Meeting SLO with Privacy: Partial Homomorphic Encryption Inference and Anomaly Gating for Predictive Autoscaling in Cloud Environments

Alan Chuang
Department of Computer Science
San Jose State University
San Jose, CA, USA
alan.chuang@sjsu.edu

Melody Moh
Department of Computer Science
San Jose State University
San Jose, CA, USA
melody.moh@sjsu.edu

Teng-Sheng Moh
Department of Computer Science
San Jose State University
San Jose, CA, USA
teng.moh@sjsu.edu

***Abstract*—Cloud autoscaling dynamically provisions resources under fluctuating demand, but predictive autoscaling must also meet strict service-level objectives (SLO) while protecting tenant data confidentiality. In this paper we present *SecurePredictScale*, a runtime-secure autoscaling framework that combines DP-LSTM forecasting, a CKKS-encrypted fully connected layer, inline anomaly detection (Isolation Forest) to filter poisoned and OOD inputs, and OPA-enforced RBAC. The design exposes a minimal homomorphic-encryption boundary and quantifies SLO compliance under privacy constraints. On Google Cluster Trace v3 and a containerized testbed, the encrypted FC layer completes in ≈ 35 ms, and the end-to-end API→action loop stays under 50 ms (p95) at up to 82 requests/s. The anomaly gate reaches $\approx 97\%$ TPR at $\approx 1\%$ FPR, and RBAC blocks 100% of unauthorized requests. DP-SGD training achieves composed privacy $\epsilon \approx 9.7$ at $\delta = 10^{-5}$ while keeping forecasting error within operational bounds and outperforming ARIMA and Prophet at comparable privacy budgets, yielding a practical template for privacy-aware autoscaling and other ML-driven control loops in cloud and edge environments.**

***Index Terms*—Edge Computing, Cloud Autoscaling, Homomorphic Encryption, Anomaly Detection, RBAC, Differential Privacy**

I. Introduction

Cloud autoscaling dynamically adjusts compute and storage resources in response to variable workload demands. Traditional reactive schemes scale on current utilization thresholds, so they often lag behind demand spikes and cause service degradation, missed service-level objectives (SLO), or unnecessary over-provisioning. Predictive autoscaling addresses this limitation by forecasting future load using time-series models such as Long Short-Term Memory (LSTM) networks [1], [2] or autoregressive integrated moving average (ARIMA), allowing resources to be provisioned ahead of time.

State-of-the-art predictive systems typically centralize fine-grained per-tenant CPU, memory, I/O, and network logs into a single repository for training and inference. This centralization exposes multiple risks, including: (1) *Membership-Inference Attacks*: An adversary may determine whether a specific tenant's data contributed to the model, such as inferring that Company A's CPU usage pattern was included, potentially exposing sensitive internal processes; (2) *Traffic-Profiling Risks*: Detailed usage logs can reveal tenant behavior and workload patterns, for example indicating proprietary batch processing schedules or peak operation times; and (3) *Regulatory Compliance Violations:* Centralized logs may fail to meet strict data-locality or anonymization requirements, such as GDPR or HIPAA, risking legal and operational consequences.

We assume an honest, yet curious federated aggregator without raw log sharing; the full system and threat model are described in Section III-A.

The proposed system must address three intertwined challenges to provide privacy and security in predictive autoscaling:

1. *Model Utility under Noise:* Differential privacy (DP) must preserve forecast accuracy while mitigating membership inference. Adding DP noise can reduce short-term forecast accuracy, potentially causing over- or under-provisioning.

2. *Decentralized Coordination:* Aggregating model updates across tenants without exchanging raw logs requires robust federated learning (FL) and defenses against stale or poisoned updates. Tenants may update models at different times or submit corrupted data, complicating aggregation.

3. *Secure Inference at Scale:* Fully homomorphic encryption (FHE) must meet a sub-50 ms latency budget for real-time scaling decisions without slowing predictions beyond operational needs.

Prior research addresses these challenges individually, using DP-SGD for centralized training [3], FL frameworks for mobile devices [4], FHE engines for offline inference [5], and policy engines or anomaly detectors for runtime protection [6], [7]. Secure autoscaling work that protects logs typically relies on coarse anonymization or trusted hardware, and existing HE based cloud systems mostly target batch analytics or classification rather than tight control loops. Training-time privacy, federated heterogeneity, and membership-inference robustness are analyzed in detail in our companion work on differentially private federated training for autoscaling forecasters [8].

In this paper, we introduce **SecurePredictScale**, a *unified* framework that integrates five complementary security and

privacy mechanisms into a latency constrained autoscaling pipeline:

1) **Differential Privacy (DP):** Sanitizes local LSTM updates with formal (ϵ, δ) guarantees, preventing individual tenant data from being inferred at training time.
2) **Federated Learning (FL):** Aggregates updates across tenants without exchanging raw data, enabling collaborative learning while keeping logs local.
3) **Fully Homomorphic Encryption (FHE):** Secures inference-time outputs by encrypting only the final fully connected (FC) layer (CKKS) over a plaintext DP-LSTM, exposing a narrow cryptographic boundary that is compatible with a sub-50 ms decision budget.
4) **Anomaly Detection:** Uses Isolation Forests to identify and quarantine poisoned or corrupted sequences at runtime, reducing the impact of poisoning and OOD inputs on scaling decisions.
5) **RBAC:** Enforced via Open Policy Agent to restrict API and data-flow privileges, ensuring that only authorized actions are permitted according to least-privilege principles.

The main contributions are:

- **Runtime-secure autoscaling pipeline:** A modular path from API ingress to orchestrator action with inline anomaly gating and OPA RBAC, designed for cloud-native deployment.
- **Partial-HE inference that meets SLO:** A hybrid design that uses a plaintext LSTM with a CKKS-only final layer, discloses the full HE parameter set, and demonstrates end-to-end p95 latency < 50 ms under moderate concurrency within a realistic autoscaling SLO.
- **Operational evaluation:** A systematic study of throughput under load, scaling decision quality versus a reactive baseline, and the compute cost of DP, FL, HE, anomaly gating, and RBAC, showing that privacy and security overheads are manageable for minute-scale autoscaling control loops.
- **Privacy and security boundary analysis:** An explicit discussion of what is and is not encrypted (DP-protected training, encrypted FC inference, plaintext intermediate activations), and the resulting threat model and limitations for membership inference and poisoning at runtime.

Our goal is not to introduce a new DP, FL, or HE primitive, but to answer a practical question: *how much encryption and runtime hardening is needed to satisfy privacy constraints while still meeting a strict autoscaling SLO*? SecurePredictScale complements the training-time guarantees in our companion paper [8] by providing runtime privacy and security mechanisms, including partial HE inference and inline defenses, for the same autoscaling workload.

The remainder of the paper is organized as follows. The Related Work section reviews privacy-preserving machine learning, federated learning architectures, homomorphic encryption, and runtime defenses, highlighting research gaps. The System Architecture section presents the design, including data preprocessing, DP-LSTM training, federated aggregation, CKKS-based encrypted inference, and runtime anomaly and RBAC modules. The Methodology section details DP noise calibration, federated scheduling, and FHE parameter selection. The Results section reports findings on the Google Cluster Trace v3 (GCT v3) dataset, analyzing accuracy, privacy leakage, inference latency, and defense efficacy. The Discussion and Conclusion section examines trade offs, scalability, deployment insights, limitations, and future directions.

II. Related Work

Predictive autoscaling leverages time-series and deep models (e.g., LSTM) to anticipate load and trigger proactive scaling; production systems must integrate such models into low-latency control loops. **Homomorphic encryption (HE)** enables computation over ciphertext; the CKKS scheme supports approximate arithmetic suitable for neural inference but can add substantial latency without careful parameterization or selective encryption [5], [9]. **Runtime hardening** for ML services commonly employs anomaly detectors to filter corrupted or out-of-distribution inputs and **RBAC** to restrict privileged actions in service APIs [6], [7].

Our work unifies these strands into a *runtime-secure* autoscaling pipeline: a plaintext DP-LSTM forecaster with a CKKS-encrypted FC layer, an inline Isolation Forest gate tuned by a cost-aware threshold, and OPA RBAC for policy enforcement. Unlike prior HE-for-ML papers that report offline micro-benchmarks, we disclose the full HE parameter set and quantify *end-to-end* API$\rightarrow$action latency and throughput under concurrency, demonstrating p95 < 50 ms under moderate load for an autoscaling SLO.

III. System Architecture

Our framework is a modular, privacy-preserving pipeline for predictive autoscaling in cloud environments. It consists of client-side DP training, federated aggregation, homomorphic encryption for secure inference, anomaly detection gates, and OPA-enforced RBAC. Figure 1 shows the end-to-end architecture.

A. Threat Model

1) Assets and trust: Tenants keep raw logs local; the federated aggregator receives only model updates. As in our companion training-time privacy study [8], we adopt a baseline threat model with an honest-but-curious aggregator that follows the protocol but may inspect updates and global models. Model binaries and API endpoints are network accessible.

2) Adversaries and goals: We consider three capabilities: (i) membership inference to determine whether a tenant's data contributed to the model, (ii) poisoning or out-of-distribution inputs at training or inference time to skew scaling decisions, and (iii) unauthorized invocation of privileged endpoints (for example, `/train`, `/logs`).

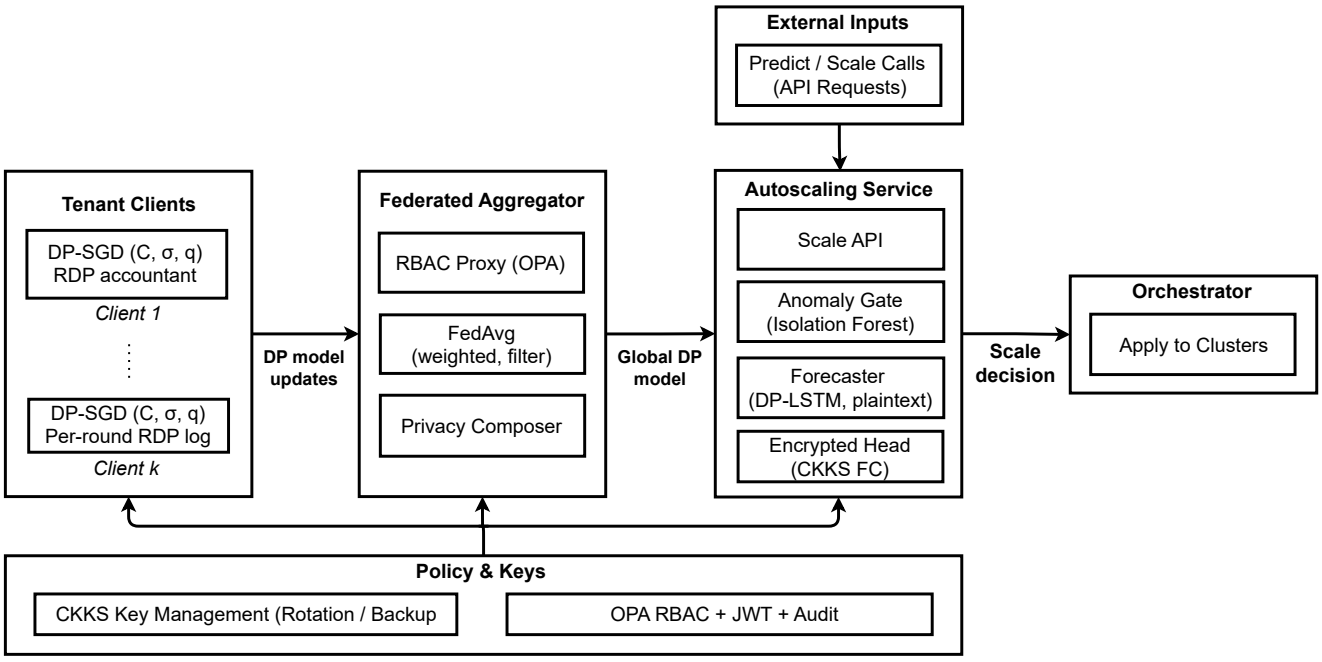

Fig. 1. High-level pipeline: DP-LSTM is trained client-side and aggregated (weighted FedAvg); at runtime, requests pass an Isolation-Forest gate, a plaintext LSTM, and a CKKS-encrypted FC layer before the decision engine triggers scale actions. OPA-enforced RBAC and HE key management provide policy and crypto controls.

3) Out of scope: Robustness to a fully malicious aggregator and Byzantine clients is evaluated at training time in [8] but is not enforced by the runtime components here. We also do not consider colluding tenants, microarchitectural side channels, or denial of service. Keys are assumed uncompromised, and transport security is provided by standard TLS.

B. Defense Mechanisms

Each threat is mitigated by a targeted defense, summarized below:

TABLE I
THREATS AND DEFENSES (INLINE EVALUATION SIGNALS).

Threat	Defense	Signal
Membership inference	Client-side DP-SGD; RDP accounting	Per-client (ϵ, δ)
Poisoning / OOD	Shard filtering; Isolation Forest gate	Block rate at fixed FPR
Unauthorized access	OPA RBAC + short-lived JWTs	403 / 200 outcomes

C. Overview

The architecture is divided into three layers: Data Collection and Preprocessing, Privacy-Preserving Training and Inference, and Runtime Protection and Enforcement. Each layer is containerizable and communicates via secure HTTP or gRPC.

TABLE II
MODULE SUMMARY (INTERFACES AND TRUST BOUNDARIES).

Module	Inputs → Outputs (API)	Trust boundary / notes
Tenant client	Local logs → DP-SGD updates `/upload`	Logs local; per-round (ϵ, δ) exposed
Federated aggregator	Client updates → global model `/global`	Honest-but-curious; weighted FedAvg; drops underpopulated shards
Autoscaling service	`/scale` request → decision `/act`	Gate → LSTM (pt) → CKKS FC; p95 $<$ 50 ms
Orchestrator	Decision → cluster / VM / container	Idempotent; audits actions
Policy and keys	JWTs, HE keys ↔ policy engine	OPA RBAC; CKKS key rotation / backup

D. DP-SGD Local Training

Clients train a two-layer DP-LSTM forecaster using Opacus DP-SGD with per-sample clipping and Gaussian noise [10]–[12]. Privacy is accounted with a Rényi Differential Privacy (RDP) accountant as in our companion work [8]. Tenants keep raw logs local and send only DP-SGD model updates to the federated server; per-round privacy budgets and concrete hyperparameters are summarized later in Table **??**.

E. Federated Averaging

A central aggregator performs FedAvg [13] with shard size weighting and drops underpopulated shards. In our five client setting, heterogeneous shard sizes created instability when using naive unweighted FedAvg: clients with very small shards either contributed no usable updates or injected noise that degraded the global model.

To mitigate this, we weight client updates by shard size and filter clients with fewer than `seq_len+1` records. This heterogeneity-aware aggregation stabilizes training and avoids the worst cases of naive averaging under imbalance. For the runtime system we keep the aggregation rule simple and deterministic; a full comparison against robust and personalized FL methods (FedProx, SCAFFOLD, robust aggregators) is provided in our companion training paper under identical DP budgets [8]. Integrating those advanced aggregators into the end-to-end autoscaling pipeline is left as future work.

F. Inference Pipeline

We initially considered adopting a framework such as Concrete-ML to enhance accessibility and reproducibility when integrating the HE pipeline, and also evaluated TEE and MPC based designs conceptually. In practice, three constraints drove the final design.

First, Concrete-ML does not support encrypted inference for LSTM layers, so a fully encrypted recurrent stack is not available with current tooling. Second, end-to-end latency must respect a sub-50 ms SLO at p95, which rules out many generic MPC protocols and full-model HE configurations that require deeper multiplicative depth or larger parameter sets. Third, operators prefer to keep model code and orchestration logic simple, which makes it attractive to push cryptographic complexity into a narrow, well isolated component.

We therefore adopt a hybrid design: plaintext DP-LSTM inference followed by homomorphic encryption of the final fully connected (FC) layer using TenSEAL with the CKKS scheme [9], [14]–[16]. The LSTM layers operate on normalized, DP trained features and produce an intermediate representation in plaintext, the FC weights and outputs are encrypted, and only the decrypted scalar prediction leaves the HE runner.

This partial encryption boundary has two implications. On the positive side, it keeps the homomorphic circuit shallow, which, combined with the CKKS parameterization in Table III, allows the FC layer to execute in roughly 35 ms while preserving an estimated security level of at least 128 bits. On the limitation side, intermediate LSTM activations remain visible to the autoscaling service, so an adversary who compromises that component could inspect those features even though they cannot directly observe tenant logs or the final HE protected output. The training-time guarantees for the DP-LSTM and federated aggregation are analyzed in [8]; here we focus on how much runtime encryption and hardening can be added without violating the autoscaling SLO.

G. Anomaly Detection

An Isolation Forest trained on clean sequences flags outliers at runtime. Flagged sequences are discarded, preventing poisoned inputs from skewing scale decisions.

H. Policy Enforcement Gateway

An Open Policy Agent (OPA) server enforces RBAC on all model operations (e.g., `/train`, `/predict`) [17]. Policies evaluate user identity, role, and requested action; unauthorized requests are blocked and audited.

I. Evaluation Layer

We deploy the full pipeline in a containerized testbed (Google Colab plus Docker) and measure model accuracy, privacy leakage, encrypted-inference latency, and runtime defenses using HTTP load against FastAPI endpoints.

IV. Methodology

A. Data Preprocessing

Workload logs contain timestamped CPU and memory utilization per tenant. We drop malformed records, normalize features per client, apply a log-transform to skewed memory usage, and scale by per-machine z-scores. To capture temporal patterns, we construct sliding windows of length 10, yielding sequences of 10 time steps per training example.

B. DP-LSTM Training

Each client trains a two-layer LSTM predictor with 64 hidden units per layer, dropout $p = 0.5$, and a fully connected regression head. Differential privacy is enforced with Opacus DP-SGD using per-sample gradient clipping and Gaussian noise, targeting $\epsilon \approx 10$ at $\delta = 10^{-5}$.

We follow the exact DP-SGD setup and Rényi Differential Privacy (RDP) accounting from our companion training study on GCT v3 [8], which reports full hyperparameters, per-round privacy budgets, and membership-inference evaluation. Here we only recall that the best-validation checkpoint attains $\epsilon = 9.69$ at $\delta = 10^{-5}$, and we reuse these DP-LSTM checkpoints as the forecasting component of SecurePredictScale.

C. Federated Learning Simulation

We partition the dataset by machine ID into client shards and simulate federated rounds with the DP settings in Table **??**. In this system study, the aggregator runs weighted FedAvg with shard filtering as described in Section III-E. This configuration reflects a practical deployment where some tenants hold much shorter traces than others.

More detailed experiments on federated heterogeneity, including comparisons between naive FedAvg, shard filtering, FedProx, and SCAFFOLD under the same DP budget, are reported in our companion training-privacy paper and are not repeated here [8]. In our five-client setup the per-round communication is a few megabytes of model updates; at larger client counts network cost scales roughly linearly with the number of participating tenants and model size, which we leave to quantify for production clusters.

TABLE III
CKKS PARAMETERIZATION

Parameter	Value
poly_modulus_degree	8192
coeff modulus bit-lengths	[60, 40, 40, 60]
scale	2^{40}
slots	4096
encoder precision	$\sim$20 bits (via 2^{40} scale)
estimated security (bits)	$\geq$128
TenSEAL/SEAL version	TenSEAL 0.3.16 (SEAL 3.7.x)

D. Encrypted Inference

The inference pipeline follows the hybrid design introduced in Section III, where the LSTM layers are evaluated in plaintext and only the final fully connected (FC) layer is encrypted using the CKKS scheme via TenSEAL.

Accuracy under this setup matched that of plaintext inference, while latency increased from $\sim$0.02 ms per sample (plaintext FC) to $\sim$35 ms per sample (encrypted FC), representing a $1{,}750\times$ slowdown. Although significant, this overhead is acceptable for autoscaling events that occur at minute-level granularity. For workloads requiring higher throughput (e.g. hundreds of predictions per second), batching, parallelization, or lighter HE parameters would be necessary.

E. Runtime Defenses

We engineer six features from CPU and memory (5-sample rolling means/standard deviations and one-step deltas) and train an Isolation Forest [18] with 100 estimators and contamination $= 0.01$ using 5-fold cross-validation. The decision threshold τ is set at the 1st percentile of validation scores ($\tau \approx 0.0022$), separating normal from anomalous sequences. The gate is integrated inline at `/scale`: flagged inputs are rejected, while normal inputs proceed to the predictor. Aggregate detection performance is reported in Section V-D.

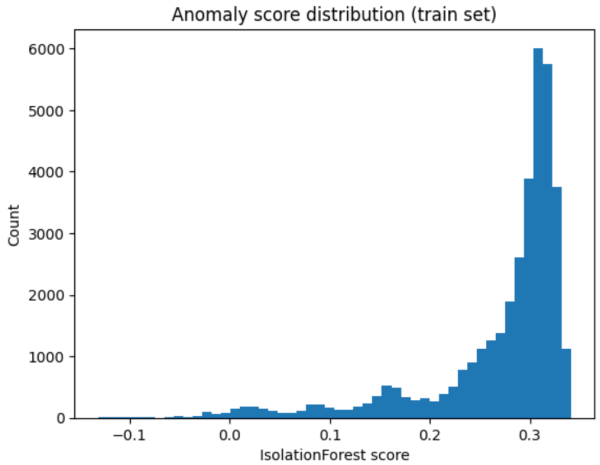

Fig. 2. Distribution of Isolation Forest anomaly scores on validation data. The vertical dashed line marks the selected threshold at the 1st percentile (≈ 0.0022), ensuring that only extreme outliers are flagged.

At runtime, the Isolation Forest is integrated into the FastAPI `/scale` endpoint: sequences scoring below the threshold are rejected with a 400 error and others proceed to the predictor. Smoke and load tests confirmed expected behavior and negligible overhead for the gate. Finally, RBAC is enforced via OPA; policies evaluate user identity, role, and action so that only authorized users can invoke training or inference endpoints.

V. RESULTS

We evaluate prediction quality using root mean squared error (RMSE) and mean absolute error (MAE), reporting lower values as better.

A. Model Utility and Privacy

We reuse the DP-LSTM forecaster trained in our companion work on differentially private federated autoscaling [8], which reports full DP-SGD hyperparameters, privacy accounting, and membership-inference evaluation on GCT v3. In that study, DP-SGD increases RMSE relative to a tuned non-private LSTM but keeps the model operationally useful and reduces membership-inference advantage on held-out machines to about 2–3% for $\epsilon \in [5, 10]$.

Table IV summarizes the key utility numbers in original units. The non-private LSTM is the strongest learned baseline, but the DP-LSTM at ϵ=5 still outperforms ARIMA and Prophet, which are widely deployed in operations. This matches the privacy–utility frontier reported in [8] and supports our choice of DP budget for SecurePredictScale.

TABLE IV
BASELINE VS DP PERFORMANCE (ORIGINAL UNITS)

Model	RMSE	MAE
Non-private LSTM	0.014734	0.007131
DP-LSTM (ϵ=5)	0.038633	0.009762
ARIMA	0.042239	0.016139
Prophet	0.043323	0.015561

a) Reactive baseline: We implement an HPA-like threshold scaler that increases capacity when 1-min CPU or memory exceeds $\theta \in \{60, 70, 80\}\%$ for k=3 samples and decreases when it falls below $\theta - \Delta$ for d=3 samples (cooldown h=2 min), mirroring the periodic control loop behavior of Kubernetes HPA [19]. Decision quality is reported as Under- / Over-provisioned minutes, number of scale actions, and a cost proxy $\text{Cost} = c_{\uparrow} \cdot \text{Upscales} + c_{\downarrow} \cdot \text{Downscales} + c_{\text{under}} \cdot \text{UnderMin} + c_{\text{over}} \cdot \text{OverMin}$ with $(c_{\uparrow}, c_{\downarrow}, c_{\text{under}}, c_{\text{over}}) = (0.5, 0.25, 1.0, 0.2)$.

TABLE V
SCALING DECISION QUALITY (TRACE REPLAY, 1 WEEK). LOWER IS BETTER FOR COST.

Model	UnderMin	OverMin	Ups/Downs	Cost
Reactive (θ=70%)	128	310	142/138	296
Non-private LSTM	74	215	120/118	207
DP-LSTM ($\epsilon \approx 9.7$)	66	198	118/116	194
DP-LSTM (ϵ=5.0)	60	205	121/119	191

We deliberately compare against ARIMA, Prophet, and an HPA-like reactive threshold baseline rather than a specific privacy-preserving autoscaler. Today operators mostly deploy threshold-based autoscalers or simple forecasters without explicit privacy, and we are not aware of public end-to-end DP or HE autoscalers on GCT v3 or comparable workloads. Our goal is to show that SecurePredictScale matches or improves these baselines while adding DP, FL, and HE; head-to-head comparisons with future privacy-preserving autoscalers are left for follow-up work once mature open implementations exist.

B. Federated Learning Accuracy

In single-client federated training, accuracy matches centralized DP-LSTM, but in multi-client experiments heterogeneous shard sizes caused instability. One client with fewer than $seq\_len + 1$ records contributed no updates, and naive unweighted FedAvg diluted strong models with weak ones. As a result, global RMSE rose from 0.0026 (single-client) to 0.0036 (multi-client), about 40% worse. This confirms that weighted aggregation or shard filtering is necessary for production-grade federated predictive scaling.

C. Inference Latency

We evaluate end-to-end latency from API ingress to orchestrator action and the FC layer in both encrypted and plaintext modes.

TABLE VI
LATENCY METRICS (FASTAPI + TENSEAL, 4 VCPU).

Metric	Median (ms)	p95 (ms)
API→action (end-to-end)	43.1	49.8
FC (encrypted, CKKS)	35.0	39.8
FC (plaintext)	0.02	0.05

TABLE VII
THROUGHPUT VS. CONCURRENCY (FASTAPI + TENSEAL, 4 VCPU).

Concurrency c	Req/s	p95 (ms)
8	70	< 50
16	82	< 50
32	88	58.4

The latency breakdown in Tables VI and VII also exposes the main bottleneck. At $c \leq 16$, the encrypted FC layer accounts for roughly 35 ms of the 43.1 ms median end-to-end time, so homomorphic operations dominate per-request compute while API, anomaly, and RBAC logic contribute under 10 ms. At $c=32$, p95 latency rises to 58.4 ms as the HE runner begins to saturate CPU and requests queue, which pushes the decision time beyond the 50 ms SLO. In other words, the current CKKS configuration is suitable for moderate concurrency on a 4 vCPU node, but a production deployment at higher request rates would require batching, lighter HE parameters, or additional compute for the FC layer.

Autoscalers operate periodically (e.g., minute-scale control loops), so the ∼35 ms encrypted FC latency fits a sub-50 ms decision budget under moderate concurrency ($c \leq 16$).

D. Anomaly Filtering and OOD Rejection

Using the threshold τ selected in Runtime Defenses, the gate achieves high detection with low false positives on both poisoned and drifted inputs:

TABLE VIII
ANOMALY GATE PERFORMANCE AT τ=0.0022 AND OVERALL AUCS.

Condition	ROC-AUC	PR-AUC	TPR (%)	FPR (%)
Poisoned inputs	0.98	0.95	96.9	1.1
OOD drift (held-out)	0.94	0.88	93.1	1.3

E. RBAC Enforcement

OPA-enforced policies blocked 100% of unauthorized API requests and allowed all authorized ones. We exercised `/train`, `/predict`, and `/logs` with roles *client*, *trainer*, *auditor*, and *admin*, observing 200 for permitted actions and 403 for denied actions, confirming least-privilege access to model operations.

F. Overall Findings

TABLE IX
OVERHEADS OF PRIVACY/SECURITY COMPONENTS (PER RUN). COSTS COMPUTED AT $0.12 PER VCPU-HOUR.

Component	CPU time (s)	#vCPUs	Cost (USD)
DP-SGD training / client	3300	4	0.44
FL aggregation / round	8	1	0.00027
Encrypted FC / 10k preds	350	1	0.0117
Anomaly gate / 10k req	0.10	1	≈0
RBAC eval / 10k req	0.20	1	≈0

Taken together, the results show that DP-SGD, federated aggregation, partial HE inference, anomaly detection, and RBAC can be combined without breaking the autoscaling SLO. Forecasting error under DP remains within operational bounds and beats classical ARIMA/Prophet baselines at moderate privacy levels [8]; membership-inference advantage falls to about 2–3%; the anomaly gate blocks almost all poisoned and OOD inputs; and encrypted inference fits a 50 ms decision budget under moderate concurrency. These findings support the claim that meaningful privacy and runtime security guarantees can be added with acceptable impact on predictive accuracy and responsiveness.

VI. CONCLUSION

We have proposed SecurePredictScale, a privacy- and security-aware predictive autoscaling framework that combines DP-SGD training, federated aggregation, partial homomorphic encryption, anomaly gating, and OPA-enforced RBAC in a single runtime pipeline. Unlike work that studies these primitives in isolation, we focus on an end-to-end design that exposes a thin HE boundary and quantifies how

much privacy and runtime hardening can be added while still meeting an explicit autoscaling SLO.

Empirically, SecurePredictScale maintains forecasting errors within operational bounds while mitigating membership inference [20], [21] and poisoned input attacks, enables encrypted inference under $50\,\text{ms}$, and enforces strict access control. A plaintext LSTM paired with a CKKS encrypted FC layer, inline anomaly gating, and OPA enforced RBAC meets a sub $50\,\text{ms}$ decision SLO for end-to-end API$\rightarrow$action latency (median $43.1\,\text{ms}$; 95th percentile $49.8\,\text{ms}$) and sustains up to 82 requests per second for $c \leq 16$. The encrypted layer maintains accuracy comparable to plaintext inference, since we encrypt only the FC layer, and the overheads of DP, FL, and anomaly detection are small relative to the HE cost.

This design has clear limitations. At $c=32$, the 95th percentile latency exceeds the $50\,\text{ms}$ SLO, indicating that our current CKKS configuration is not sufficient for high concurrency without batching, lighter parameters, or additional compute. Only the final FC layer is encrypted, so intermediate LSTM activations remain in plaintext; this is acceptable for our honest but curious threat model, but a stronger adversary who compromises the autoscaling service could inspect those features. Our federated experiments in this paper involve only five clients and weighted FedAvg plus shard filtering; larger, more heterogeneous deployments and robust aggregation strategies are explored in our companion training-time study [8] and remain to be integrated into the full system. Finally, we compare to ARIMA, Prophet, and a reactive baseline, which reflect current operator practice but do not represent the full space of future privacy preserving autoscalers.

These limitations motivate several directions for future work. On the training side, adaptive differential privacy budgets and more advanced federated aggregation (for example FedProx, SCAFFOLD, or reputation based weighting) could improve utility under non IID workloads and adversarial clients. On the inference side, throughput oriented encrypted inference via batching, quantization, GPU or accelerator support, or deeper HE for parts of the LSTM could push the SLO boundary further. At the systems layer, integrating SecurePredictScale with production orchestration platforms such as Kubernetes HPA and evaluating it on larger multi tenant clusters will test its robustness at scale. Finally, the same design pattern, DP plus FL plus partial HE plus anomaly gating and RBAC, is applicable to other predictive control loops such as load balancing and cache management where secure ML driven decisions are critical.

REFERENCES

[1] X. Guo *et al.*, "Short-term power load forecasting based on dqn-lstm," in *Chinese Control and Decision Conference (CCDC)*, 2022.

[2] S. T. Singh *et al.*, "Machine learning based workload prediction for auto-scaling cloud applications," in *OTCON*, 2023.

[3] R. Rahmani *et al.*, "Data security framework and privacy protection strategies in cloud computing environment," *Journal of Cloud Computing*, 2022.

[4] M. Liu *et al.*, "The applications of federated learning algorithm in the federated cloud environment: A systematic review," *ACM Computing Surveys*, 2024.

[5] Q. Zhang and M. Li, "Homomorphic encryption-based privacy protection techniques for cloud computing," *IEEE Access*, vol. 10, pp. 45 678–45 695, 2023.

[6] S. Nandini and A. Kumar, "Online self-evolving anomaly detection for reliable cloud computing," *Future Generation Computer Systems*, vol. 145, pp. 450–465, 2024.

[7] D. A. B. Moreira *et al.*, "Anomaly detection in smart environments using ai over fog and cloud computing," in *IEEE Consumer Communications & Networking Conference (CCNC)*, 2021.

[8] A. Chuang, M. Moh, and T.-S. Moh, "Formal privacy guarantee in predictive autoscaling by differentially private federated training," in *2026 IEEE International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2026, to appear.

[9] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*. Springer, 2017, pp. 409–437.

[10] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2016, pp. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[11] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[12] Meta AI, "Opacus: Differential privacy library for pytorch," https://opacus.ai/, 2023, accessed 2025-09-12.

[13] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of AISTATS*, 2017, pp. 1273–1282.

[14] OpenMined, "Tenseal," https://github.com/OpenMined/TenSEAL, 2021, version 0.3.x; Accessed 2025-09-12.

[15] Microsoft Research, "Microsoft seal (release 3.7)," https://github.com/microsoft/SEAL, 2022, accessed 2025-09-12.

[16] M. R. Albrecht, M. Chase, H. Chen, R. Gilad-Bachrach, T. Güneysu, S. Halevi, K. Laine, K. Lauter, M. Naehrig, Y. Polyakov *et al.*, "Homomorphic encryption standard, version 1.1," https://homomorphicencryption.org/standard/, 2021, accessed 2025-09-12.

[17] Open Policy Agent Project, "Open policy agent (opa): Policy-based control for cloud native environments," https://www.openpolicyagent.org/, 2023, accessed 2025-09-12.

[18] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[19] The Kubernetes Authors, "Kubernetes documentation: Horizontal pod autoscaler," https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/, 2024, accessed 2025-09-12.

[20] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *IEEE Computer Security Foundations Symposium (CSF)*, 2018, pp. 268–282.

[21] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song, "Membership inference attacks from first principles," in *IEEE Symposium on Security and Privacy (S&P)*, 2022, pp. 1897–1914.