Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Automated Runtime Verification of EV-Charging Software Using Physics-Informed Oracles and Search-Based Threshold Synthesis

Mahmoud El Amri

mahmoudelamri@gmail.com

University of Sousse

Ferdaws Ben Naceur

University of Sousse

Chokri Ben Salah

University of Monastir

Research Article

Keywords: automated runtime verification, cyber-physical software, test oracle, search-based software engineering, electric vehicle charging

Posted Date: November 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11027842/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Automated Runtime Verification of EV-Charging Software Using Physics-Informed Oracles and Search-Based Threshold Synthesis

Ferdaws Ben Naceur[1], Mahmoud El Amri[2], and Chokri Ben Salah[2,3]

[1] University of Sousse, National Engineering School of Sousse, LATIS - Laboratory of Advanced Technology and Intelligent Systems, 4023 Sousse, Tunisia

[2] Laboratory of Automation, Electrical Systems and Environment (LASEE), University of Monastir, Monastir 5000, Tunisia

[3] Higher Institute of Applied Sciences and Technology of Sousse, University of Sousse, Sousse 4003, Tunisia

* Corresponding author: mahmoudelamri@gmail.com; Tel.: +216 50493749

Abstract

This paper presents an automated runtime verification approach for electric vehicle charging software that checks Open Charge Point Protocol (OCPP) telemetry against an independent physical oracle at the edge. Integrity faults or attacks can preserve statistical plausibility while violating the physical charging process, which limits software-side anomaly monitors that observe only reported data. The proposed approach treats the charging station as a cyber-physical software system and combines two executable runtime oracles: a protocol and statistical oracle over OCPP MeterValues and a physics-informed oracle derived from local hardware power measurements and constant-current/constant-voltage charging dynamics. Their scores are normalized and fused by a transparent decision rule. Two decision thresholds are synthesized automatically with particle swarm optimization under a shared false-alarm budget, turning calibration into a small search-based software engineering problem. A seeded simulation benchmark evaluates five telemetry mutation operators: scaling, bias, replay, stealthy value manipulation, and protocol-timing corruption. At a 5% target false-positive rate, the cross-layer monitor detects 93.1% of attacks with a 4.6% false-positive rate and an area under the ROC curve of 0.953, compared with 72.8% and 0.845 for the physics-only monitor and 44.1% and 0.691 for the cyber-only monitor. Average detection latency is 2.36 min and processing cost is 50.8 μs per session. The results show that physics-informed runtime oracles can strengthen automated software assurance for cyber-physical telemetry while remaining interpretable and lightweight. External validation on measured charging-station data remains future work.

Keywords: automated runtime verification; cyber-physical software; test oracle; search-based software engineering; electric vehicle charging

1 Introduction

Automated Software Engineering increasingly addresses verification, testing, monitoring, and quality assurance for software that interacts with physical environments. Recent work in this journal has shown the value of automated requirement falsification for cyber-physical systems, machine-readable test specifications, search-based testing, and continuously updated operational models [1-4]. These lines of work share a common objective: replacing manual or ad hoc assurance activities with repeatable mechanisms that can generate, execute, or evaluate evidence about software behavior.

Electric vehicle charging stations are a representative cyber-physical software system. A modern charge point executes embedded control software, exchanges MeterValues and transaction information through OCPP, and exposes remote interfaces for billing, smart charging, and fleet management. Security surveys and system assessments have documented realistic weaknesses in the charging ecosystem and its protocol deployments [5, 6]. The engineering problem is therefore not only intrusion detection. It is also the construction of an automated software assurance mechanism that can decide, at runtime, whether an application-layer telemetry trace is consistent with the physical behavior that the software is supposed to report.

This distinction matters because software-visible telemetry is not an independent test oracle. False data injection can alter values while keeping them statistically plausible, and classic residual detectors can be bypassed when the attacker understands the model used by the detector [7-10]. Charging-specific detectors are increasingly based on machine learning, graph representations, deep learning, anomaly correlation, or reinforcement learning [11-16]. These approaches are valuable, but many of them still evaluate a reported trace against other software-visible information rather than against a physically independent oracle.

The charger itself offers such an oracle. Its local meter and state-of-charge information are produced inside the physical charging loop and are outside the write access assumed for an application-layer telemetry adversary. This work uses that observation to construct an automated runtime monitor. The monitor combines a physics-informed oracle with a cyber/protocol oracle and automatically calibrates their two decision thresholds by particle swarm optimization [17]. The optimization objective explicitly enforces a common false-positive budget, which makes the calibration step reproducible and the comparison with single-oracle baselines fair.

The study is organized around three software-engineering questions. RQ1 asks whether the combined runtime monitor improves detection effectiveness at a controlled false-positive rate. RQ2 asks whether the two oracles are complementary across heterogeneous telemetry mutations. RQ3 asks whether the automated monitor is sufficiently lightweight for edge execution in the charging-station software stack.

The contributions are fourfold. First, we formulate charging-telemetry integrity checking as an automated runtime verification problem and define a hardware-anchored physics oracle for OCPP MeterValues. Second, we combine the physics oracle with a protocol/statistical oracle through an interpretable decision rule and synthesize the two thresholds automatically as a small search-based software engineering problem. Third, we define five telemetry mutation operators that exercise both value and protocol failure modes under a seeded evaluation protocol. Fourth, we report an equal-false-positive empirical comparison of signature, cyber-only, physics-only, and cross-layer monitors, including attack-wise effectiveness, ROC area, detection latency, and processing cost.

The remainder of the article is structured as follows. Section 2 reviews automated verification and charging-security work. Section 3 presents the runtime verification architecture, oracles, mutation operators, and search-based calibration procedure. Section 4 defines the empirical methodology and research questions. Section 5 reports the results. Section 6 discusses implications and threats to validity, and Section 7 concludes the article.

2 Background and related work

2.1 Automated verification and testing of cyber-physical software

Automated verification of cyber-physical software depends on executable or machine-checkable evidence about system behavior. Peltomäki and Porres [1] formulate requirement falsification as automated search for counterexamples to formalized cyber-physical requirements. Meyer et al. [2] show how machine-readable test specifications and automated execution can improve reuse and scalability of scenario-based testing. Dakhama et al. [3] combine search-based testing and differential oracles to expose mismatches between system simulators and hardware behavior. Poth et al. [18] emphasize that the value of automated testing techniques depends on systematic evaluation at realistic operating points, while Mazkatli et al. [4] demonstrate how operational monitoring can automatically maintain model consistency during software evolution.

The present work is closest to runtime verification and oracle-based testing. Instead of generating new control inputs, it evaluates each observed OCPP telemetry interval against two runtime oracles. The first oracle is protocol and statistics based. The second is a cyber-physical oracle anchored in independently measured charging behavior. The latter plays a role analogous to a differential oracle: the reported software output is checked against a reference that the telemetry path cannot modify. This separation between the system output and the oracle is essential for avoiding circular verification.

2.2 False data injection and EV-charging software security

False data injection was formalized in power-system state estimation by Liu et al. [7], with related analyses of malicious and integrity attacks by Kosut et al. [8] and Xie et al. [19]. Surveys subsequently organized attack models and detection strategies [9, 10, 20]. Representative detectors include Kalman-filter residual monitoring [21], ensemble learning [22], time-series analysis [23], graph convolution [24], and autoencoder reconstruction [25]. The recurring

difficulty is that a detector based only on reported or model-consistent quantities can be evaded by carefully constructed values.

In the EV domain, security research has covered charging-ecosystem vulnerabilities, grid-level impacts, coordinated switching, and resilience [6, 26-28]. Charging-station detection has moved toward data-driven intrusion detection and anomaly analysis, including traffic classifiers, graph representations, hidden Markov models, session anomaly mining, deep learning, and multi-agent resilience frameworks [11-16, 29]. Physics-guided learning has also been demonstrated inside the EV powertrain using device-level signals [30]. The gap addressed here is narrower: automated runtime verification of the charging telemetry interface using a local physical reference that is independent of the OCPP reporting path.

2.3 Positioning of the proposed approach

Simulation is widely used to evaluate charging and cyber-physical assurance methods before field deployment. EVLibSim supports configurable charging-station simulation [31], digital-twin and co-simulation approaches support cyber-resilience studies [32, 33], and public resources such as ACN-Data and CICEVSE2024 provide potential measured-data targets for later validation [34, 35]. The present study remains a seeded proof-of-concept simulation. Its novelty is not a new battery model or a new intrusion classifier; it is the software-assurance structure that separates the reported artifact from an independent physical oracle and automates threshold synthesis under a shared false-positive constraint.

Table 1 Positioning against representative automated assurance and EV-security approaches

Work	Automation target	Oracle or evidence	Automated search / tuning	Primary domain
Peltomäki and Porres [1]	Requirement falsification	Formal robustness metric	Generative search	Generic CPS
Meyer et al. [2]	Scenario-based testing	Machine-readable pass criteria	Parameter exploration	Automated driving
Dakhama et al. [3]	System-simulator testing	Differential hardware behavior	Search-based fuzzing	System simulators
Girdhar et al. [12]	Charging anomaly detection	Cyber-physical event correlation	Model-based	EV charging
Jahangir et al. [14]	Charge-manipulation detection	Learned cyber features	Learned detector	EV charging
Guo et al. [30]	Vehicle attack detection	Powertrain physical signals	Learned detector	EV powertrain
Proposed	Runtime verification of OCPP telemetry	Hardware-anchored physics + protocol oracle	PSO threshold synthesis	EV charging software

3 Automated runtime verification approach

3.1 System under observation and monitor placement

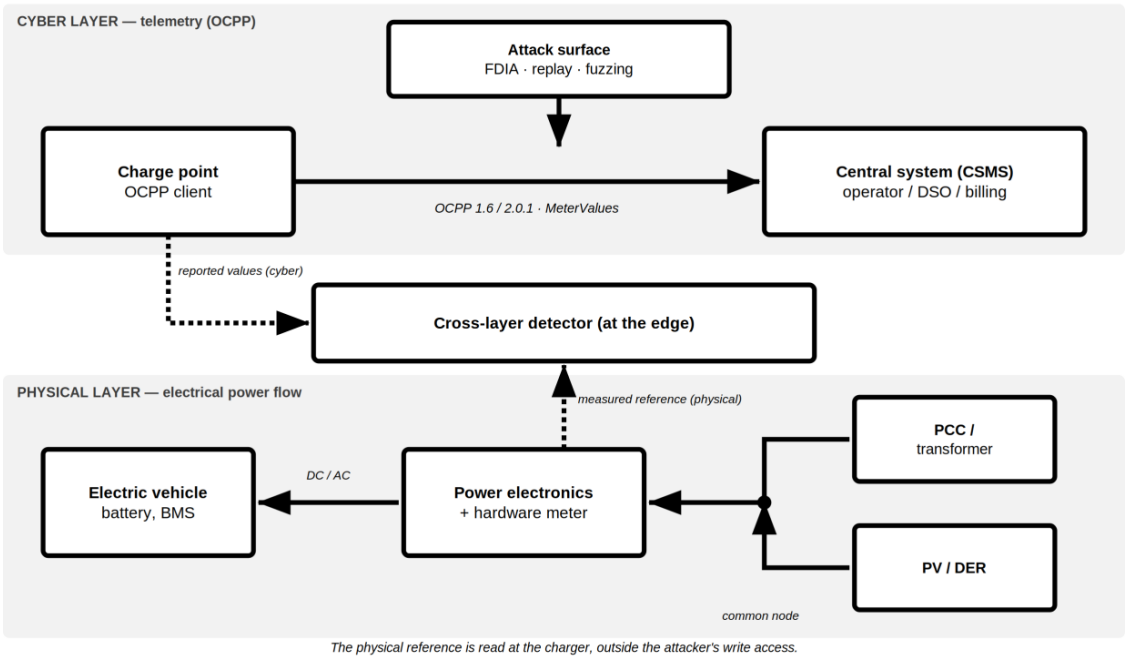


Fig. 1 EV-charging cyber-physical architecture, OCPP telemetry path, and runtime-monitor placement

Figure 1 defines the system under observation. An electric vehicle is connected to a charge point that draws power through the point of common coupling and reports OCPP MeterValues to a central charging-station management system. The runtime monitor executes at the edge, inside or beside the charge point. It reads two classes of artifacts: the software-visible telemetry sent by the OCPP client and hardware-anchored measurements of delivered power and state of charge. The threat model assumes that an adversary can alter the telemetry values or their metadata after they are formed by the charger software, but cannot alter the physical power delivered by the power electronics or the local hardware meter. This separation is the trust boundary on which the oracle is built [5, 6].

3.2 Physical charging model and oracle basis

The physical oracle uses the constant-current/constant-voltage charging trajectory as a compact executable model. Let s denote battery state of charge. Over one reporting interval, the energy balance is:

$$SOC(t + \Delta t) = SOC(t) + \frac{\eta_c P(t) \Delta t}{C_b} \quad (1)$$

Where ηc is charging efficiency, P(t) is delivered power, Cb is battery capacity, and Δt is the reporting interval. The expected power as a function of state of charge is represented by:

$$P_{phys}(s) = P_{max} \cdot min\{1, e^{-(s - s_{cv})/\tau}\} \quad (2)$$

Where Pmax is rated charger power, scv is the transition state of charge, and τ is the constant-voltage decay parameter. The minimum operator selects the constant-current branch before the transition and the exponential-decay branch afterwards. The model is intentionally lightweight because the oracle must be executable at every reporting interval. Open charging data show the same qualitative constant-current then constant-voltage behavior, although the quantitative evaluation in this paper uses the seeded simulation described in Sect. 4 [34].

The reason for using an independent physical oracle can be seen from the classic measurement model:

$$z = h(x) + e \quad (3)$$

Where h is the measurement function and e is measurement noise. A conventional bad-data detector compares the residual with a threshold:

$$r_{BDD} = \|z - H\hat{x}\|_2 > \tau_{BDD} \tag{4}$$

If an attacker injects a carefully constructed vector in the range of the measurement model, the reported residual can remain unchanged [7]:

$$z_a = z + a,\ a = Hc \ \Rightarrow\ \|z_a - H\hat{x}_a\|_2 = \|z - H\hat{x}\|_2 \tag{5}$$

Equation (5) motivates the software-engineering design choice in this work. A monitor that only recomputes properties from the same reported telemetry can be made self-consistent by an informed adversary. The proposed oracle instead compares the reported value with a hardware-anchored physical reference that lies outside the assumed telemetry write boundary.

3.3 Telemetry mutation operators

The empirical benchmark uses five mutations of an otherwise valid execution trace. They are treated as adversarial telemetry mutation operators because they modify the software-visible artifact observed by the monitor while leaving the underlying charging process unchanged. A scaling mutation is defined as:

$$P_{rep}(t) = \alpha P(t) + n(t),\ \ \alpha \neq 1 \tag{6}$$

An additive bias mutation is:

$$P_{rep}(t) = P(t) + \delta + n(t),\ \ \delta \neq 0 \tag{7}$$

A replay mutation substitutes the power profile of another session, following the general replay threat studied in control systems [36]:

$$P_{rep}(t) = P^{(j)}(t) + n(t),\ \ j \neq i \tag{8}$$

A stealthy value mutation uses a factor close to unity so that the deviation approaches normal model uncertainty:

$$P_{rep}(t) = \alpha_s P_{phys}\big(SOC(t)\big) + n_s(t),\ \ \alpha_s \to 1 \tag{9}$$

Finally, a protocol-timing mutation leaves the physical value unchanged but corrupts freshness or sequence metadata:

$$P_{rep}(t) = P(t),\ \ \varphi_{fresh} = 1 \tag{10}$$

Here n(t) denotes metering noise and φfresh is a binary freshness indicator. The five operators deliberately span two different failure surfaces: value consistency and protocol integrity. Table 2 summarizes which runtime oracle is expected to observe each mutation.

Table 2 Telemetry mutation operators and expected oracle coverage

Mutation operator	Transformation	Eq.	Physics oracle	Cyber/protocol oracle
Scaling	reported power multiplied by α	(6)	yes	partial
Bias	constant offset δ added	(7)	yes	partial
Replay	foreign session substituted	(8)	yes	partial
Stealthy value	rescaling with α close to unity	(9)	partial	no
Protocol timing	true values, corrupted metadata	(10)	no	yes

3.4 Runtime-oracle construction

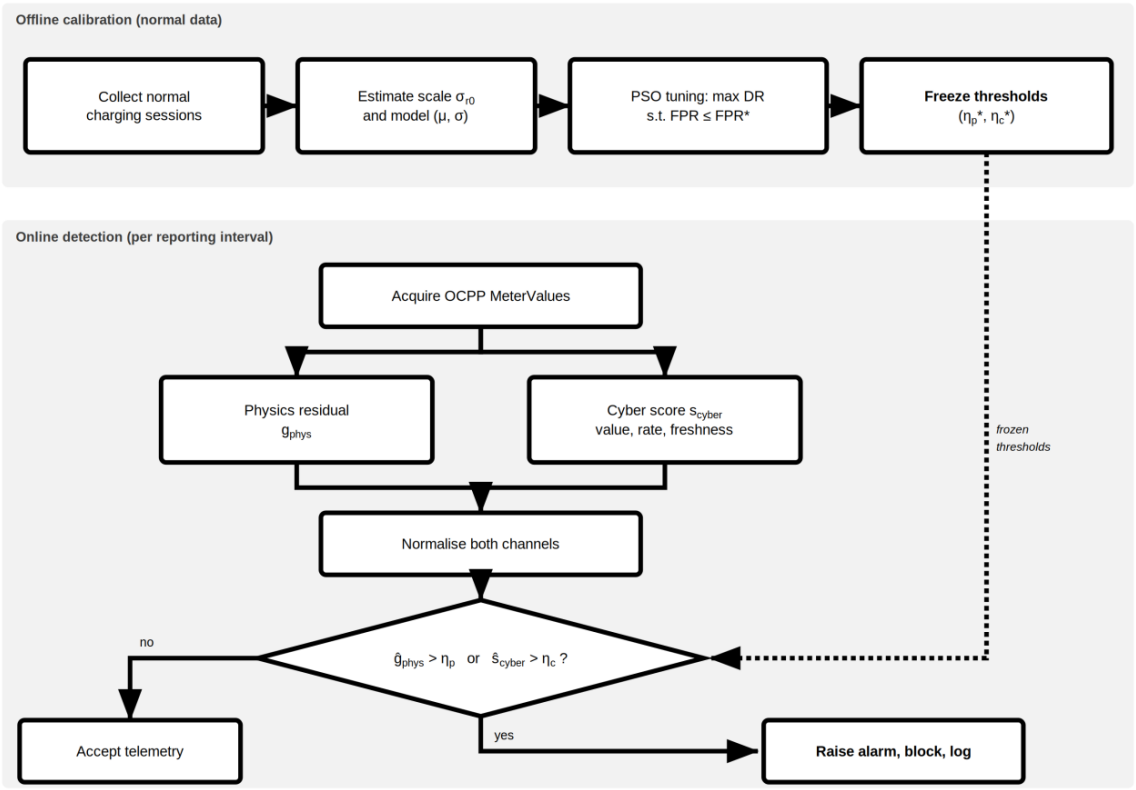


Fig. 2 Automated calibration and online runtime-verification pipeline

Figure 2 separates the approach into an offline calibration stage and an online verification stage. Offline, normal sessions estimate the scale of the physics residual and the distribution of cyber features. A labelled validation set is then used by the search procedure to synthesize the two thresholds under a target false-positive budget. Online, every OCPP reporting interval is evaluated by both oracles and then fused into a single decision.

The physics oracle forms the residual between reported power and the expected power evaluated at hardware-measured state of charge:

$$r(t) = P_{rep}(t) - P_{phys}\big(SOC_{hw}(t)\big) \tag{11}$$

Because the state of charge and delivered power used by the reference are read from charger hardware, the application-layer telemetry adversary cannot directly rewrite the oracle input. The residual is summarized by a statistic combining root-mean-square and peak error, normalized by the normal residual scale σr0:

$$g_{phys} = \frac{0.6}{\sigma_{r0}}\sqrt{\frac{1}{T}\sum_{t=1}^{T} r^2(t)} + \frac{0.4}{\sigma_{r0}} max_t abs(r(t)) \tag{12}$$

The fixed 0.6/0.4 weighting gives persistent and short excursions representation in the same score. These weights are not tuned separately per attack. The cyber/protocol oracle summarizes three properties of the reported power stream, namely its mean, dispersion, and maximum one-step variation:

$$f = \left[\mu_{rep}, \sigma_{rep}, \Delta_{max}\right]^T \tag{13}$$

The standardized feature deviations are combined with the protocol freshness indicator:

$$s_{cyber} = max(max_{j\in\{1,2,3\}} \frac{abs(f_j - \mu_j)}{\sigma_j}, \lambda_f \varphi_{fresh}) \tag{14}$$

Where μj and σj are feature mean and standard deviation on normal calibration data and λf is the freshness weight. This oracle intentionally remains lightweight: it is meant to detect protocol corruption and obvious statistical deviations rather than replace the physical oracle with a large learned classifier.

3.5 Oracle fusion and executable decision logic

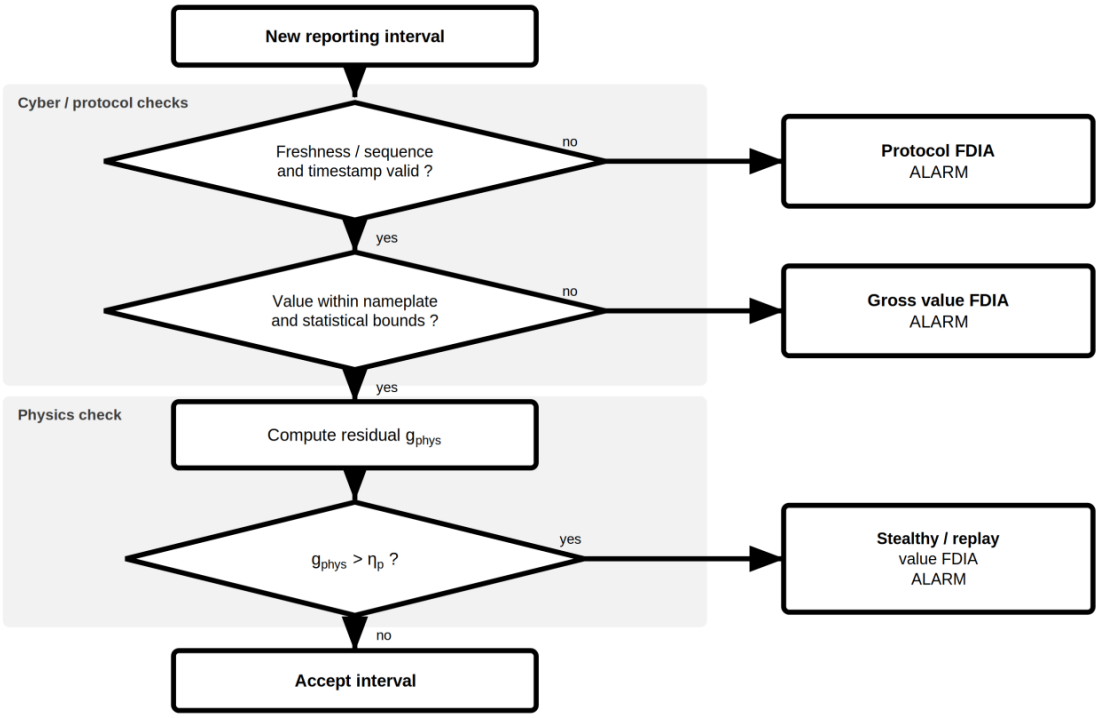


Fig. 3 Executable runtime-monitor decision cascade

The two oracle scores have different scales. They are therefore normalized by reference values defined as the 99th percentile of each score on normal validation data:

$$\hat{g}_{phys} = \frac{g_{phys}}{g_{ref}}, \quad \hat{s}_{cyber} = \frac{s_{cyber}}{s_{ref}} \tag{15}$$

The monitor raises an alarm when either normalized score exceeds its synthesized threshold:

$$A = 1\{\hat{g}_{phys} > \eta_p\} \ \vee \ 1\{\hat{s}_{cyber} > \eta_c\} \tag{16}$$

The disjunction is deliberately simple. The main discriminating power comes from the independence of the physical oracle rather than from a complex fusion model. The decision cascade in Fig. 3 makes each alarm traceable to a concrete software-assurance rule: freshness violation, nameplate/statistical violation, or physical inconsistency.

The monitor therefore provides an operational explanation with every alarm instead of only returning a classifier label.

3.6 Search-based threshold synthesis

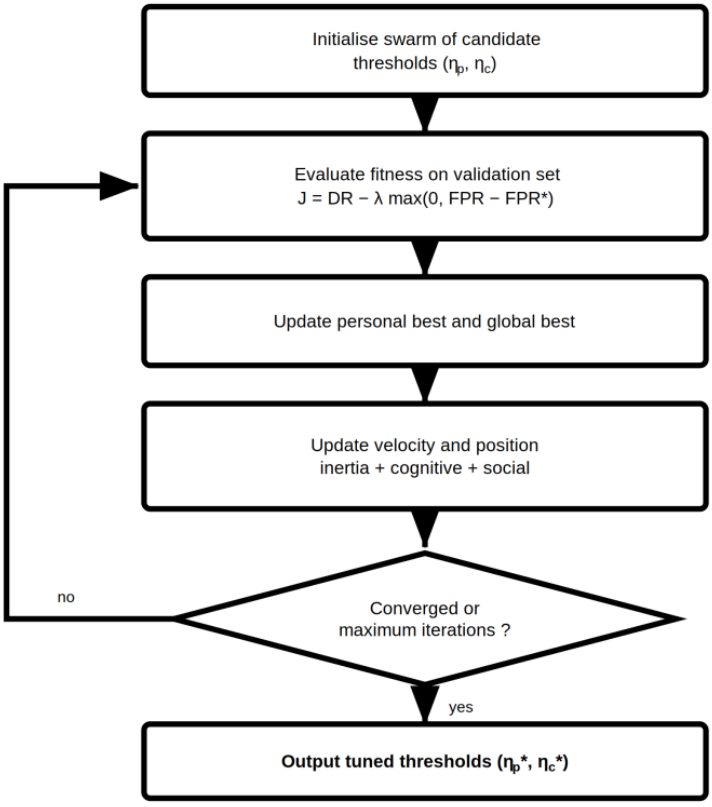


Fig. 4 Search-based synthesis of the two runtime-monitor thresholds

Manual threshold selection creates a repeatability problem and can make baseline comparisons unfair. We therefore cast threshold calibration as a two-variable search problem. Particle swarm optimization [17] maximizes a penalized fitness that rewards attack detection while penalizing violation of the target false-positive rate:

$$J(\eta_p, \eta_c) = DR - \lambda\, max(0, FPR - FPR_{tgt}) \tag{17}$$

Where DR and FPR are measured on the validation set, FPR_tgt is the target budget, and λ penalizes excess false positives. Each particle represents a candidate threshold pair and is updated by the canonical velocity and position rules:

$$v_i^{k+1} = \omega v_i^k + c_1 r_1 \left(p_i - x_i^k\right) + c_2 r_2 \left(g - x_i^k\right) \tag{18}$$

$$x_i^{k+1} = \Pi_{[x_{min}, x_{max}]} \left(x_i^k + v_i^{k+1}\right) \tag{19}$$

Here ω is inertia, c1 and c2 are cognitive and social coefficients, r1 and r2 are uniform random variables, pi is the particle best, g is the global best, and Π projects onto the allowed search box. Only two scalars are synthesized, which keeps the search space small and the resulting monitor interpretable.

Algorithm 1 Automated calibration and search-based threshold synthesis

Require: normal calibration set Dcal; labelled validation set Dval; target FPR_tgt; PSO parameters (S, K, ω, c1, c2, λ); search box [xmin, xmax]
Ensure: thresholds (ηp*, ηc*) and calibration constants
1. For each session in Dcal, compute residual r(t) by Eq. (11) and feature vector f by Eq. (13).
2. Estimate σr0 and feature means/standard deviations (μj, σj) from normal calibration data.
3. Compute gphys and scyber on Dval using Eqs. (12) and (14).
4. Set gref and sref to the 99th percentile of normal validation scores and normalize by Eq. (15).
5. Initialize S candidate threshold pairs and velocities.
6. Evaluate fitness J by Eq. (17); update personal and global best solutions.
7. Iterate velocity and position updates by Eqs. (18)-(19) for K iterations.
8. Return the best threshold pair (ηp*, ηc*) and calibration constants.

Algorithm 2 Online automated runtime verification at the charge point

Require: OCPP MeterValues stream; hardware power/state measurements; calibration constants; thresholds (ηp*, ηc*); nameplate limit Pnp
Ensure: one verification decision per reporting interval
1. Receive Prep(t), timestamp, and sequence metadata from the OCPP client.
2. Read delivered power and SOChw(t) from charger hardware.
3. Evaluate protocol freshness and nameplate constraints.
4. Evaluate Pphys(SOChw(t)) with Eq. (2), update residual by Eq. (11), and compute gphys by Eq. (12).
5. Compute cyber/protocol score scyber by Eq. (14) and normalize both scores by Eq. (15).
6. Apply the fusion rule in Eq. (16).
7. If the rule fires, raise an alarm and log the interval, both scores, and the oracle branch; otherwise accept the interval.

4 Empirical methodology

4.1 Research questions

RQ1 - Effectiveness. Does the combined physics-informed and cyber/protocol runtime monitor improve attack detection at a controlled false-positive operating point compared with single-oracle and signature baselines?

RQ2 - Complementarity. Do the two runtime oracles detect different telemetry mutation families, and does their fusion preserve the strongest behavior of each oracle?

RQ3 - Efficiency. Is the runtime monitor sufficiently fast for execution at the charging-station edge, considering detection latency and measured processing cost?

4.2 Benchmark generation

Charging sessions are generated from the model in Sect. 3.2. Each session samples battery capacity, charger power, initial state of charge, constant-voltage transition point, and decay parameter from the ranges in Table 3 and integrates the charging trajectory at a one-minute reporting interval. A stochastic model error and a start transient are added so that valid sessions are heterogeneous and are not perfectly matched by the physical oracle. This uncertainty is important because an unrealistically exact physical model would overstate detection performance, particularly for the stealthy mutation.

Table 3 Model and benchmark parameters

Parameter	Symbol	Value
Battery capacity	Cb	40 to 80 kWh
Rated charger power	Pmax	22, 50, 60 kW
Initial state of charge	s0	0.10 to 0.40
Constant-voltage transition	scv	0.75 to 0.85
Decay constant	τ	0.06 to 0.12
Charging efficiency	ηc	0.95
Reporting interval	Δt	1 min
Session length	T	35 to 70 min
Physics-model error	σm	2.5% of Pmax
Residual scale	σr0	1.35 kW
Scaling factor	α	0.55, 0.65, 1.35, 1.50
Bias magnitude	δ	3 to 6 kW
Stealthy factor	αs	0.90 to 0.94
Freshness weight	λf	6
Nameplate limit	Pnp	66 kW

4.3 Data split and mutation injection

Each mutation operator in Sect. 3.3 is injected into otherwise valid sessions. A calibration set of 700 normal sessions estimates the residual and cyber-feature distributions. A validation set contains 600 normal sessions and 160 sessions for each of the five mutation families, giving 1400 validation sessions. An independent test set contains 1200 normal sessions and 300 sessions for each mutation family, giving 2700 test sessions. Calibration, validation, and test sets are disjoint and are generated from the same parameter distributions. The full pipeline is seeded, so the numerical results in this article are reproducible under the stated simulation model.

4.4 Baselines and operating point

Four runtime checkers are compared. The signature baseline flags values outside charger nameplate limits. The cyber-only checker uses the score in Eq. (14). The physics-only checker uses the statistic in Eq. (12). The proposed checker fuses both oracles using Eq. (16). To separate detection capability from threshold aggressiveness, the cyber-only and physics-only thresholds are placed at a common 5% false-positive operating point on normal validation data, while the cross-layer thresholds are synthesized under the same budget. This equal-operating-point design follows the general software-testing principle that competing techniques should be compared under a common resource or error budget rather than at arbitrarily favorable configurations [18].

4.5 Metrics and search configuration

Effectiveness is summarized by detection rate and false-positive rate:

$$DR = \frac{TP}{TP+FN}, \quad FPR = \frac{FP}{FP+TN} \tag{20}$$

The F1 score captures the balance between precision and recall:

$$F_1 = \frac{2\,TP}{2\,TP+FP+FN} \tag{21}$$

We also report area under the receiver operating characteristic curve (AUC). Operational efficiency is measured with detection latency, defined as the expected delay from mutation onset to the first alarm:

$$L = E[t_{alarm} - t_{attack}] \tag{22}$$

The particle swarm uses 30 particles for 60 iterations, inertia ω = 0.6, cognitive and social coefficients c1 = c2 = 1.5, a threshold search box from 0.2 to 3.0, penalty λ = 6, and target FPR_tgt = 5%. It converges to ηp = 0.86 and ηc = 1.42. No attack-specific threshold is used at test time.

5 Results

5.1 Normal behavior and oracle sanity check

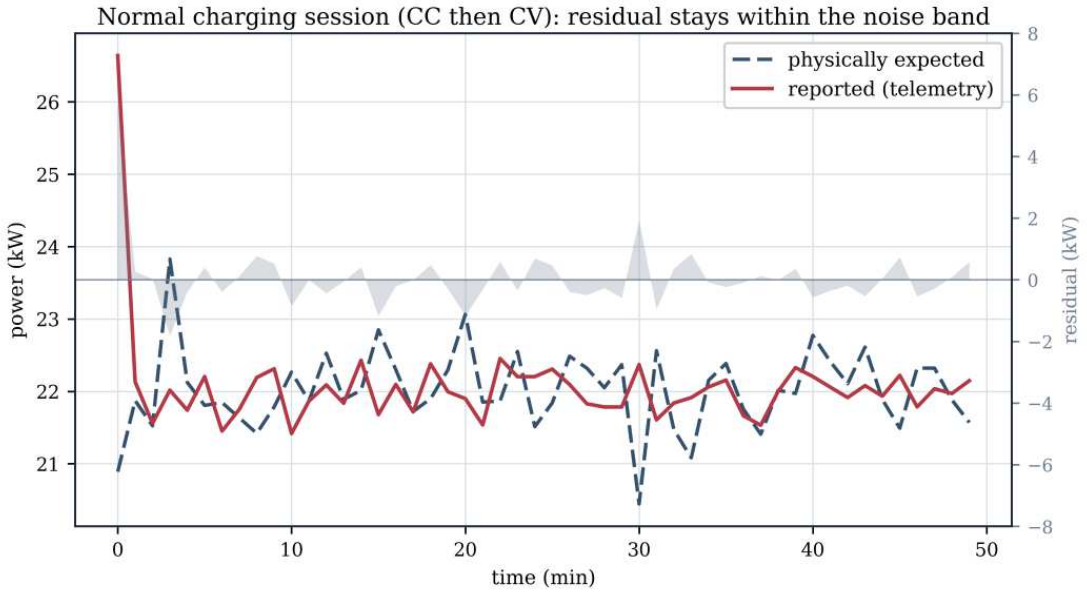


Fig. 5 Representative valid charging trace and physics-oracle residual

Figure 5 shows a representative valid execution. The reported power follows the expected constant-current/constant-voltage trend and the residual remains within the normal noise band. The remaining residual variation is deliberate because the benchmark includes model uncertainty and transient behavior. This sanity check establishes that the physical oracle is not an exact duplicate of the generator and therefore cannot achieve perfect separation by construction.

5.2 RQ1: Overall verification effectiveness

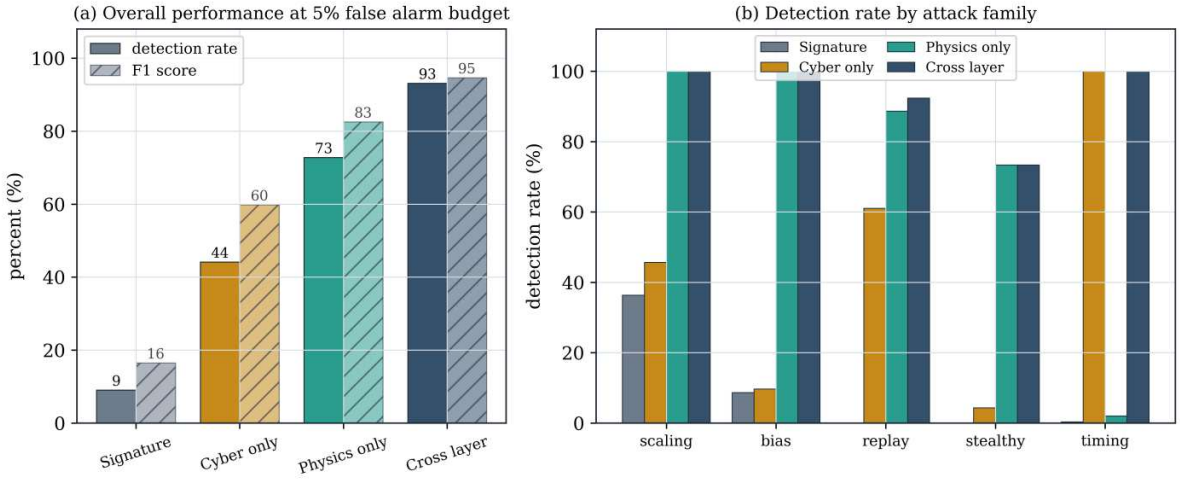


Fig. 6 Overall and mutation-wise verification performance at the 5% false-positive budget

Table 4 and Fig. 6 answer RQ1. The signature baseline detects only 9.1% of mutated sessions. The cyber-only checker reaches 44.1% detection with 4.3% false positives, while the physics-only checker reaches 72.8% with 4.4% false positives. The proposed cross-layer runtime monitor reaches 93.1% detection with 4.6% false positives and an F1 score of 94.6%. Relative to the stronger single oracle, the combined monitor improves detection by 20.3 percentage points while remaining inside the common 5% false-positive budget.

Table 4 Test-set performance at the common 5% false-positive operating point

Checker	Detection rate (%)	False-positive rate (%)	F1 score (%)	AUC
Signature	9.1	1.3	16.5	n.a.
Cyber only	44.1	4.3	59.8	0.691
Physics only	72.8	4.4	82.6	0.845
Cross-layer (proposed)	93.1	4.6	94.6	0.953

Fig. 7 Receiver operating characteristic curves of the single-oracle and cross-layer monitors

Figure 7 shows that the cross-layer monitor also dominates the two single-oracle curves across the false-positive range. Its AUC is 0.953, compared with 0.845 for the physics-only monitor and 0.691 for the cyber-only monitor. The improvement is therefore not restricted to the selected operating point. RQ1 is answered positively for the simulated benchmark: the independent physical oracle and protocol oracle provide substantially stronger runtime verification when fused than when used separately.

5.3 RQ2: Complementarity across telemetry mutations

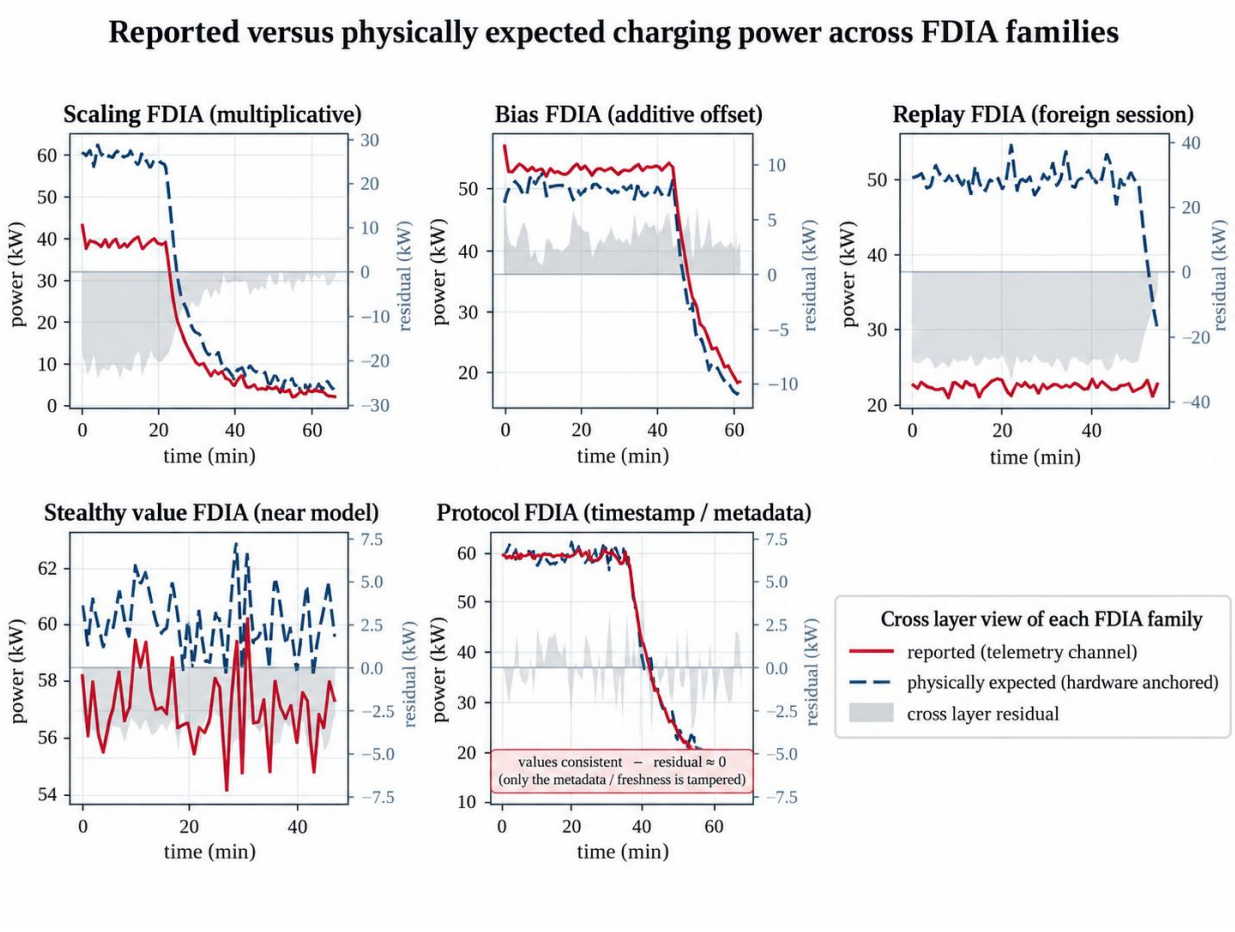


Fig. 8 Representative telemetry traces under the five mutation operators

Figure 8 illustrates why the oracles are complementary. Scaling, bias, and replay produce a discrepancy between reported power and the physical reference. The stealthy mutation keeps the discrepancy close to the normal model uncertainty and is therefore the most difficult value mutation. The protocol-timing mutation is different: its reported power remains physically correct, so a physical residual alone should not detect it. The cyber/protocol oracle is required because the inconsistency is in freshness metadata rather than in power magnitude.

Table 5 Detection rate by telemetry mutation family at the 5% false-positive operating point

Checker	Scaling	Bias	Replay	Stealthy	Timing
Signature	36.3	8.7	0.0	0.0	0.3
Cyber only	45.7	9.7	61.0	4.3	100.0
Physics only	100.0	100.0	88.7	73.3	2.0
Cross-layer (proposed)	100.0	100.0	92.3	73.3	100.0

Table 5 answers RQ2. The physics oracle perfectly detects scaling and bias and detects 88.7% of replay mutations, but it detects only 2.0% of timing mutations. The cyber/protocol oracle shows the complementary pattern: it detects all timing mutations but is weak on stealthy value manipulation. The fused monitor never performs below the better single oracle on any mutation family and improves replay detection to 92.3%. The result supports the design premise that oracle diversity, not a complicated fusion model, is the main source of the performance gain.

5.4 RQ3: Detection latency and processing cost

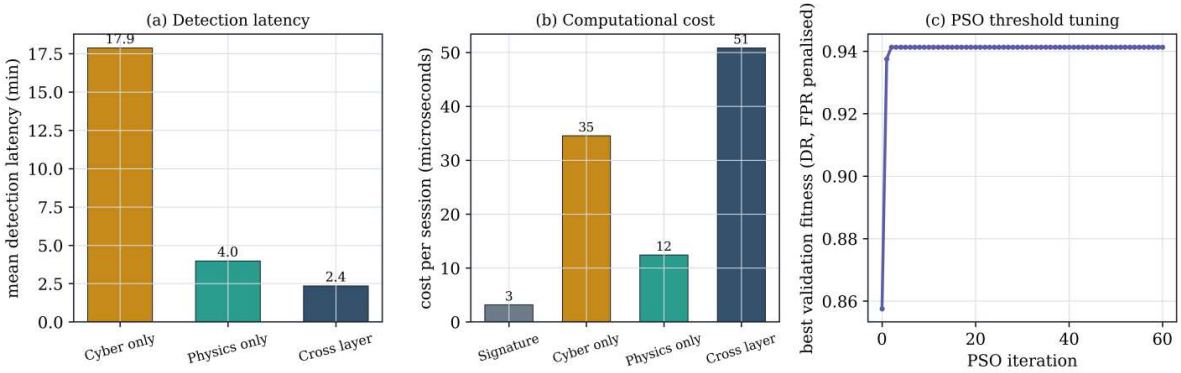


Fig. 9 Detection latency, measured processing cost, and PSO convergence

Table 6 Detection latency and measured processing cost

Checker	Latency (min)	Cost (μs/session)	Relative cost
Signature	n.a.	3.2	1.0
Cyber only	17.88	34.5	10.8
Physics only	3.98	12.4	3.9
Cross-layer (proposed)	2.36	50.8	15.9

The proposed monitor has the shortest mean detection latency, 2.36 min, because the decision rule fires as soon as either oracle provides sufficient evidence. Its measured processing cost is 50.8 μs per session. This cost is higher than the single-oracle baselines but remains several orders of magnitude below the one-minute reporting interval used in the benchmark. Figure 9 also shows rapid convergence of the two-parameter search. RQ3 is therefore answered positively for the reported implementation and workload: the monitor is computationally compatible with edge execution under the tested conditions.

6 Discussion

6.1 Implications for automated software engineering

The main software-engineering contribution is the separation of the runtime oracle from the software artifact being checked. OCPP MeterValues are software outputs. If a detector derives all of its evidence from those same outputs, a plausible forgery can remain internally consistent. The hardware-anchored oracle introduces an independent evidence source without requiring a large learned model. This is directly relevant to automated assurance of cyber-physical software, where physical sensors and actuators can serve as test oracles for software-visible state.

The second implication concerns calibration. Runtime monitors often contain thresholds that are manually selected and difficult to reproduce. Here, the two thresholds are synthesized automatically under an explicit false-positive constraint. The search problem is intentionally small, but it demonstrates a useful pattern: search-based software engineering can be applied not only to test generation or program improvement, but also to operational configuration of verification logic when the objective and constraints are measurable.

The third implication is interpretability. Each alarm is associated with the oracle branch that fired. This gives an operator a concrete reason such as stale metadata, impossible reported power, or disagreement with the physical charging model. Such traceability is easier to inspect than a single opaque anomaly score and is useful when the monitor must be integrated into operational incident handling or regression testing.

6.2 Threats to validity

Construct validity. The five telemetry mutations capture important value and protocol-integrity failure modes, but they are abstractions of real attacks rather than a complete OCPP vulnerability taxonomy. Detection rate therefore measures effectiveness against these defined mutation operators, not against every possible compromise. The 5% false-positive operating point is useful for controlled comparison but may not match the tolerance of a specific deployment.

Internal validity. Calibration, validation, and test sets are disjoint, which reduces direct data leakage, and the evaluation is seeded. However, all sets are generated from the same family of simulation models. A modeling assumption that systematically favors the physical oracle could influence all three splits. The fixed 0.6/0.4 physics-score weighting is not attack-specific, but a broader sensitivity analysis would provide stronger evidence that performance is not tied to this choice.

External validity. The quantitative results come from synthetic charging sessions rather than measured charger traces. The hardware-trust assumption may also vary across products: if an attacker can alter the local meter or state-of-charge source, the independence of the physical oracle is weakened. The current study considers a single-charger view and does not model coordinated fleet attacks, heterogeneous charger firmware, network outages, or vendor-specific OCPP extensions.

Conclusion validity. The benchmark reports one seeded evaluation and does not quantify confidence intervals across repeated seeds. The observed gaps between the proposed monitor and the baselines are large, but repeated randomized experiments and measured datasets are needed before making population-level claims about charging-station deployments.

6.3 Limitations and future work

The immediate next step is external validation with recorded charging traces. ACN-Data provides open charging-session measurements, while CICEVSE2024 provides a recent security-oriented EV charging dataset [34, 35]. A second extension is to strengthen the physical oracle with cumulative energy accounting so that low-amplitude persistent manipulation can accumulate evidence over a whole session. A third extension is fleet-level aggregation of local oracle scores for coordinated-campaign detection. Finally, the threshold-synthesis study should be repeated over multiple seeds and deployment profiles, with sensitivity analysis for model uncertainty and reporting interval.

7 Conclusion

This paper reformulated charging-telemetry integrity checking as an automated runtime verification problem for cyber-physical software. The proposed monitor checks OCPP telemetry against two independent forms of evidence: a protocol/statistical oracle over the reported stream and a physics-informed oracle anchored in charger hardware measurements. A transparent fusion rule combines the two scores, and particle swarm optimization automatically synthesizes the two decision thresholds under a shared false-positive budget. On the seeded benchmark, the cross-layer monitor achieved 93.1% detection at a 4.6% false-positive rate with AUC 0.953, outperforming both single-oracle baselines. It also reduced mean detection latency to 2.36 min while requiring 50.8 μs of processing per session. The results support the use of independent physical oracles and search-based calibration as lightweight mechanisms for automated assurance of cyber-physical telemetry. The main evidence gap is external validity: measured charging-station data and repeated multi-seed studies are required before deployment-level claims can be made.

Statements and Declarations

Funding: The authors received no financial support for the research, authorship, and/or publication of this article.

Data availability: The quantitative results reported in this article are generated from the seeded simulation protocol and parameter ranges described in Sect. 4; no external measured dataset is used for the reported test-set results. Public datasets cited in Sect. 6.3 are proposed for future external validation.

References

[1] Peltomäki, J., Porres, I.: Requirement falsification for cyber-physical systems using generative models. Autom. Softw. Eng. 32, 33 (2025). https://doi.org/10.1007/s10515-025-00503-x

[2] Meyer, M.-A., Zouari, M., Bannenberg, S. et al.: Machine-readable specification and intelligent cloud-based execution of logical test cases for automated driving functions. Autom. Softw. Eng. 32, 10 (2025). https://doi.org/10.1007/s10515-024-00481-6

[3] Dakhama, A., Even-Mendoza, K., Langdon, W. et al.: Enhancing search-based testing with LLMs for finding bugs in system simulators. Autom. Softw. Eng. 32, 63 (2025). https://doi.org/10.1007/s10515-025-00531-7

[4] Mazkatli, M., Monschein, D., Armbruster, M. et al.: Continuous integration of architectural performance models with parametric dependencies - the CIPM approach. Autom. Softw. Eng. 32, 54 (2025). https://doi.org/10.1007/s10515-025-00521-9

[5] Garofalaki, Z., Kosmanos, D., Moschoyiannis, S., Kallergis, D., Douligeris, C.: Electric vehicle charging: a survey on the security issues and challenges of the Open Charge Point Protocol (OCPP). IEEE Commun. Surv. Tutor. 24 (2022). https://doi.org/10.1109/COMST.2022.3184448

[6] Antoun, J., Kabir, M.E., Moussa, B., Atallah, R., Assi, C.: A detailed security assessment of the EV charging ecosystem. IEEE Netw. 34(3), 200-207 (2020)

[7] Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. ACM Trans. Inf. Syst. Secur. 14(1), Article 13 (2011). https://doi.org/10.1145/1952982.1952995

[8] Kosut, O., Jia, L., Thomas, R.J., Tong, L.: Malicious data attacks on the smart grid. IEEE Trans. Smart Grid 2(4), 645-658 (2011)

[9] Liang, G., Zhao, J., Luo, F., Weller, S.R., Dong, Z.Y.: A review of false data injection attacks against modern power systems. IEEE Trans. Smart Grid 8(4), 1630-1638 (2017)

[10] Reda, H.T., Anwar, A., Mahmood, A.: Comprehensive survey and taxonomies of false data injection attacks in smart grids: attack models, targets, and impacts. Renew. Sustain. Energy Rev. 163, 112423 (2022)

[11] ElKashlan, M., Elsayed, M.S., Jurcut, A.D., Azer, M.: A machine learning-based intrusion detection system for IoT electric vehicle charging stations. Electronics 12(4) (2023)

[12] Girdhar, M., Hong, J., Lee, H., Song, T.-J.: Hidden Markov models-based anomaly correlations for the cyber-physical security of EV charging stations. IEEE Trans. Smart Grid 13(5), 3903-3914 (2022)

[13] Kern, D., Krauss, C., Hollick, M.: Detection of anomalies in electric vehicle charging sessions. In: Proceedings of the 39th Annual Computer Security Applications Conference, pp. 298-309 (2023). https://doi.org/10.1145/3627106.3627127

[14] Jahangir, H., Lakshminarayana, S., Poor, H.V.: Charge manipulation attacks against smart electric vehicle charging stations and deep learning-based detection mechanisms. IEEE Trans. Smart Grid 15(5), 5182-5194 (2024)

[15] Li, Y., Chen, G., Dong, Z.: Multi-view graph contrastive representative learning for intrusion detection in EV charging station. Appl. Energy 385, 125439 (2025)

[16] Ramul, A.R., Shahraki, A.S., Bachache, N.K., Sadeghi, R.: Cyberspace enhancement of electric vehicle charging stations in smart grids based on detection and resilience measures against hybrid cyberattacks: a multi-agent deep reinforcement learning approach. Energy 325, 136038 (2025). https://doi.org/10.1016/j.energy.2025.136038

[17] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942-1948 (1995)

[18] Poth, A., Rrjolli, O., Arcuri, A.: Technology adoption performance evaluation applied to testing industrial REST APIs. Autom. Softw. Eng. 32, 5 (2025). https://doi.org/10.1007/s10515-024-00477-2

[19] Xie, L., Mo, Y., Sinopoli, B.: Integrity data attacks in power market operations. IEEE Trans. Smart Grid 2(4), 659-666 (2011)

[20] Musleh, A.S., Chen, G., Dong, Z.Y.: A survey on the detection algorithms for false data injection attacks in smart grids. IEEE Trans. Smart Grid 11(3), 2218-2234 (2020)

[21] Manandhar, K., Cao, X., Hu, F., Liu, Y.: Detection of faults and attacks including false data injection attack in smart grid using Kalman filter. IEEE Trans. Control Netw. Syst. 1(4), 370-379 (2014)

[22] Ashrafuzzaman, M., Das, S., Chakhchoukh, Y., Shiva, S., Sheldon, F.T.: Detecting stealthy false data injection attacks in the smart grid using ensemble-based machine learning. Comput. Secur. 97, 101994 (2020). https://doi.org/10.1016/j.cose.2020.101994

[23] Yang, L., Zhang, X., Li, Z., Li, Z., He, Y.: Detecting bi-level false data injection attack based on time series analysis method in smart grid. Comput. Secur. 96, 101899 (2020)

[24] Han, Y., Feng, H., Li, K., Zhao, Q.: False data injection attacks detection with modified temporal multi-graph convolutional network in smart grids. Comput. Secur. 124, 103016 (2023)

[25] Wang, C., Tindemans, S., Pan, K., Palensky, P.: Detection of false data injection attacks using the autoencoder approach. In: Proceedings of the International Conference on Probabilistic Methods Applied to Power Systems (PMAPS) (2020)

[26] Acharya, S., Dvorkin, Y., Pandžić, H., Karri, R.: Cybersecurity of smart electric vehicle charging: a power grid perspective. IEEE Access 8 (2020)

[27] Sayed, M.A., Atallah, R., Assi, C., Debbabi, M.: Electric vehicle attack impact on power grid operation. Int. J. Electr. Power Energy Syst. 137, 107784 (2022)

[28] Kabir, M.E., Ghafouri, M., Moussa, B., Assi, C.: A two-stage protection method for detection and mitigation of coordinated EVSE switching attacks. IEEE Trans. Smart Grid 12(5), 4377-4388 (2021)

[29] Sepehrzad, R., Khodadadi, A., Adinehpour, S., Karimi, M.: A multi-agent deep reinforcement learning paradigm to improve the robustness and resilience of grid connected electric vehicle charging stations against the destructive effects of cyber-attacks. Energy 307, 132669 (2024)

[30] Guo, L., Ye, J., Yang, B.: Cyberattack detection for electric vehicles using physics-guided machine learning. IEEE Trans. Transp. Electrif. 7(3), 2010-2022 (2021)

[31] Rigas, E.S., Karapostolakis, S., Bassiliades, N., Ramchurn, S.D.: EVLibSim: a tool for the simulation of electric vehicles' charging stations using the EVLib library. Simul. Model. Pract. Theory 87, 99-119 (2018). https://doi.org/10.1016/j.simpat.2018.06.007

[32] Epiphaniou, G., Hammoudeh, M., Yuan, H., Maple, C., Ani, U.: Digital twins in cyber effects modelling of IoT/CPS points of low resilience. Simul. Model. Pract. Theory 125, 102744 (2023). https://doi.org/10.1016/j.simpat.2023.102744

[33] Jayaratne, D.N.D., Kamtam, S.H., Shaikh, S.A., Ramli, M.A., Lu, Q., Mepparambath, R.M., Nguyen, H.N., Rakib, A.: A simulation framework for automotive cybersecurity risk assessment. Simul. Model. Pract. Theory 136, 103005 (2024). https://doi.org/10.1016/j.simpat.2024.103005

[34] Lee, Z.J., Li, T., Low, S.H.: ACN-Data: analysis and applications of an open EV charging dataset. In: Proceedings of the ACM International Conference on Future Energy Systems (e-Energy) (2019)

[35] Buedi, E.D., Ghorbani, A.A., Dadkhah, S., Ferreira, R.L.: Enhancing EV charging station security using a multi-dimensional dataset: CICEVSE2024. In: Data and Applications Security and Privacy XXXVIII, Lecture Notes in Computer Science, pp. 171-190. Springer (2024)

[36] Mo, Y., Sinopoli, B.: Secure control against replay attacks. In: Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing, pp. 911-918 (2009)

[37] Abur, A., Gómez Expósito, A.: Power System State Estimation: Theory and Implementation. Marcel Dekker, New York (2004)

[38] Gao, L., Chen, B., Yu, L.: Fusion-based FDI attack detection in cyber-physical systems. IEEE Trans. Circuits Syst. II Express Briefs 67(8), 1487-1491 (2020)