GOOSEIDRS: Intrusion Detection and Automated Response for IEC61850 GOOSE Masquerade Attacks

Hermenegildo da Conceição Alberto
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
University Gaston Berger (UGB)
Saint Louis, Senegal
hermenegildo.alberto@kit.edu

Gustavo Sánchez
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
gustavo.sanchez@kit.edu

Jean Marie Dembel
University Gaston Berger (UGB)
Saint Louis, Senegal
jeanmarie.dembele@ugb.edu.sn

Idy Diop
Ecole Supérieure Polytechnique (ESP/UCAD)
Dakar, Senegal
idy.diop@esp.sn

Ghada Elbez
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
ghada.elbez@kit.edu

Veit Hagenmeyer
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
veit.hagenmeyer@kit.edu

Abstract

Advanced cyber-attacks on smart grids expose limitations of conventional defenses, particularly for detecting protocol-compliant masquerading and supporting timely response. This paper presents GOOSEIDRS, a reproducible intrusion detection and response framework for the IEC 61850 GOOSE protocol. To support reproducible training and evaluation, we use a protocol-aware augmentation pipeline that extends limited Generic Object Oriented Substation Event (GOOSE) captures into long-duration masquerading traces, while Explainable Artificial Intelligence (XAI) supports decision auditing. We engineer a compact feature set centered on State Number (stNum)/Sequence Number (sqNum) semantics with timing context (inter-arrival time, rolling statistics, and violation indicators). Our LightGBM-based IDS operates online with a two-threshold warning/attack logic, achieves over 99% on accuracy, precision, recall, and F1-score, and triggers protocol-compliant corrective GOOSE responses in live HIL experiments. We evaluate the framework against state-dominant masquerading attacks and release the dataset and artifacts for reproducibility[1].

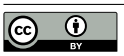
CCS Concepts

• **Security and privacy** → *Intrusion detection systems*; **Network security**.

Keywords

Machine learning, masquerade attack, smart grid, testbed, IEC 61850, digital substation

[1]Dataset: https://doi.org/10.5281/zenodo.19969460. Artifacts: https://doi.org/10.5281/zenodo.20040820.



ACM Sustainability Week Companion '26, Banff, AB, Canada

ACM ISBN 979-8-4007-2199-1/26/06
https://doi.org/10.1145/3765611.3815139

ACM Reference Format:
Hermenegildo da Conceição Alberto, Gustavo Sánchez, Jean Marie Dembel, Idy Diop, Ghada Elbez, and Veit Hagenmeyer. 2026. GOOSEIDRS: Intrusion Detection and Automated Response for IEC61850 GOOSE Masquerade Attacks. In *ACM Sustainability Week 2026 (ACM Sustainability Week Companion '26), June 22–25, 2026, Banff, AB, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3765611.3815139

1 Introduction

Cyber-attacks on industrial control systems can cause severe physical and economic disruption, as illustrated by incidents such as Stuxnet and attacks on national power grids [3, 26]. Power systems are increasingly exposed because digitalization, distributed generation, and real-time networking enlarge the attack surface.

International Electrotechnical Commission 61850 (IEC61850) is the de facto standard for substation automation, and its GOOSE service supports low-latency multicast signaling for protection and control [24]. In many deployments, however, GOOSE traffic lacks strong source authentication, so an adversary able to observe and inject station-bus traffic can exploit stNum/sqNum semantics to mount protocol-compliant masquerading attacks [4, 24, 31]. Segmentation, VPNs, and authentication remain important, but they do not remove the need for detection and response once the bus is exposed.

Although machine-learning-based Intrusion Detection Systems (IDSs) have been studied, deployment is constrained by two gaps: the lack of public GOOSE datasets with long-duration attack traces [8] and limited explainability, which weakens reproducibility, operator trust, and safety-critical use. In practice, unresolved masquerading can prolong denial-of-control, complicate operator diagnosis, and increase the risk of unstable or unsafe switching actions. Prior work also shows that detection without timely mitigation is incomplete, motivating integrated intrusion detection and response systems [11]; this same need for traceable, reviewable decisions also connects the problem to emerging high-risk Artificial Intelligence (AI) requirements, including the forthcoming EU Artificial Intelligence Act (EUAIA) [2, 14].

To address these gaps, we present GOOSEIDRS, a reproducible intrusion detection and response framework for IEC61850 GOOSE masquerading attacks. The main contribution is an integrated online detection-and-response loop; augmentation supports reproducible training and evaluation, and XAI supports decision auditing.

Contributions. The main contribution is GOOSEIDRS, an operational IEC61850 GOOSE intrusion detection and response framework for state-dominant masquerading. It is supported by four components:

- An online, publisher-scoped IDS combining `stNum`/`sqNum` tracking, protocol-semantic indicators, Light Gradient Boosting Machine (LightGBM) with monotone constraints, and dual-threshold escalation.
- A GOOSE-compliant response loop that generates corrective state-dominance messages and emits auditable response records for live HIL evaluation.
- A protocol-aware augmentation pipeline that builds a reproducible corpus from limited Hardware-in-the-Loop (HIL) captures for attack generation, training, calibration, and comparison.
- Released dataset, code, and artifacts for reproducibility[2].

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the dataset and HIL testbed. Section 4 presents the methodology. Section 5 reports results. Section 6 discusses limitations and implications. Section 7 concludes the paper.

2 Related Work

This section reviews prior work on attacks, intrusion detection, and response mechanisms for IEC61850 GOOSE communications, with emphasis on masquerading threats and operational constraints. The discussion identifies gaps that motivate a state-aware intrusion detection and response framework.

2.1 Attacks on the GOOSE Protocol

Prior work reports multiple GOOSE attack variants (Table 1); however, hardware-in-the-loop experiments indicate that replay [30, 32, 34, 35], poisoning [5, 28, 33], and flooding [5, 28, 32, 33] did not induce observable misoperation on protection devices under realistic conditions [4]. In contrast, masquerading attacks—injecting protocol-compliant messages while impersonating legitimate publishers—reliably induced misoperation by exploiting `stNum`/`sqNum` semantics and timing constraints [4]. Table 1 shows the different forms of attack discussed in the literature.

2.2 GOOSE Intrusion Detection Methods

In contrast to rule and specification-based IDSs, anomaly-based methods are able to detect zero-day attacks [8]. Previous work has explored anomaly detection approaches ranging from threshold-based alarms [15] to timing irregularities [7] to flag deviations from expected GOOSE behavior [4]. However, these methods often suffer from limited adaptability to novel attack patterns and high false-alarm rates in the presence of legitimate network jitter.

[2]Dataset: https://doi.org/10.5281/zenodo.19969460. Artifacts: https://doi.org/10.5281/zenodo.20040820.

Table 1: Attack variants.

Attack type	Description	Refs
Masquerade / Spoofing	Protocol-compliant messages crafted to impersonate a legitimate publisher.	[4, 5, 28–30, 32, 35]
Replay	Previously captured messages re-transmitted to mimic valid traffic.	[5, 28–30, 32, 34, 35]
High `stNum` / Poisoning	Forged updates with artificially increased `stNum` to disrupt state dominance.	[5, 28, 30, 32–35]
Message injection	Crafted messages inserted into the stream to induce unauthorized state transitions.	[28, 30, 32, 33]
Flooding	High-rate transmission of GOOSE frames to degrade communication and processing.	[5, 28, 32, 33]

2.2.1 *Operational Constraints in Protection and Control Systems.*

GOOSE-based protection systems operate under strict real-time constraints, where additional latency or indiscriminate traffic blocking may disrupt legitimate protection functions. Prior work has shown that security mechanisms for GOOSE must respect tight timing requirements to remain compatible with protection workflows [5, 7]. Consequently, mitigation strategies commonly used in enterprise networks are often unsuitable in substation environments. A consolidated overview of related studies is provided in Appendix B (Table 4).

2.3 Datasets and Data Augmentation

Publicly available datasets for industrial control networks (e.g., SWaT [19]) typically focus on process-level telemetry. To train robust detectors, researchers have begun applying data-augmentation techniques [20, 23, 27] to amplify short captures into longer, more varied traces. These synthetic expansions aim to preserve timing semantics while introducing controlled anomalies; however, reproducibility and open sharing remain significant challenges.

2.4 Machine Learning and Explainable AI

Machine-learning classifiers e.g., gradient-boosting trees [12] have demonstrated strong performance in distinguishing malicious from benign Industrial Control System (ICS) traffic when supplied with carefully engineered features [1]. XAI techniques (e.g., SHapley Additive exPlanations (SHAP) [16]) are increasingly adopted to surface feature contributions at both the model and per-decision level, enabling operators to audit anomaly detections, tune alert thresholds, and comply with emerging transparency regulations [14]. On the attack side, researchers have explored attacks that exploit learned model boundaries via XAI in smart grid use-cases [22, 25], underscoring the need for defense strategies that account for adaptive threat behavior.

2.5 Automated Intrusion Remediation for GOOSE

Beyond detection, effective defense in critical infrastructures demands rapid, automated response mechanisms that can neutralize threats within protocol timing constraints. Existing work focuses on GOOSE attacks' vulnerability and impact analysis, without addressing remediation [10, 24, 31].

2.6 Summary and Research Gap

Overall, prior GOOSE security research focuses mainly on detection with limited datasets and seldom evaluates attacker impact on real protection devices. Consequently, state-aware detection and automated, protocol-compliant remediation of masquerading attacks under realistic timing constraints remain insufficiently explored.

3 Dataset and HIL Testbed

We follow a bottom-up dataset development approach. We first collect and analyze a multi-hour GOOSE trace from the HIL testbed under normal and masquerading conditions to extract protocol-level invariants in timing and `stNum`/`sqNum` evolution. These observations guide a protocol-aware augmentation framework that scales the dataset while preserving the original system behavior. A dataset summary (source, duration, labels, and splits) is provided in Appendix C (Table 5); labeling policy, reproducibility notes, and dataset-structure details are documented in Appendix C, Sections C.1, C.2, and C.4.

3.1 GOOSE Hardware-in-the-Loop (HIL) Testbed Setup

The experimental evaluation uses an HIL IEC61850 testbed emulating a line feeder protection bay. Bay-level control is implemented using a Siemens SIPROTEC 6MD85 [6, 9], with protection functions provided by multi-vendor relays from Siemens 7SX85, GE Multilin F60, and Hitachi Energy REL670 [13]. Devices exchange trip, status, and interlocking information via GOOSE messages over a shared Ethernet segment.

This configuration supports the network-level masquerading threat model: an adversary injects forged GOOSE messages with stNum set to $\text{stNum}_{\text{legit}} + 1$ and sqNum reset, causing devices to reject subsequent legitimate updates. All traffic is passively mirrored to an out-of-band monitoring node for intrusion detection without affecting protection performance [9, 21]. As illustrated in Figure 1, a network-level adversary injects forged GOOSE messages with elevated `stNum` values, causing subsequent legitimate updates to be rejected.

3.2 Data Augmentation Pipeline

Based on the extracted invariants, the original labeled GOOSE trace is expanded using a protocol-aware augmentation process. Empirical statistics from the testbed capture define inter-message timing, post-state-change burst behavior, and maximum sequence lengths for legitimate and masqueraded traffic. Synthetic GOOSE streams are generated to preserve these temporal and semantic properties. Masquerading attacks are injected under constrained conditions: forged messages start at a higher state number, occur in short one- or two-state blocks, and terminate once legitimate traffic regains state dominance, ensuring protocol compliance.

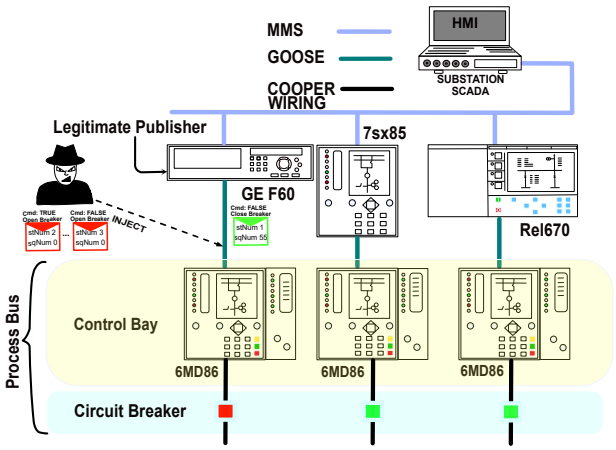


Figure 1: GOOSE masquerading attack scenario in a line feeder protection bay.

In this paper, augmentation supports data availability, calibration, and model comparison. It expands the dataset to over one million messages, increases coverage of benign burst and steady-state retransmission regimes, provides repeated state-dominant masquerade intervals for threshold tuning, and creates a reproducible substrate for comparing the evaluated detectors. An overview of the augmentation pipeline is shown in Figure 2.

4 Methodology for GOOSEIDRS

This section presents the methodology of our intrusion detection and response framework. We describe the system overview and design goals, summarize the GOOSE semantics used for feature extraction (formalized in Appendix A), and detail the detection and remediation workflow.

4.1 System Overview

This work proposes GOOSEIDRS, a passive intrusion detection and response framework for IEC61850 GOOSE communications, targeting masquerading attacks that induce semantic misoperation in protection systems. Operating on mirrored traffic, it performs protocol-aware feature extraction, state-aware detection, automated response, and audit logging. Since GOOSE freshness is inferred from monotonic state/sequence evolution—stNum for state changes and sqNum for retransmissions—plus timing constraints [7, 24], the framework is designed for four goals: semantic awareness of `stNum`/`sqNum` dynamics, real-time compatibility with protection constraints, protocol-compliant response generation, and auditable detection/response decisions. Operationally, the IDS runs on an out-of-band monitoring node or substation gateway fed by mirrored traffic rather than inside relay firmware, which preserves the primary protection path and avoids assuming spare compute budget on field Intelligent Electronic Devices (IEDs). On attack detection, it injects corrective GOOSE messages to restore state consistency without blocking or delaying legitimate traffic.

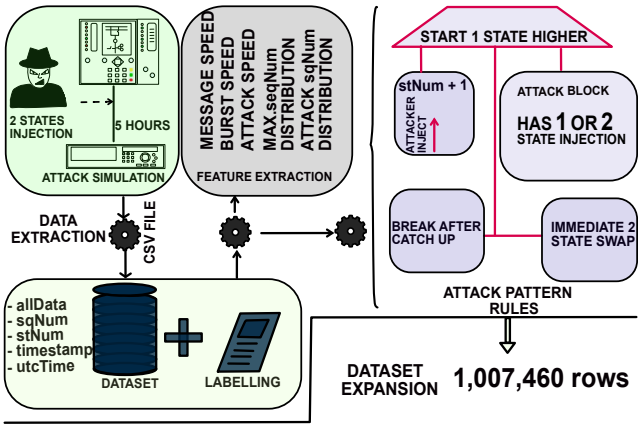


Figure 2: Protocol-aware data augmentation pipeline for IEC61850 GOOSE traffic.

4.2 Threat Model and Attacker Assumptions

Based on subsection 2.1, we focus on network-level GOOSE masquerading. The adversary can passively observe process-bus traffic and inject forged but protocol-compliant frames that manipulate stNum/sqNum to impersonate a legitimate publisher [24]. This models a realistic post-perimeter attacker that reaches the process bus through a compromised engineering workstation, or a maintenance asset. The attack succeeds when a forged publication sets $\text{stNum} = \text{stNum}_{\text{legit}} + 1$, so subscribers accept it as fresher and later reject legitimate updates as stale. Device compromise and access to cryptographic keys are out of scope because we isolate the minimum network-level capability needed to produce this effect. The protocol semantics used in this model are summarized in Appendix A.

Recent IEC61850 dataset and threat-modeling studies suggest that such attacks are difficult both to synthesize faithfully and to detect, because they preserve much of the protocol's normal syntax, timing, and state behavior [28].

4.2.1 Two-State Masquerading Attack Model. The attack proceeds in two phases. First, the attacker injects a forged breaker-open publication with $\text{stNum} = \text{stNum}_{\text{legit}} + 1$, sqNum = 0, and the boolean command set to `True`; in the absence of authentication, subscribers accept it [4]. Second, the attacker sustains state dominance by retransmitting forged frames and then issuing a breaker-close command with a further state increment. When the legitimate publisher later transmits an update, its stNum is lower, so subscribers discard it as stale, creating a denial-of-control condition until a corrective state is restored. Appendix D details the observation and injection procedure used in our experiments.

4.3 Intrusion Detection Methodology

Our intrusion detector GOOSEIDRS builds on [4], which showed that spoofed GOOSE messages can preserve protocol structure while manipulating state evolution. Relative to prior GOOSE IDSs, the contribution is not the use of LightGBM alone, but its adaptation to online GOOSE operation through publisher-scoped state tracking, streaming `stNum`/`sqNum` differentials, timing guards learned from normal traffic, monotone constraints aligned with protocol semantics, and two-threshold escalation from warning to attack decisions. LightGBM is chosen for strong tabular ICS performance under low-latency constraints [1, 12].

For each publisher p (identified by `gocbRef`), the IDS maintains a lightweight streaming state with the latest timestamp, `stNum`, `sqNum`, and short/long histories ($w = 10, 100$). From this state, it computes protocol-semantic and timing features online: step-wise deltas, modulo-256 sequence differences, wrap-around flags, rolling statistics, and `allData`-based command-consistency indicators. Together, these features capture anomalies in state transitions, sequence progression, timing, and persistence.

Packets are processed in arrival order and classified independently. For each message i, the model outputs an attack probability $\hat{y}_i \in [0, 1]$. The IDS then applies two thresholds, early-warning τ_e and attack τ ($0 \le \tau_e < \tau \le 1$): scores in $[\tau_e, \tau)$ trigger a *WARNING*, and scores above τ trigger an *ATTACK* with automated remediation. Alert records keep recent score trajectories to show escalation from nominal to suspicious behavior.

4.3.1 Feature Engineering. Feature engineering extends by jointly modeling GOOSE state semantics [4], timing behavior [8, 31], protocol conformance, and persistence within a unified per-publisher representation. The resulting features are grouped into six families summarized in Table 6 (Appendix C). All features are computed online in a single pass using only the current packet and maintained publisher state, preserving low overhead and real-time applicability. The complete feature schema is available in the accompanying repository[3].

4.3.2 Training and Threshold Selection. We train LightGBM using a stratified random split by label, with 30% of samples reserved for testing and 20% of the remaining 70% used for validation, yielding an overall 56/14/30 train/validation/test split (`random_state=42`). Timing guard baselines are estimated only from normal training samples ($y = 0$) to avoid leakage, then applied unchanged at validation and test time. The final decision threshold is selected on the validation set according to the deployment policy (e.g., target recall, bounded FPR, or maximum F1) and fixed for test evaluation. For transfer to a new substation, the same procedure is repeated using a clean local baseline and validation traffic; a practical calibration workflow is outlined in Appendix H.

4.3.3 Explainable AI (XAI). Explainability is integrated to support operator trust, forensic analysis, and dataset validation. In deployment, the online prototype provides lightweight real-time explanations by mapping activated violation indicators (e.g., backward `sqNum`, non-unit jumps, and state-change-without-command flags) to human-readable alerts. TreeSHAP [16] is used post-hoc in the evaluation pipeline to compute per-event feature attributions for auditing, reporting, and dataset validation.

Explanations are computed over an *event window* around protocol transitions (Δ`stNum`$\neq 0$, `sqNum`≤ 1), covering 10 packets before and 20 after the boundary. Within this window, protocol-aware anchors highlight violations such as backward or non-unit `sqNum`

[3] https://doi.org/10.5281/zenodo.20040820

changes, state changes without command, and consistency breakdowns.

4.3.4 Model Selection with LightGBM as the Primary Detector. Our deployment uses LightGBM as the primary online IDS. It operates on a fixed 50-feature schema (44 engineered protocol features and 6 timing guard features) while preserving packet order for temporal consistency. Domain knowledge is encoded through monotone constraints: normal-behavior features reduce attack likelihood, whereas protocol-violation features increase it. Publisher-specific timing baselines are learned only from normal training traffic and reused unchanged at validation and test time to prevent leakage. Decision thresholds are policy-driven (e.g., F1, target recall, or bounded false-positive rate). Combined with deterministic training and early stopping, this yields strong accuracy, interpretability, and sub-millisecond model-inference latency for resource-constrained Supervisory Control and Data Acquisition (SCADA) gateways and monitoring nodes.

For comparison, Random Forest (RF) and Extreme Gradient Boosting (XGBoost) are trained on a compact 6-feature subset (`allData`, `stNum`, `sqNum`, `time_delta`, `stNum_diff`, `sqNum_diff`) with cross-chunk state preservation.

Other ML and neural approaches in prior IEC61850 studies can achieve competitive accuracy [31], but they often impose higher inference cost and lower interpretability in real-time protection settings [1]. We include them as comparative baselines, while our deployment focus remains on low-latency and interpretable detection (see Section 5.1).

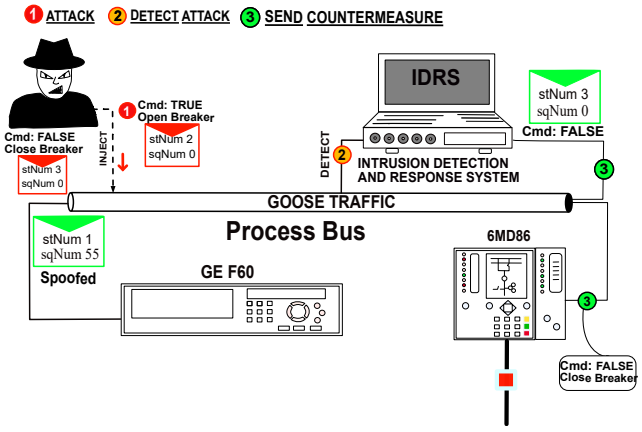


Figure 3: GOOSEIDRS: Bounded operator-in-the-loop GOOSE remediation following spoofed breaker control.

4.3.5 Bounded Operator-in-the-Loop Response. Because a persistent network-level adversary can continuously retransmit forged GOOSE frames, manual remediation may be too slow in practice [18]. When the IDS raises an *ATTACK* decision ($\hat{y}_i \geq \tau$), the framework generates a protocol-compliant corrective GOOSE response with a higher state number (e.g., from `stNum` = 2, `sqNum` = 0 to `stNum` = 3, `sqNum` = 0), changing only the dynamic fields needed to restore legitimate state consistency.

This bounded, operator-in-the-loop response preserves control stability while the attack persists and gives operators time to restore legitimate states via SCADA. In the current Python prototype, however, the audit record is generated before the call that sends the corrective packet, so the logged *Countermeasure Time* is not a clean packet-sent timestamp. We therefore report live response behavior in Section 5.3 as a detect-to-response-record interval rather than as deterministic wire-time remediation latency. Detection and remediation events are exported as Security Information and Event Management (SIEM) records for auditability and regulatory alignment in high-risk critical-infrastructure AI deployments [2, 14]. The workflow is shown in Figure 3.

5 Results and Analysis

5.1 Evaluation

The proposed LightGBM-based GOOSEIDRS is evaluated live on the IEC61850 HIL testbed [9, 13] for real-time GOOSE masquerade detection. Comparative results for alternative models are obtained offline on 1,007,460 protocol-aware augmented GOOSE messages. This evaluation also separates the framework by function: augmentation supplies reproducible coverage for calibration and model comparison, protocol-aware feature engineering provides the separability needed for detection, and the response layer evaluates mitigation after an alert rather than classification accuracy alone.

We follow a progressive model-complexity pipeline. A minimalist **Random Forest** baseline uses six features (`stNum`, `sqNum`, `allData`, Δt, Δ_{stNum}, Δ_{sqNum}) with cross-chunk state continuity, achieving F1 = 99.89%; differential counters contribute 84% of feature importance. We then expand to an **optimized 23-feature set** by adding rolling-window statistics, protocol-violation flags (e.g., backward steps, non-unit jumps, unauthorized state changes), and consistency metrics for expected GOOSE behavior.

The **LightGBM-based GOOSEIDRS** further extends to 44 protocol features and six timing guard features, totaling 50 features. Advanced sequence analysis includes circular sequence differences, wrap detection, and long-window (100-packet) persistence metrics, while monotone constraints encode protocol semantics directly into the model. This configuration normalizes benign drift while amplifying adversarial deviations. LightGBM achieves an F1 score of 99.98% and accuracy of 99.99%, with only 7 false positives and 6 false negatives, and sub-millisecond inference suitable for external monitoring gateways; its confusion-matrix counts are summarized in Table 2, while the full matrix visualization is provided in Appendix E (Figure 6).

Computational performance and deployment feasibility. Because event-triggered GOOSE retransmissions can arrive in short bursts, computational overhead is critical for deployment. The online pipeline therefore updates publisher-local counters and rolling statistics in a single pass using only the current frame and maintained state, without packet buffering or batch reprocessing. Appendix F reports a mean per-frame model-inference latency of 0.049 ms for LightGBM and a 149.2 MB RSS memory footprint, compared with 0.445 ms/248.9 MB for XGBoost and 3.307 ms/146.6 MB for RF. These inference figures support deployment on an external

monitoring node or substation gateway attached to mirrored traffic. Live response timing is reported separately below as a logged detect-to-response-record interval rather than a deterministic wire-time remediation latency. We do not claim direct execution inside protection-relay firmware: the current implementation is better aligned with an external cyber appliance than with many low-power IEDs. Direct benchmarking on relay-class hardware remains future work.

Table 2: LightGBM confusion matrix on the test set.

Actual class	Pred. attack	Pred. benign	Total
Attack	50,064 (TP)	6 (FN)	50,070
Benign	7 (FP)	252,161 (TN)	252,168
Total	50,071	252,167	302,238

For comparison, the **Random Forest** yields 99.96% accuracy with 17 false positives and 97 false negatives over 302,238 test samples, while the **XGBoost** classifier, using imbalance-aware scaling, achieves comparable performance (F1 99.98%, accuracy 99.99%) with 13 false positives and 9 false negatives.

5.2 Analysis of Comparative Results

Appendix F (Table 7) shows that all evaluated models exceed 99% F1, but with different operational trade-offs. Among tree-based methods, **LightGBM** achieves the lowest error (7 FP, 6 FN) and 99.99% recall (Fig. 4) by combining monotone constraints with policy-driven thresholding; its event-window explainability (10 packets before, 20 after) also supports forensic analysis. **XGBoost** is close in accuracy (13 FP, 9 FN), while **Random Forest** (17 FP, 97 FN) confirms the strength of the compact 6-feature representation, with differential counters contributing 84.1% of total feature importance.

Neural models capture longer temporal structure but are more sensitive to benign timing variation: **GRU** (144 FP, 37 FN), **Mamba** (116 FP, 27 FN), and **Transformer** (176 FP, 21 FN). Although the **weighted ensemble** achieves the lowest total error, **LightGBM** is selected as the primary deployment model because it provides the best trade-off among single-model accuracy, inference latency, memory footprint, and interpretability.

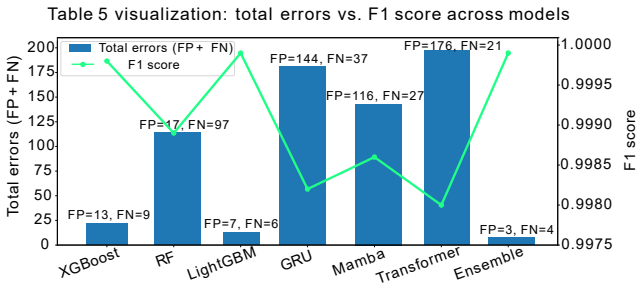


Figure 4: Comparison of evaluated models total errors (FP+FN) and F1 score across evaluated models.

Comparison with prior GOOSE IDSs. Prior IEC61850 GOOSE IDS studies (e.g., Ustun [29, 30], Quincozes [28, 32, 33], Yang [34], and Zaboli [35]) mainly evaluate replay, flooding, or synthetic anomalies rather than state-dominant masquerading that can induce relay misoperation. Unlike most prior work, which typically reports laboratory or trace-driven detection without validating disruption of real IED protection logic, our hardware-in-the-loop evaluation verifies `stNum`/`sqNum`-dominant masquerading against real devices and captures its operational safety implications. Table 3 summarizes differences in hardware validation, masquerade coverage, explainability, and automated remediation.

Table 3: Comparison with prior IEC 61850 GOOSE studies (performance shown when reported).

Work	HIL	Masq.	XAI	Auto resp.	Reported performance
Silveira & Franco [5]	No	Yes	No	No	N/A (attack/impact study)
Quincozes et al. [33]	No	Yes	No	No	Acc 99.4%
Quincozes et al. [32]	No	Yes	No	No	Acc 76.12%, F1 65.84% (GOOSE-only feat.)
Quincozes et al. [28]	No	Yes	No	No	Precision 56.16%, F1 65.96%
Zaboli et al. [35]	N/A	N/A	N/A	N/A	N/A (survey/position)
Alberto et al. [4]	Yes	Yes	No	No	>99% Acc (reported)
GOOSEIDRS	**Yes**	**Yes**	**Yes**	**Yes**	**Acc 99.99%, F1 99.98% (FP=7, FN=6)**

5.3 Remediation Performance

The remediation loop is executed live on our HIL testbed by the external monitoring/response node. We report response timing separately from model-inference latency because the response path includes software reporting, corrective-frame construction, and packet transmission steps. In the current Python prototype, the available audit logs provide a detect-to-response-record interval rather than a hardware timestamp of the corrective frame on the wire.

Across 94 completed live reports, the detect-to-response-record interval had median 7.808 ms, mean 14.149 ms, p95 47.822 ms, and maximum 101.504 ms; 9 reports were at or below 4 ms. Because detection is performed on mirrored traffic, this response path does not insert inline latency into legitimate GOOSE exchanges. These results demonstrate live automated response on a gateway-style deployment, while deterministic sub-4 ms wire-time remediation remains future implementation work.

5.4 Explainable AI (XAI)

Explainability is provided through TreeSHAP attributions for the LightGBM IDS [16], yielding both local and global explanations for masquerade detections. Explainability in the online prototype

is provided via lightweight, rule-based feature triggers for immediate operator feedback; TreeSHAP is used post-hoc for auditing and reporting in the evaluation pipeline. In deployment, we additionally generate lightweight, real-time explanations by mapping activated violation indicators to interpretable protocol conditions (e.g., backward/non-unit `sqNum`, state changes without command, and timing anomalies).

Explanations are computed over an *event window* around protocol transitions (Δ`stNum`$\neq 0$, `sqNum`≤ 1), covering 10 packets before and 20 after the boundary, see Appendix G (Fig. 7). Within this window, protocol-aware anchors highlight violations such as backward or non-unit `sqNum` changes, state changes without command, and consistency breakdowns.

6 Discussion

GOOSE masquerading is consequential because it can induce protection misoperation while remaining syntactically compliant with IEC61850 message semantics. Replay, flooding, and generic anomalies have been widely studied, but they do not reliably reproduce the same effect on real IEDs. By contrast, state-dominant masquerading exploits the `stNum`/`sqNum` freshness logic used by subscribers, making it operationally meaningful while still leaving semantic traces for detection.

In our setting, LightGBM with monotone constraints outperforms the evaluated deep learning models, suggesting that, for protocol-driven GOOSE detection, explicitly encoding protocol semantics can be more effective than relying on representation learning alone. In this study, tree-based models also yield lower false-positive rates because monotonicity, timing consistency, and state progression are encoded explicitly rather than learned implicitly.

Compared with prior GOOSE IDS studies based on simulations, generic features, or offline traces, our contribution is the integration of real-time detection, explainability, and bounded remediation. We validate this reproducible, GOOSE-specific detection-and-response pipeline in a hardware-in-the-loop setting with real protection devices.

6.1 Limitations

Validation was conducted in a controlled laboratory testbed, so latency may increase in larger deployments under heavier background traffic, switching delay, and jitter. Because decisions are made per packet using two thresholds (τ_e, τ), detection delay also depends on the first high-confidence forged packet and selected operating point. Training uses a controlled single-publisher stream, whereas evaluation uses a multi-vendor HIL environment with multiple communicating IEDs and cross-vendor masquerading attacks. Most importantly for deployment, we did not benchmark a native implementation inside vendor IED firmware or other relay-class hardware; the present results therefore support gateway-based deployment, not direct on-IED execution. Portability depends on site-specific recalibration of timing references and decision thresholds, and Appendix H outlines this deployment procedure for new substations.

7 Conclusion and Future Work

GOOSEIDRS addresses practical IEC61850 GOOSE security gaps. It provides a reproducible end-to-end framework for protocol-aware data construction, feature engineering, intrusion detection, explainability, and bounded response. Its main contribution is the integration of these components into a GOOSE-specific detection-and-response pipeline: augmentation constructs a reproducible corpus from limited captures, protocol-aware features support detection, and XAI supports auditing. Using `stNum`/`sqNum` dominance and timing context, the LightGBM-based IDS detects state-dominant publisher-impersonation attacks with accuracy above 99% and low per-frame model-inference latency in benchmarked runs. Live HIL experiments show variable detect-to-response-record intervals in the current Python prototype rather than deterministic sub-4 ms wire-time remediation, so gateway-based real-time detection is supported while remediation timing remains an implementation target. Future work will align the framework with IEC 62351-6 and EUAIA transparency requirements by packaging decision traces, SHAP-based audits, and countermeasure logs as compliance artifacts, and by porting the detector to reduced-footprint relay-class targets for direct computational benchmarking.

Acknowledgments

This work was partially funded by Engineering Secure Systems of the Helmholtz Association (HGF), KASTEL Security Research Labs, structure 46.23.02, and the Regional Scholarship and Innovation Fund (RSIF), the flagship program of PASET, under budget code B8501N10024.

References

[1] Eslam Amer and Tamer Elboghdadly. 2024. Evaluating machine learning techniques for ICS security: Insights from dataset limitations and classifier performance. In *2024 MIUCC*. IEEE, 368–373.
[2] Niklas Bunzel. 2025. Compliance Made Practical: Translating the EU AI Act into Implementable Security Actions. In *2025 IEEE/ACM RAIE*. IEEE, 69–73.
[3] Hermenegildo da Conceição Alberto, Jean Marie Dembele, Idy Diop, and Alassane Bah. 2024. Review of Intrusion Detection Systems for Supervisor Control and Data Acquisition: A Machine Learning Approach. In *Science, Engineering Management and Information Technology*. Springer.
[4] Hermenegildo da Conceição Alberto, Gustavo Sánchez, Jean Marie Dembele, Idy Diop, Ghada Elbez, and Veit Hagenmeyer. 2025. Masquerading IEC 61850 GOOSE Protocol: Cyber-Physical Experiments and Detection. In *e-Energy*. ACM.
[5] Mauricio Gadelha da Silveira and Paulo Henrique Franco. 2019. IEC 61850 Network Cybersecurity: Mitigating GOOSE Message Vulnerabilities. In *6th Annual PAC World Americas Conference*.
[6] Ghada Elbez, H. B. Keller, and Veit Hagenmeyer. 2018. A Cost-Efficient Software Testbed for Cyber-Physical Security in IEC 61850-Based Substations. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 1–6.
[7] Ghada Elbez, Hubert B Keller, and Veit Hagenmeyer. 2019. Authentication of GOOSE messages under timing constraints in IEC 61850 substations. In *6th International Symposium for ICS & SCADA Cyber Security Research 2019*.
[8] Ghada Elbez, Klara Nahrstedt, and Veit Hagenmeyer. 2024. Early attack detection for securing GOOSE network traffic. *IEEE Transactions on Smart Grid* 15, 1 (2024), 899–910. doi:10.1109/TSG.2023.3272749
[9] Ghada Elbez, Gustavo Sánchez, Sine Canbolat, Sophie Corallo, Clemens Fruböse, Florian Lanzinger, and Veit Hagenmeyer. 2025. Insights and Lessons Learned from a Realistic Smart Grid Testbed for Cybersecurity Research. In *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems (ACM e-Energy)*. 805–812.
[10] Ahmed Elmasry. 2024. *Developing and Evaluating Simulation Testbeds for IEC61850 GOOSE Protocol: Enhancing Cybersecurity in Substations and Smart Grids*. Master's thesis. Hamad Bin Khalifa University.
[11] Zakira Inayat, Abdullah Gani, Nor Badrul Anuar, Muhammad Khurram Khan, and Shahid Anwar. 2016. Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications* (2016).

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* (2017).
[13] Gustav Keppler, Andrea Bonetti, Sine Canbolat, Aneeqa Mumrez, Veit Hagenmeyer, and Ghada Elbez. 2024. Interoperability Assessment of IEC 61850 Devices in a Multivendor Digital Substation. In *2024 6th Global Power, Energy and Communication Conference (GPECOM)*. IEEE, 635–640.
[14] Riikka Koulu, Suvi Sankari, Hanne Hirvonen, and Tatjaana Heikkinen. 2023. Artificial intelligence and the law: can we and should we regulate AI systems? In *Research Handbook on Law and Technology*. Edward Elgar Publishing.
[15] YooJin Kwon, Huy Kang Kim, Yong Hun Lim, and Jong In Lim. 2015. A behavior-based intrusion detection technique for smart grid infrastructure. In *2015 IEEE Eindhoven PowerTech*. IEEE, 1–6.
[16] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* (2017).
[17] Tomáš LÁDR. 2019. *IED Simulator with a Support of Industrial Communication GOOSE*. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology.
[18] Georgios Michail Makrakis, Constantinos Kolias, Georgios Kambourakis, Craig Rieger, and Jacob Benjamin. 2021. Vulnerabilities and Attacks Against Industrial Control Systems and Critical Infrastructures. *arXiv:2109.03945* (2021).
[19] Aditya P Mathur and Nils Ole Tippenhauer. 2016. SWaT: A water treatment testbed for research and training on ICS security. In *2016 CySWater*. IEEE.
[20] Rasheed Mohammad, Faisal Saeed, Abdulwahab Ali Almazroi, Faisal S Alsubaei, and Abdulaleem Ali Almazroi. 2024. Enhancing intrusion detection systems using a deep learning and data augmentation approach. *Systems* (2024).
[21] Aneeqa Mumrez, Muhammad M. Roomi, Heng Chuan Tan, Daisuke Mashima, Ghada Elbez, and Veit Hagenmeyer. 2023. Comparative Study on Smart Grid Security Testbeds Using MITRE ATT&CK Matrix. In *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 1–7.
[22] Aneeqa Mumrez, Gustavo Sánchez, Ghada Elbez, and Veit Hagenmeyer. 2023. On Evasion of Machine Learning-based Intrusion Detection in Smart Grids. In *SmartGridComm '23*. IEEE.
[23] Uneneibotejit Otokwala, Andrei Petrovski, and Harsha Kalutarage. 2021. Improving intrusion detection through training data augmentation. In *2021 SIN*. IEEE.
[24] Haftu Tasew Reda, Biplob Ray, Pejman Peidaee, Adnan Anwar, Abdun Mahmood, Akhtar Kalam, and Nahina Islam. 2021. Vulnerability and impact analysis of the IEC 61850 GOOSE protocol in the smart grid. *Sensors* (2021).
[25] Gustavo Sánchez, Ghada Elbez, and Veit Hagenmeyer. 2024. Attacking Learning-based Models in Smart Grids: Current Challenges and New Frontiers. In *e-Energy*.
[26] Gustavo Sanchez, Ghada Elbez, and Veit Hagenmeyer. 2025. A Global Analysis of Cyber Threats to the Energy Sector:"Currents of Conflict" from a geopolitical perspective. *atp magazin* 67, 9 (2025), 56–66.
[27] SungUk Shin, Inseop Lee, and Changhee Choi. 2019. Anomaly dataset augmentation using the sequence generative models. In *2019 ICMLA*. IEEE.
[28] S. E. Quincozes, C. Albuquerque, D. Passos, and D. Mossé. 2024. ERENO: A Framework for Generating Realistic IEC–61850 Intrusion Detection Datasets for Smart Grids. *IEEE Transactions on Dependable and Secure Computing* (2024).
[29] T. S. Ustun, S. M. Farooq, and S. M. S. Hussain. 2019. A Novel Approach for Mitigation of Replay and Masquerade Attacks in Smartgrids Using IEC 61850 Standard. *IEEE Access* (2019).
[30] T. S. Ustun, S. M. S. Hussain, A. Ulutaş, A. Önen, M. M. Roomi, and D. Mashima. 2021. Machine Learning-Based Intrusion Detection for Achieving Cybersecurity in Smart Grids Using IEC 61850 GOOSE Messages. *Symmetry* (2021).
[31] Taha Selim Ustun, SM Suhail Hussain, Ahsen Ulutas, Ahmet Onen, Muhammad M Roomi, and Daisuke Mashima. 2021. Machine learning-based intrusion detection for achieving cybersecurity in smart grids using IEC 61850 GOOSE messages. *Symmetry* (2021).
[32] V. E. Quincozes, S. E. Quincozes, C. Albuquerque, D. Passos, and D. Mossé. 2022. Feature Extraction for Intrusion Detection in IEC-61850 Communication Networks. In *CSNet*. Rio de Janeiro, Brazil.
[33] V. E. Quincozes, S. E. Quincozes, D. Passos, and D. Mossé. 2024. Towards Feature Engineering for Intrusion Detection in IEC–61850 Communication Networks. *Annals of Telecommunications* (2024).
[34] Liqun Yang, You Zhai, Yipeng Zhang, Yufei Zhao, Zhoujun Li, and Tongge Xu. 2022. A New Methodology for Anomaly Detection of Attacks in IEC 61850-Based Substation System. *Journal of Information Security and Applications* (2022).
[35] Aydin Zaboli, Seong Lok Choi, Tai-Jin Song, and Junho Hong. 2024. ChatGPT and Other Large Language Models for Cybersecurity of Smart Grid Applications. *Unpublished manuscript* (2024).

Table 4: Overview of related studies.

Reference	Topic	Year
Ládr [17]	GOOSE protocol	2019
Silveira and Franco [5]	GOOSE, cybersecurity	2019
Ustun et al. [29]	GOOSE, cybersecurity	2019
Alberto et al. [4]	GOOSE, ML, cybersecurity	2025
Zaboli et al. [35]	GOOSE, ML, cybersecurity	2024
Yang et al. [34]	GOOSE, ML, cybersecurity	2022
Ustun et al. [30]	GOOSE, ML, cybersecurity	2021
V. E. Quincozes et al. [33]	GOOSE, feature extraction	2024
V. E. Quincozes et al. [32]	GOOSE, feature extraction	2022
S. E. Quincozes et al. [28]	GOOSE, data generation, cybersecurity	2024

Appendix A — IEC61850 GOOSE Protocol Semantics

The IEC61850 Generic Object Oriented Substation Event (GOOSE) protocol provides high-speed, event-driven publisher–subscriber communication directly over Ethernet. By avoiding TCP/IP, it supports IEC 61850-5 Class P2/P3 protection latencies of about 3 ms [5].

State evolution is encoded by the state number (`stNum`), which increments on application changes, and the sequence number (`sqNum`), which increases across retransmissions of the same state. Each new state triggers immediate publication followed by burst retransmissions with increasing intervals until steady state is reached [28]. Subscribers assess freshness from monotonic `stNum`/`sqNum` progression and timing constraints such as `timeAllowedToLive`.

Because GOOSE lacks native authentication and integrity protection, receivers infer publisher legitimacy from state and timing behavior. An attacker that preserves valid `stNum`/`sqNum` dynamics can therefore inject protocol-compliant masquerade messages, establish state dominance, and suppress later legitimate updates [4, 5].

Appendix B — Overview of Related Studies

This appendix summarizes representative prior studies on IEC61850 GOOSE security, with emphasis on three themes used throughout this paper: protocol analysis, Machine Learning (ML)-based detection, and data/feature engineering. The table is intended as a compact chronology of the field and highlights how recent work has shifted from protocol-focused discussions to data-driven detection and reproducible dataset construction.

Appendix C — Dataset Schema and Labeling Policy

This appendix documents the dataset schema, field definitions, and labeling rules used throughout the augmentation and evaluation pipeline.

C.1 Labeling policy

Labels are assigned per message using injection intervals defined by the controller and validated against GOOSE semantics. A frame is benign if it originates from the legitimate publisher and follows

Table 5: Dataset summary.

Item	Value
Source	HIL IEC61850 testbed with multi-vendor protection relays (Section 3.1)
Raw capture duration	Baseline capture (`autosave_capture.csv`) filtered to `E03A101_PROT/LLN0$DS_GENERAL_TRIP`; augmented to 1110 h (46.25 days)
Number of pcaps / sessions	1 capture session (CSV export)
Number of unique publishers (`gocbRef`)	1 (single dataset stream after filtering)
Number of IED types	Multi-vendor relays (Siemens, GE, Hitachi Energy); Siemens publisher stream used for augmentation
Total messages (augmented)	1,007,460
Attack type(s)	Network-level masquerading / state dominance (`stNum`/`sqNum`)
Number of attack intervals (`attack_id`)	Generated synthetically as 1–2-state blocks with ≥1-state gaps (target: 40% of states)
Attack message ratio	Targeted at ≈40% at the state level (message-level ratio depends on sampled sequence lengths)
Labels included	`label` (0/1), `label_reason`, `attack_id`, `event_id`
Splits used in this paper	56/14/30 train/val/test (Section 3.4, "Training and threshold selection")
Artifacts	Dataset CSV + augmentation code and parameters ($p_{attack} = 0.4$, horizon=1110 h; fixed random seeds for reproducibility)

expected `stNum`/`sqNum` progression and timing behavior, including normal burst retransmissions after state changes. A frame is labeled masquerade if it is attacker-injected and establishes state dominance, i.e., $\texttt{stNum} > \texttt{stNum}_{\text{legit}}$ with `sqNum` reset and monotonic retransmissions consistent with protocol syntax. An attack interval starts at the first injected frame and ends when a legitimate publisher reclaims dominance (its `stNum` exceeds the attacker's highest state) or when injection stops. Frames outside these intervals are benign. Malformed ASN.1 frames, missing fields, or inconsistent publisher identity are excluded from training and reported as `label_reason=invalid_frame`. Timing-only anomalies without state dominance are not labeled as attacks.

C.2 Reproducibility notes

All labels are derived from a deterministic injection schedule and validated by re-parsing the raw captures. The released dataset includes the full attack schedule (`attack_id`, start/end timestamps), enabling third-party recomputation of labels from raw csv files.

C.3 Dataset summary

Table 5 provides a compact summary of the dataset used in this paper. A full data dictionary (field-by-field description) is maintained with the public artifacts.

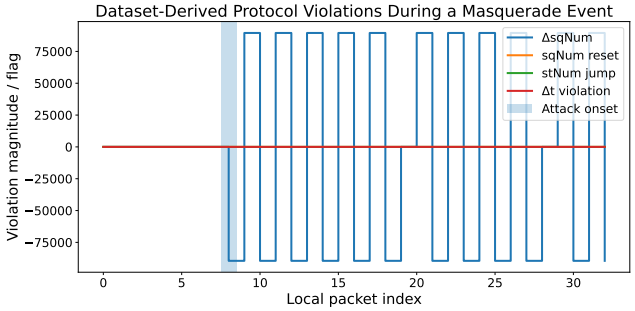


Figure 5: Dataset-derived protocol-semantic violations observed during a masquerade attack window.

Appendix C.4 — Dataset structure

Each record corresponds to one parsed GOOSE frame, ordered by capture time and grouped by publisher identity. The dataset is released in a flat, row-wise format (CSV) with the following field categories:

To illustrate how these fields capture protocol semantics in practice, Fig. 5 shows a dataset-derived view of a single masquerade event. The figure is computed directly from the raw and derived fields listed above (e.g., `stNum`, `sqNum`, `sqNum_diff`, and timing context) and highlights sequence resets, state transitions, and timing deviations observed at attack onset.

C.5 Feature Schema and Rationale

This subsection summarizes the feature-engineering schema used in the released dataset. Table 6 groups features into six families and provides representative examples that link protocol semantics to detection signals used by the model.

Appendix D — Algorithmic Realization of the GOOSE Masquerading Attack

This appendix summarizes the toolchain used to realize the GOOSE masquerading attack in the threat model. On real devices, successful masquerading requires reproducing the legitimate publisher's frame structure and state evolution, so we implement (i) passive observation/dissection and (ii) protocol-compliant injection.

GOOSE observation and injection. A `Scapy`-based dissector passively captures multicast GOOSE frames, filters by EtherType, and decodes the `[APPLICATION 1]` PDU via recursive ASN.1 TLV parsing. It extracts control-block identifiers, dataset references, timestamps, `allData`, and the dynamic variables `stNum`/`sqNum` without interacting with relays. A companion injector uses `Scapy` for Ethernet framing and `PyASN1` for BER encoding, clones static fields, and changes only `stNum` and `sqNum`. The forged sequence sets $\texttt{stNum} > \texttt{stNum}_{i-1}$, resets `sqNum` to zero, and then retransmits with increasing `sqNum` under standard GOOSE timing.

Subscriber impact. Since subscribers validate freshness through monotonic `stNum`/`sqNum` progression, the forged sequence is accepted and establishes state dominance. Subsequent legitimate

Table 6: Feature families, intuition, and representative examples.

Category	Rationale and representative examples
Baseline protocol context	Preserves raw GOOSE semantics and identifiers for calibration and per-publisher tracking (e.g., `stNum`, `sqNum`, `allData`).
State/sequence differentials	Captures non-monotonic evolution and non-unit transitions indicative of state-dominance manipulation (e.g., Δ`stNum`, Δ`sqNum`, wrap-around flags).
Temporal dynamics	Models burst retransmissions and timing deviations relative to learned baselines (e.g., Δt, timing z-scores, baseline exceedance indicators).
Conformance and violation indicators	Encodes protocol-semantic violations of expected state/sequence behavior (e.g., backward/jump flags, unauthorized state-change flags, identity/consistency scores).
Deviation and rolling statistics	Quantifies deviations from typical behavior over short windows (e.g., rolling mean/std, deviation magnitude, normalized scores).
Pattern diversity and persistence	Detects sustained anomalies and repeated patterns over long windows (e.g., anomaly counts over $w = 100$, persistence and diversity metrics).

publications are rejected due to lower `stNum`, leading to denial-of-control until a corrective state is restored by the automated response mechanism.

Appendix E — LightGBM Confusion Matrix Visualization

This appendix provides the confusion-matrix view of the deployed LightGBM detector on the held-out evaluation set. It complements aggregate metrics by explicitly showing true/false positives and true/false negatives, which helps assess the practical trade-off between missed attacks and false alarms in operational settings.

Appendix F — Comparative Model Metrics

This appendix reports a side-by-side comparison of all evaluated models using classification quality (Accuracy, Precision, Recall, Specificity, F1), confusion-matrix counts, and computational cost (latency and memory). Latency is reported as mean model inference time per frame in milliseconds, and memory as approximate runtime footprint in megabytes on the evaluation host. These measurements inform deployment on external monitoring gateways; they are not a direct benchmark of vendor IED firmware.

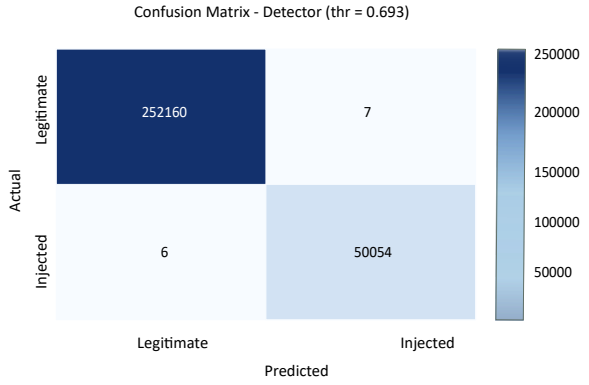


Figure 6: LightGBM confusion-matrix visualization.

Table 7: Comparison of evaluated models. Latency is mean model-inference time per frame after warm-up on the external monitoring host; memory is approximate runtime footprint in MB.

Metric	XGB	RF	LGBM	GRU	Mamba	Trans.	Ens.
Acc.	0.9999	0.9996	1.0000	0.9994	0.9995	0.9993	1.0000
Prec.	0.9997	0.9997	0.9999	0.9971	0.9977	0.9965	0.9999
Rec.	0.9998	0.9981	0.9999	0.9993	0.9995	0.9996	0.9999
Spec.	0.9999	0.9999	1.0000	0.9994	0.9995	0.9993	1.0000
F1	0.9998	0.9989	0.9999	0.9982	0.9986	0.9980	0.9999
TP	50,061	49,973	50,064	50,026	50,036	50,042	50,066
FP	13	17	7	144	116	176	3
FN	9	97	6	37	27	21	4
TN	252,155	252,151	252,161	252,012	252,040	251,980	252,165
Inf. lat.	0.445	3.307	0.049	0.931	n/a	0.450	n/a
RSS mem.	248.9	146.6	149.2	201.8	n/a	203.3	n/a

Appendix G — Event-Window Explainability Example

This appendix illustrates a representative event window used for post-hoc explainability analysis around a detected state transition. The figure shows model confidence evolution before and after the event boundary and provides visual context for how protocol-semantic anomalies accumulate prior to attack confirmation.

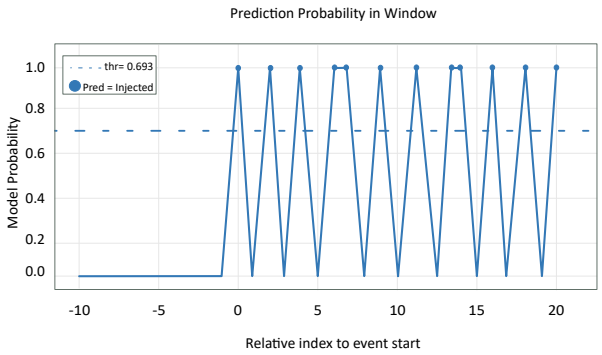


Figure 7: Event window covering 10 packets before and 20 packets after detection.

Appendix H — Deployment Calibration

For deployment in a new substation, calibration proceeds in five steps: (1) collect a clean baseline of normal publisher traffic; (2) estimate publisher-specific timing and sequence-reference statistics from that baseline only; (3) tune decision thresholds on a validation partition according to the target operating policy (e.g., high recall or bounded false-positive rate); (4) verify behavior in shadow mode or controlled replay before activation; and (5) freeze the selected thresholds for online use until a planned recalibration cycle is performed. This procedure makes the static parameters reproducible and portable across installations while preserving a clear separation between training, calibration, and evaluation.