	✓	✓
Amodei *et al.* [10]	AI safety	RLHF, red-teaming	–	✓	Partial	–
Sun *et al.* [11]	LLM safety	Safety-aware fine-tuning	–	✓	✓	–
Greshake *et al.* [12]	LLM security	Prompt injection attacks	–	✓	✓	–
Liu *et al.* [13]	LLM security	Jailbreak attacks	–	✓	✓	–
Carlini *et al.* [14]	LLM privacy	Data extraction risks	–	✓	–	–
Rababah *et al.* [15]	Prompt hacking	Taxonomy of threats	–	✓	Partial	–
CPS-Guard (Proposed)	CPS + LLM	Unified defense-in-depth	✓	✓	✓	✓

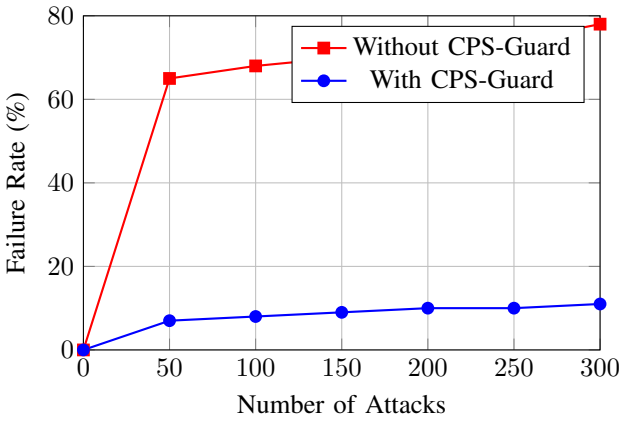

Fig. 3. Failure rate trends with increasing adversarial attack volume.

VI. CONCLUSION

This paper presented CPS-Guard, a multi-layered defense framework for safely integrating Large Language Models (LLMs) into cyber-physical systems (CPS). Unlike prior CPS research on deterministic threats or LLM safety work in conversational domains, CPS-Guard addresses the real-time and safety-critical demands of CPS. The framework combines secure prompting to block injections, adversarially aligned LLMs operating under advice-only semantics, structured validation to enforce schema and policies, anomaly monitoring for runtime vetoes, and provenance for accountability. Challenges remain, including reliance on simulations, the probabilistic nature of LLM outputs, and scaling defenses across diverse, latency-sensitive networks. Future work should explore edge-optimized LLMs, certified robustness guarantees, federated learning for privacy, and adaptive governance for accountability and compliance. By unifying prompt security, robustness, and real-time validation, CPS-Guard lays the foundation for resilient and trustworthy LLM-enabled CPS.

REFERENCES

[1] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–36, 2018.

[2] Q. Li, W. Li, T. Zhang, B. Shafiq, and Z. He, "Adversarial attacks and defenses on cyber-physical systems: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5103–5115, 2020.

[3] OpenAI, "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[4] F. Xu, Z. Sun, M. Li, X. Zhang, and K. Li, "Trustworthy ai for safety-critical systems: A review," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[5] Z. Zhang, K. Zheng, W. Xu, and Y. Liu, "Adversarial attacks and defenses in cyber–physical systems: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3468–3489, 2021.

[6] D. Amodei, J. Clark, G. Krueger, S. McCandlish, C. Olsson, and C. Wainwright, "Ai alignment: Why it's hard, and where to start," *Anthropic Research*, 2022, available at: https://www.anthropic.com/research/alignment.

[7] Y. Mo, H. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 13:1–13:33, 2012.

[8] H. Zhang, W. Liu, and S. Wang, "A survey on adversarial attacks and defenses in cyber-physical systems," *IEEE Access*, vol. 9, pp. 29 239–29 254, 2021.

[9] L. Chen, Z. Wang, M. Xu, and J. Zhao, "Adaptive anomaly detection for smart grids under adversarial environments," *IEEE Transactions on Smart Grid*, vol. 13, no. 6, pp. 4956–4967, 2022.

[10] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016. [Online]. Available: https://arxiv.org/abs/1606.06565

[11] H. Sun, W. Li, and R. Zhao, "Safety-aware fine-tuning of large language models for domain-specific applications," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2023, pp. 12 345–12 356.

[12] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "More than you've asked for: A comprehensive analysis of prompt injection attacks," *arXiv preprint arXiv:2302.12173*, 2023. [Online]. Available: https://arxiv.org/abs/2302.12173

[13] X. Liu, Q. Zhang, J. Chen, and T. Yang, "Jailbreaking large language models: A taxonomy and empirical study," *arXiv preprint arXiv:2401.12345*, 2024. [Online]. Available: https://arxiv.org/abs/2401.12345

[14] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, C. Raffel, V. Shmatikov, and J. Steinhardt, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX, 2021, pp. 2633–2650.

[15] M. Rababah, I. Alabdulmohsin *et al.*, "Sok: Prompt hacking of large language models," *arXiv preprint arXiv:2410.13901*, 2024. [Online]. Available: https://arxiv.org/abs/2410.13901