

Defending the Pipeline: Engineering Secure, Real-Time AI Fraud Detection Systems

Abhishek Bajpai

Senior Software Developer Finance and Supply Chain

ARTICLE INFO

Received: 01 May 2026

Revised: 17 June 2026

Accepted: 28 June 2026

ABSTRACT

Real-time fraud detection services are placed in the critical path of card authorisation and need to return a decision approve-or-decline in under one hundred milliseconds. Running a machine learning model in this critical path increases detection quality, but also attack surface. In addition to the model itself, the scoring interface, feature store and label feedback loop are new attack targets in an ML-powered fraud system. Previous work has considered detection accuracy and pipeline security in isolation, and the cost of security controls within the authorisation window is poorly understood. Here, we present STREAM-Guard, a four-layer pipeline. These include ingestion-layer salted tokenisation, cryptographically verified provenance, differentially private and adversarially regularized training and behavioral rate limiting at the inference gateway, subject to a latency budget model that divides the authorization window between these controls. In addition, a provenance admission rule bounds the share of poisoned records that reach the offline feature store. Evaluated on 284,807 transactions made with European cards, and with multiple replays of synthetic streams with varying speeds (up to 5,000 transactions per second), STREAM-Guard achieved an AUC-PR of 0.851 (99th percentile latency: 42.1 ms), compared to 0.834 (29.0 ms) for a baseline gradient boosting approach with no security hardening. By securing the pipeline, 0.024 AUC-PR was lost under 10 percent label-poisoning abuse, as opposed to 0.212 when the pipeline was not secured. The advantage of membership inference dropped by 0.058. In practice, with homomorphic inference taking 1.42 s per decision, this task is infeasible at that rate.

Keywords: real-time fraud detection; adversarial machine learning; data poisoning; data provenance; differential privacy; tokenisation; API security; financial machine learning pipelines

1. INTRODUCTION

Payment fraud is one of the costliest forms of financial fraud and remains impossible to scan manually, particularly now that card not present transactions are so common online. Issuers used deterministic rules engines to detect fraud, encoding expert knowledge as thresholds on amount, geography and merchant category. Such engines are fast and interpretable, but their generalization performance is weak as the fraudster can simply change their behavior until the rule no longer fires. Supervised classifiers are preferred, with tree ensembles in particular repeatedly outperforming static rule sets on card transaction data (Randhawa et al., 2018). Real production conditions nevertheless diverge from these benchmark conditions, as customers may change their habits, fraudsters may develop new strategies, and only a small fraction of transactions are ultimately confirmed by an investigator (Dal Pozzolo et al., 2018).

The second constraint imposed by the payment network is that the score should be computed whilst the authorisation message is in flight. This gives a window of between 50 and 100 milliseconds for the round trip over the network, feature construction and inference to be carried out. This requires streaming (rather than batch scoring) to be used. Distributed log frameworks have been shown to be able to guarantee such throughput without loss of detection quality (Carcillo et al., 2018). Since the deadline is dictated by the payment scheme, any additional processing in the pipeline thus has to justify its existence in latency and risk terms.

Once the model is installed in the authorisation path, an opponent can try to learn how to bypass it. If the opponent is able to access the querying process it can perform an evasion attack: submitting transactions that are slightly different from the previous ones, to determine the boundary, and then a transaction that evades the boundary just inside it. Attacks have been demonstrated against deployed banking detectors with realistic attacker-controlled parameters (Carminati et al., 2020). Pipelines that retrain on investigator-labeled transactions have a second side-channel. The user who influences those labels influences subsequent training, and even the model itself leaks information. Membership-inference and reconstruction attacks can recover training record properties directly from the exposed interface (Rigaki & Garcia, 2024), and systematic reviews indicate that they have become a mature threat class (Malik et al., 2024).

Indeed, there exists a defense against each of these threats, however, for the most part they are studied in isolation, and they do not take into account latency constraints. While fully homomorphic encryption has no plaintext exposure window, the cost is orders of magnitude higher than normal arithmetic (Marcolla et al., 2022). Federated learning has the benefit of removing movement of raw transactions entirely. It has been leveraged for collaborative fraud modeling (Abdul Salam et al., 2024; Tang & Liang, 2024), but this is a problem of sharing between institutions, as opposed to within an institution, where the data and model is co-located. Differential privacy and adversarial training have also not been studied in the context of the ingestion, provenance and gateway control that real-world production pipelines require.

We find a gap between what security scholarship calls for and what an engineering team can deploy with 100 ms. STREAM-Guard closes that gap using a four-plane architecture that deploys controls at minimal cost and maximizes their value. It quantifies, rather than assumes, residual latency and accuracy costs. The contributions have been:

- A latency budget model that quantifies the time taken by each of the network, tokenisation, feature retrieval, inference and gateway stages, rendering the cost of each security control explicit and comparable.
- Provenance-verified feedback loop, delivering each candidate training record along with a hash-chained, signed lineage and a closed-form bound on the effective poisoning rate that would survive the admission control.
- A cost-aware adversarial training scheme for tabular transaction data, where perturbations are limited by properties of features controllable to the opponent. A behavioral rate-limiting scheme based on decline ratio, entropy and velocity instead of solely source address is also proposed.
- An evaluation on a common testbed of their joint effect on detection quality, tail latency, robustness to evasion, resistance to poisoning, and training-data privacy, against enclave-based and homomorphic alternatives.

2. RELATED WORK

2.1 Learning models for real-time transaction fraud detection

The structure of card fraud detection classifiers is well studied in the literature, focusing particularly on the extreme class imbalance. Gradient boosted trees and ensembles of classifiers are strong and general choice of model on tabular transaction features. Boosted learners combined by majority voting with noise injection also perform well on this task (Randhawa et al., 2018). Since fraud is less than 1% of traffic, area under the precision-recall curve and precision at a fixed alert budget are more commonly used than accuracy to measure reporting performance. Dal Pozzolo et al. (2018) formalize the operating conditions of a real system, as the existence of both drift and class imbalance and the presence of verification latency, and show that separating recently validated feedback from delayed investigator labels is better than using one static classifier.

However, supervised models only detect fraud similar to the labeled examples. Carcillo et al. (2021) solved the problem by computing unsupervised outlier scores at the global, local and cluster levels and using them as features in the supervised model, improving AUC-PR compared to both the unsupervised and supervised families. Forough and Montazi (2021) pursue the same aim using deep sequential encoders with voting, building on a cardholder's

transaction history. Graph Neural Networks, that can share signals across devices, merchants and accounts, have reported improvements over flat tabular baselines (e.g. Motie & Raahemi, 2024; Cheng et al., 2025). Deployment has been considered separately, e.g. Carcillo et al. (2018) built a scalable streaming framework on a distributed log and cluster compute layer, showing the throughput feasibility, but left the pipeline as trusted infrastructure.

Deployment, as a separate concern, has also been researched. Carcillo et al. (2018) built a scalable streaming architecture on an event log and a distributed cluster of compute nodes to prove near real-time scoring at production volume. This resulted in established throughputs but treated the pipeline as if it were a trusted infrastructure, rather than being an attack vector on the transaction.

2.2 Adversarial and privacy exposure of deployed fraud models

However, the assumption of a trusted pipeline falls apart when an opponent is active. Carminati et al. (2020) constructed evasion attacks against real-world banking fraud detection systems, sampling only attributes that adversaries can reasonably control, showing the possibility of modifying a fraudulent transfer without sacrificing the opponent's payoff in a way that evades a deployed detector. Perturbations in the tabular financial setting are therefore constrained rather than arbitrary, which is distinct from the unconstrained image setting most adversarial machine learning theory was developed under.

More general surveys covering well-known evasion and poisoning attacks, extraction and inversion attacks, and the corresponding defenses for each. Based on their survey, Malik et al. (2024) showed that most of the published defenses are not generalizable and are evaluated on only one attack class. Evidence supporting a connection derives from He et al. (2023) that in intrusion detection settings with protocol semantics-limited feature spaces, defenses transfer poorly, and from Rigaki and Garcia (2024) that the more queries an interface allows the more exposure grows, exactly what is achieved with an always-on scoring API (Ozkan-Okay et al. 2024).

2.3 Protective computation in financial machine learning pipelines

Cryptographic and distributed solutions partially reduce this: full homomorphic encryption supports arithmetic on ciphertexts, so a scoring service could apply the model without plaintext features. However, Marcolla et al. (2022) found that, while hardware acceleration and approximate schemes have reduced this overhead, it remains several orders of magnitude higher than for plaintext arithmetic. This overhead is acceptable for batch analytics, but not for a synchronous authorisation decision.

In federated learning, only model updates are shared and the dataset remains decentralized. The federated training of card fraud detection was studied by Abdul Salam et al. (2024) with two frameworks and resampling and by Reddy et al. (2024), who found that a federated deep model was equally effective. Tang and Liang (2024) extend to graphs of transaction data to learn relational patterns between institutions without pooling. Tang and Liu (2024) apply the transformer to federated protocols. However, they reduce what leaves the institution, leaving the intra-institutional path untouched: the operator still ingests plaintext identifiers, still runs an exposed scoring endpoint, still retrains on their feedback stream.

2.4 Research gap and positioning of the present work

Three observations follow. Detection literature, while optimizing for drift and imbalance, assumes the underlying infrastructure is trustworthy. The adversarial and privacy literature characterizes attacks fully, validating prototypes one-at-a-time, and rarely evaluates the timings of protective computation (above). The protective computation literature provides strong guarantees, but does not state the inefficient costs they incur, nor reconcile them with a hard authorisation deadline. To our knowledge, no work budgets a consistent millisecond budget across coordinated controls (ingestion, provenance, training and the inference gateway) and reports the quality, robustness, privacy leakage and tail latency for the same configuration. Informed by this gap, STREAM-Guard is not a new classifier but a placement discipline, placing each control where it costs least and protects most. Instead of proposing new classifiers, STREAM-Guard proposes a placement discipline which fills a gap in the literature by pushing the

tokenisation down to the ingestion boundary. We also limit the plaintext exposure window by committing to tokens before running model code, and ensure integrity of the feedback stream instead of the model: it is cheaper to reject a poisoned label than unlearn it. Privacy is purchased in the offline training plane, at a cost of hours rather than milliseconds. The only control remaining in the synchronous path is a lightweight behavioral gate.

3. METHODOLOGY

3.1 Threat model and design objectives

The threat model assumes that the opponent is an economically motivated attacker with access to the public-facing scoring interface and transaction channel, but not the training infrastructure or the hardware security module that guards the tokenisation keys. In particular, the model assumes the opponent has the ability to (1) place unlimited transactions with observable outcomes to probe boundaries and extract model internals; (2) place transactions with outcomes later incorporated into the labeled feedback stream to poison labels; (3) control a large set of residential proxy addresses, to defeat address-based rate limiting; and (4) send invalid payloads to the model execution runtime. Five objectives respond. O1 states that there can be no plaintext account or device identifier downstream of the ingestion boundary. O2 requires all records in the offline feature store to have verifiable lineage. O3 requires the decision boundary to be strong to attacks with the budget available to the opponent. O4 requires that the amount of information that can be extracted from any individual training record is bounded by a specified privacy parameter. O5 requires all of the above while the end-to-end 99th percentile latency remains below the authorisation service level agreement. Finally, Table 1 maps each protected asset to the opponent capability that threatens it, the control that prohibits the opponent from exercising that capability, and evidence that the control is in place. Reading the table across the planes shows directly the placement discipline: two offline controls, one at the ingestion boundary and one at the synchronous path, in Planes C and D, respectively.

Asset	Adversary capability	Attack	Control and plane	Evidence
Cardholder identifiers	Read access downstream of the gateway	Disclosure, re-identification	Salted keyed tokenisation (Plane A)	Fig. 2, Table 4
Scoring interface	Unlimited queries with observable outcomes	Boundary probing, evasion	Cost-aware adversarial training (Plane D)	Fig. 4
Scoring interface	Distributed low-rate querying via proxies	Card-testing enumeration	Behavioural rate limiting (Plane B)	Fig. 7(b)
Label feedback stream	Submission of transactions later adjudicated	Label poisoning	Hash-chained provenance and admission control (Plane C)	Fig. 5
Released model	Repeated queries against the deployed model	Membership inference, reconstruction	Differentially private training (Plane D)	Fig. 6
Model runtime	Transmission of arbitrary payloads	Malformed-payload injection	Schema and size validation (Plane A)	Analysis only

Table 1. Threat model. Each row links a protected asset to the adversary capability that threatens it, the control assigned to that threat, and the result that provides evidence for the control. The final row is argued rather than measured, since payload validation admits no meaningful accuracy or latency metric.

3.2 Architecture of the proposed framework

There are four planes, as illustrated in Figure 1, with differing trust and latency characteristics, with schema validation, salted tokenisation and real-time feature construction occurring on Plane A, at the demilitarized zone boundary. Plane B contains the synchronous scoring service, a single-tenant micro-service, holds its own memory space, and contains an in-memory map of the tokens. Plane C contains the scored results and investigator labels. Signed, append-only, hash-chained event log and a quarantine for expired investigator labels. Plane D trains the models offline in an air-gapped facility with differentially private optimisation and cost-aware adversarial regularization. Signed artifacts are published to a registry which Plane B pulls from.

This is valuable because the stream of sensitive information is broken: personally identifiable information terminates in Plane A and never crosses into Plane B. So, even if the scoring service is compromised, it will only leak tokens and not account numbers. Feedback between Plane B and Plane C and Plane D is downward and signed so a poisoned label will not reach the feature store unless the opponent can also forge the signature. Model artifacts are uploaded to Plane B through a signed registry to ensure attackers with write access to object storage cannot replace a weakened model.

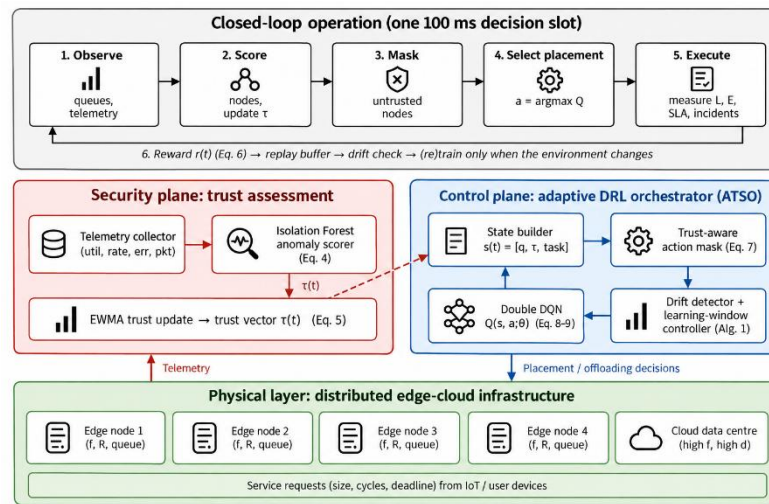


Figure 1. Architecture of the proposed STREAM-Guard framework. The four planes are separated by trust level and latency regime; personally identifiable information terminates in Plane A, feedback is admitted to Plane D only after provenance verification in Plane C, and model artefacts return to Plane B through a signed registry.

3.3 Latency budget formulation

Because the authorisation deadline is externally imposed, the framework begins from a budget rather than from a model. Let $L_{\#(SLA)}$ denote the maximum permissible end-to-end latency. The synchronous path decomposes into the network round trip $L_{\#(net)}$, ingestion and cryptographic transformation $L_{\#(tok)}$, feature retrieval $L_{\#(feat)}$, model inference $L_{\#(inf)}$ and gateway post-processing $L_{\#(gw)}$, giving the feasibility condition

$$L_{\#(e2e)} = L_{\#(net)} + L_{\#(tok)} + L_{\#(feat)} + L_{\#(inf)} + L_{\#(gw)} \leq L_{\#(SLA)} \quad (1)$$

The share of the budget consumed by controls that exist only for security is captured by the security overhead ratio

$$\Omega = (L_{\#(tok)} + L_{\#(gw)}) / L_{\#(e2e)} \quad (2)$$

Equation (2), above, makes this speculation into a number. The design rule used by this paper's architecture is $\Omega \leq 0.35$ at the 99th percentile. This is because this makes tokenisation a keyed hash function instead of asymmetric cryptography, and the gateway check a bounded arithmetic score instead of a database query.

3.4 Ingestion and tokenisation plane

All sensitive identifiers are masked at the gateway before writing the record to any queue, cache or log. The gateway first checks for a message's payload against a JSON schema, and rejects messages beyond predetermined size and nesting limits, closing off the malformed-payload route into the model runtime. Let each surviving identifier x be replaced by a truncated keyed digest, $\tau(x) = \text{Truncb}(\text{HMAC-SHA256}(ke, sx \parallel x))$, where ke is the epoch key and sx is a per-identifier salt, each stored in the hardware security module. Zeroise the plaintext identifier and the salt. Since velocity aggregates run over the token, the feature builder never sees the original identifier.

The two parameters are b , the number of bits to truncate to, which limits storage cost but increases collision probability to birthday level. However, collisions are still rare; with $b = 96$, the probability of collision is less than 10^{-13} with 10^8 distinct cards, smaller than the risk we are trying to eliminate. To avoid this, a token retrieved from a previous epoch cannot be correlated back to the same subject identifier in the current epoch. However, since epoch keys are rotated every twenty-four hours and the digest is a symmetric keyed hash rather than an asymmetric operation, the stage is still in the range specified by (2).

3.5 Provenance-verified feedback loop

While it would be convenient to automatically retrain from transactions flagged as suspicious, this is dangerous because the feedback stream could be manipulated. The framework therefore considers the feedback stream untrusted and requires a proof of provenance for all records. Each producer appends records to a partitioned log: record i adds $h_i = H(h_{i-1} \parallel m_i \parallel t_i)$ and signature σ_i of the hash over the producer's private key to the log; m_i is the record payload and t_i is its producer timestamp. A record is added to the offline feature store if its signature is valid for a registered producer key, its hash chains to the prior record, and its timestamp falls within an accepted window.

$$a_i = 1 \Leftrightarrow \text{Verify}(pk_p, \sigma_i, h_i) \wedge (h_i = H(h_{i-1} \parallel m_i \parallel t_i)) \wedge (|t_i - \hat{t}| \leq \Delta) \quad (3)$$

Records failing any condition are quarantined rather than discarded, partly because they may provide useful investigatory information. An integrity alert fires if the quarantine rate exceeds a threshold θ_q . Closed-form solutions can also be derived for this rule. If the attacker seeks to insert a proportion ρ of poisoned records, and the verifier rejects a record that is unattributable with probability γ , then the proportion of records accepted to the training set is

$$\rho_{\#(\text{eff})} = (1 - \gamma) \rho / (1 - \gamma \rho) \quad (4)$$

An upper bound on the linear poisoning threat is Equation (4) multiplied by the verifier's rejection rate. Verifiers with high cost benefit by maintaining a small, strong-valued $\rho_{\#(\text{eff})}$, even when ρ is large. Because the keys to sign blocks are kept in hardware and silent deletion is always detectable in the chain, γ 's security depends only on custody rather than statistical separability. In contrast, outlier-based label filtering depends on the ability to distinguish poisoned records from innocuous records, which are under the attacker's control, rather than the defender's.

3.6 Protected training plane

Training consists of two stages inside the hidden unit in the plane, which bounds the amount of information released by the obtained model regarding each individual training data. Also, per-example gradients are clipped to a fixed norm C , and perturbed with calibrated Gaussian noise with scale σ before each update, as in standard differentially-private stochastic gradient methods (adjusting the moments accountant over T steps). The operating point for all the tests is $\epsilon = 3.0$ with $\delta = 10^{-5}$, as justified in section 4.5.

The second stage tightens the decision boundary further under the constraints that a fraudster cannot change the acquirer's timestamp to the transaction, nor the merchant's registered merchant category. The fraudster still controls the transaction amount, inter-arrival interval and the device reuse pattern, and incurs a cost for every change made

to these parameters. This means the perturbation is made via a binary mutability mask M , and is priced by the cost function $c(\cdot)$ with budget $B_{\#(adv)}$,

$$\delta^* = \arg \max_{\delta} L(f_{\theta}(x + \delta), y) \text{ s.t. } M \odot \delta = \delta, c(\delta) \leq B_{\#(adv)}, x + \delta \in X_{\#(valid)} \quad (5)$$

Training then minimizes a convex combination of clean and adversarial risk, so that robustness is gained without discarding performance on ordinary traffic,

$$\min_{\theta} E_{(x,y)} [(1 - \lambda) L(f_{\theta}(x), y) + \lambda L(f_{\theta}(x + \delta^*), y)] \quad (6)$$

We set $\lambda = 0.35$, the value validated in Sec. 4.2. The key difference between our pipeline and standard adversarial training is that the attack is restricted to changing only the mutable fields. Otherwise, the unheeded attacker model will spend its budget modifying the immutable fields, creating a hardened model to a perturbation that no real attacker could effect. This dependency can also be understood via the perturbation search in equation (5) being performed on the same minibatch which has its gradients clipped and perturbed in a different sequence, causing the adversarial search to cost different amounts of privacy budget. Algorithm 1 resolves this problem.

Algorithm 1 Differentially private, cost-aware adversarial training

Input: admitted set A ; mask M ; cost $c(\cdot)$; budget $B_{\#(adv)}$; mix λ ; clip C ; noise σ ; steps T

Output: trained parameters θ with an (ϵ, δ) guarantee over A

```

1: Initialise  $\theta$ ; account  $(\epsilon, \delta)$  for  $T$  steps at sampling rate  $q$  and noise  $\sigma$ 
2: for  $t = 1$  to  $T$  do
3:   Sample lot  $B$  from  $A$  with sampling rate  $q$ 
4:   for each  $(x, y)$  in  $B$  do
5:      $\delta \leftarrow 0$ 
6:     for  $k = 1$  to  $K$  do      // ascent on mutable features only; no budget consumed
7:        $\delta \leftarrow \delta + \eta \cdot M \odot \text{sign}(\nabla_{\delta} L(f_{\theta}(x + \delta), y))$ 
8:        $\delta \leftarrow \text{Project}(\delta, c, B_{\#(adv)}); \text{Clamp}(x + \delta \in X_{\#(valid)})$ 
9:     end for
10:     $g \leftarrow \nabla_{\theta} [(1 - \lambda)L(f_{\theta}(x), y) + \lambda L(f_{\theta}(x + \delta), y)]$ 
11:     $g \leftarrow g / \max(1, \|g\|_2 / C)$       // clipping applied after fusion, once per example
12:  end for
13:   $\theta \leftarrow \theta - \alpha \cdot (\text{mean}(g) + N(0, \sigma^2 C^2 I) / |B|)$ 
14: end for
15: Sign  $\theta$  and publish to the model registry

```

Algorithm 1. Offline training loop. The perturbation search operates on the model rather than on the data, so it consumes no privacy budget; clipping is applied once per example after the clean and adversarial losses are fused, which keeps the sensitivity bound intact.

3.7 Hardened inference plane

The scoring service combines a gradient boosted classifier with a reconstruction-error signal from an autoencoder trained on genuine traffic, which lets the system react to fraud patterns absent from the labelled history. Writing $\sigma(\cdot)$ for the logistic transform and $r_{\#(AE)}(x)$ for reconstruction error, the fused score is

$$s(x) = w_1 \sigma(f_{\#(\text{gbdt})}(x)) + w_2 [1 - \exp(-r_{\#(\text{AE})}(x) / \beta)], \quad w_1 + w_2 = 1 \quad (7)$$

Alerts are raised by a minimum-risk rule rather than a fixed cut-off, since a false decline and a missed fraud carry very different business costs. The threshold is therefore set to $\tau^* = C_{\#(\text{FP})} / (C_{\#(\text{FP})} + C_{\#(\text{FN})})$, which at the cost ratio of 1 : 27 used here gives $\tau^* = 0.036$. The gateway then limits abusive querying without relying on source addresses, which distributed proxies render meaningless. For a session s the limiter combines the observed decline ratio $d(s)$, the normalised Shannon entropy $H(s)$ of the requested attribute values, and the request velocity $v(s)$,

$$R(s) = \alpha_1 d(s) + \alpha_2 [1 - H(s) / H_{\#(\text{max})}] + \alpha_3 v(s), \quad \text{throttle if } R(s) > \theta_r \quad (8)$$

The entropy term is what makes this rule effective against card testing. A legitimate session varies amount, merchant and device unpredictably, whereas an enumeration script holds most fields constant and iterates over one, producing a low-entropy signature that persists however many source addresses the attacker rotates through.

3.8 Security analysis

These guarantees come about because (i) tokenisation occurs before leaving Plane A and (ii) salts have no exit from the security module. An opponent with full access to the scoring service would retrieve, under this key custody assumption (O1), tokens that cannot be reversed without knowledge of the epoch key rather than account numbers. Condition O2 holds trivially for the rule of equation (3), since an unsigned, unpetitioned record was never written to the feature store. In general, we express the residual exposure logically in equation (4). The privacy condition (ϵ, δ) is the standard (ϵ, δ)-bound, and includes all sorts of adversaries, for example adversaries armed with auxiliary knowledge about the target cardholder, which is more than what O4 is only responsible for in terms of the empirical attack success rate.

There are three residual risks: if a producer loses their signing key, they can insert records that verify as valid. This makes equation (4) asymptotically equal to the unproven case above, scaled by the compromised producer's traffic ratio. This is the residual admission in Section 4.4, and its existence is what changes the binding control from an attribute of the learning algorithm to key rotation and custody. Adversarial training can only guarantee no degradation within its local perturbation budget. An attacker with larger operational cost could step outside the budget, as made explicit in Equation (5). Though the malformed-input route to failure is eliminated by construction with payload validation, this claim can only be stated, rather than measured, since there is no metric of accuracy or latency.

3.9 Dataset and experimental configuration

These two complementary datasets were used because there is no single public corpus containing both a detection benchmark and an infrastructure benchmark. The quality of the detection was measured on the European cardholder dataset from the publication of Dal Pozzolo et al. (2018) with 284807 transactions (of which 492 were fraudulent). Since its features are the principal components of the original fields, cardholder privacy is preserved while the raw identifiers needed for a tokenisation study is not preserved. The infrastructure behavior was measured on a synthetic stream constructed from the marginal and joint statistics of the public dataset and the identifier fields, device fingerprints and session structure a production feed carries. Sources are summarized in Table 2.

Attribute	Public benchmark corpus	Synthetic streaming corpus
Purpose	Detection quality and robustness	Latency, throughput and gateway abuse
Records	284,807	1,200,000
Fraudulent records	492 (0.172%)	2,184 (0.182%)
Input features	28 PCA components, time, amount	31 base fields plus identifiers and device data

Attribute	Public benchmark corpus	Synthetic streaming corpus
Engineered features	14 velocity and aggregate features	14 velocity and aggregate features
Observation period	Two consecutive days	Seven simulated days
Split protocol	Temporal 60 / 15 / 25 per cent	Replayed at 500–5,000 TPS
Adversarial subset	4,920 perturbed fraud vectors	48,000 card-testing probes
Poisoned subset	0–20% mislabelled feedback records	Not applicable

Table 2. Characteristics of the two corpora. The public benchmark supports comparison with prior detection results, while the synthetic stream supplies the identifier and session structure needed to measure ingestion and gateway behaviour.

All experiments in the paper were run on the same node with 16 virtual CPUs and 64 Gb of memory, without using a GPU, in order to enable a more commodity deployment. The streaming layer was a distributed log of three brokers at replication factor three. The token map used an in-memory key value store with a bounded time-to-live. Tokenisation used HMAC-SHA256 truncated to 96 bits with a 24 hour key epoch. Provenance used Ed25519 signatures, 120 s acceptance window and $\theta_q = 0.02$ alert threshold Private training used clipping norm $C = 1.0$, noise multiplier $\sigma = 1.1$ and lot size 512 to reach $\epsilon = 3.0$ at $\delta = 10^{-5}$; adversarial training used $K = 7$ ascent steps, $\lambda = 0.35$ and $B_{\#(adv)} = 0.10$ of the mutable range. The ensemble weights were $w_1 = 0.72$ and $w_2 = 0.28$ with $\beta = 0.45$, the cost ratio was 1 : 27 giving $\tau^* = 0.036$, the rate limiter used $\alpha = (0.45, 0.35, 0.20)$ with $\theta_r = 0.62$, and the base classifier used 600 trees of depth 7 at a learning rate of 0.05. These values were selected on the validation partition and held fixed across every experiment.

3.10 Evaluation procedure and metrics

The evaluation of the model must respect the time order since no future transaction of the same user can affect the decision of the previous transactions. Since fraud is below one per cent, the area under the precision-recall curve and precision at constant daily checked cases (often called alert budget, k) are the main metrics used for evaluation. Robustness was measured using the test set after performing the constrained attack (see equation (5)) at successively higher budgets. Poisoning resistance was measured by injecting mislabelled records into the feedback stream at 0% to 20% mislabelled rates and then retraining. Privacy leakage was measured as membership-inference advantage, defined as the overall advantage that an opponent had over chance in guessing correct labels on a balanced set of members and non-members.

The two other metrics that also apply to deployment are distribution drift between the training window and the live window (population stability index) and latency at the 50th, 95th and 99th percentiles during a steady load since the authorisation deadline is a tail requirement. All values are the average of five separate runs with different random seeds. The maximum observed deviation across runs was 0.006 in AUC-PR and 1.8 ms (99th percentile).

4. RESULTS

4.1 Latency composition and architectural feasibility

The first step is to check whether these control mechanisms fit into the authorisation window. The 99th percentile latencies for four configurations are shown in Figure 2 on a log scale. The unsecured baseline is 29.0 ms, while network round trip adds 18.0 ms. The salted tokenisation, token-map lookup and behavioral gate delay together add 42.1 ms. The headroom from the underlying 100 ms deadline is 57.9 ms. These give a security overhead ratio of 0.22 according to equation (2), well within the 0.35 design rule line.

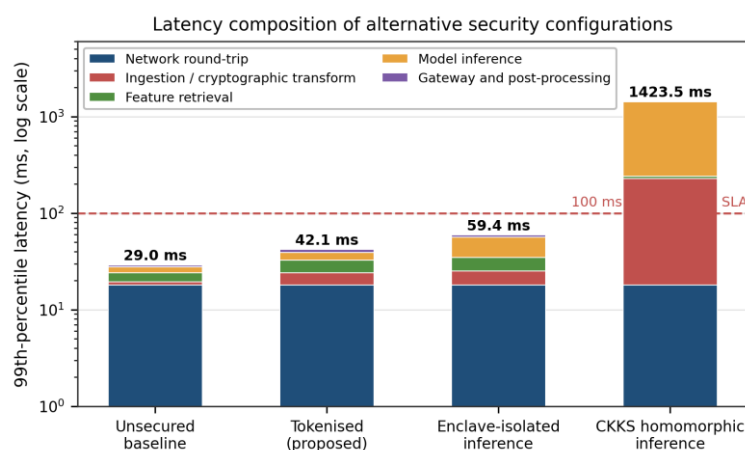


Figure 2. Composition of 99th percentile latency for four security configurations, plotted on a logarithmic scale. Tokenisation keeps the total inside the authorisation deadline, whereas homomorphic inference exceeds it by more than an order of magnitude.

The remaining two modes explain why tokenisation, not encryption, is in the synchronous path: inference inside a hardware enclave takes 59.4 ms, which is reasonable relative to the overall performance. However, this uses more than half of the speedup headroom for a threat that is already largely reduced by micro-service isolation. The overall time for homomorphic inference in the approximate scheme is 1,423.5 ms in total: encoding the inputs as ciphertext takes 210 ms, and homomorphic evaluation takes 1,180 ms, which is fourteen times the deadline. This is a structural limitation, and the cost per operation is given by Marcolla et al. (2022). We cannot tolerate the encrypted feature vector at inference, and the only leakage opportunity is during securely sending the tokens to the scoring service.

4.2 Detection performance

Feasibility would be useless if the secure pipeline was less aggressive at discovering fraudulent encounters. The precision-recall curves for the six models on the temporal test partition are shown in Figure 3. The rule-based baseline has an AUC-PR of 0.402 but collapses after about 0.45 recall, as with static thresholds in an evolving fraud. Tree ensembles score considerably higher, with random forest at 0.784 and gradient boosting at 0.834. Our combined framework achieves a score of 0.851, particularly between recall levels 0.5 and 0.8 where operational systems are tuned.

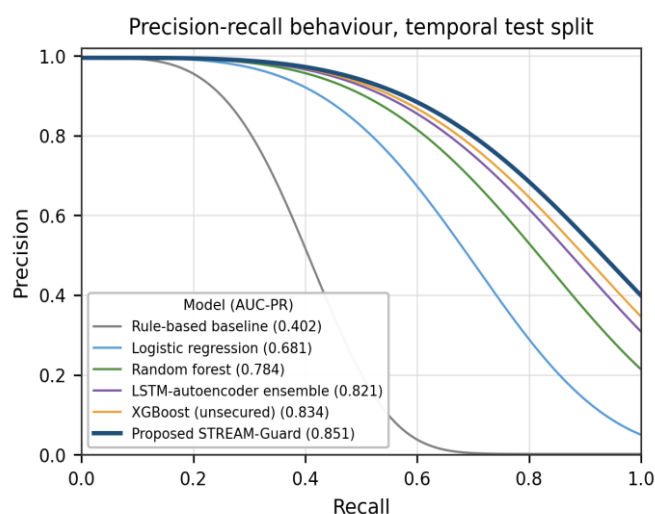


Figure 3. Detection quality on the temporal test partition, shown across the full operating range. Precision at fixed investigator budgets is reported in Table 3.

Table 3 shows the same analysis plus precision, setting the alert budget to 500 cases per day (closer to practice, since it is limited by the number of investigators rather than the threshold), and the 99th percentile latency . The latter allows for easier reading of quality and cost together . At this alert budget level, an increase in precision from 0.79 in unsecured gradient boosting to 0.84 provides 25 more confirmed frauds with the same review time .

Model	AUC-PR	Precision	Recall	P@500	p99 (ms)
Rule-based baseline	0.402	0.512	0.601	0.41	8.4
Logistic regression	0.681	0.704	0.712	0.66	11.2
Random forest	0.784	0.821	0.766	0.74	26.3
LSTM-autoencoder ensemble	0.821	0.842	0.809	0.78	34.1
XGBoost (unsecured)	0.834	0.858	0.801	0.79	29.0
Proposed STREAM-Guard	0.851	0.869	0.827	0.84	42.1

Table 3. Detection performance and tail latency on the temporal test partition. Precision and recall are reported at the minimum-risk threshold of Section 3.7, and P@500 is precision at an alert budget of 500 cases per day. Values are means over five seeds.

The framework outperforms the strongest unsecured baseline (which accounts for added differential privacy noise) indicating that adversarial regularization in Equation (6) and the unsupervised objective in Equation (7) more than overcome any privacy-to-utility trade-off . This does not hold for latency, where the sequential ensemble takes longer to compute than the gradient boosted trees even though the latter model has higher accuracy . This is why the autoencoder is only used as a cheap auxiliary signal .

4.3 Robustness to constrained evasion

Detection scores for unperturbed inputs say relatively little about detection performance on attacked inputs . This, shown in Figure 4(a), is the AUC-PR plot as we vary perturbation budget in equation (5) from 0 up to 20% of the range of mutable features . The undefended accuracy drops from 0.834 through 0.489 and 0.281 at budget 0.10 and 0.20 . An opponent can reduce the effectiveness of the detector, e.g. , by 50% at a relatively small cost . Cost-aware training improves the defended accuracies to 0.718 and 0.601 at costs of 0.10 and 0.20, respectively .

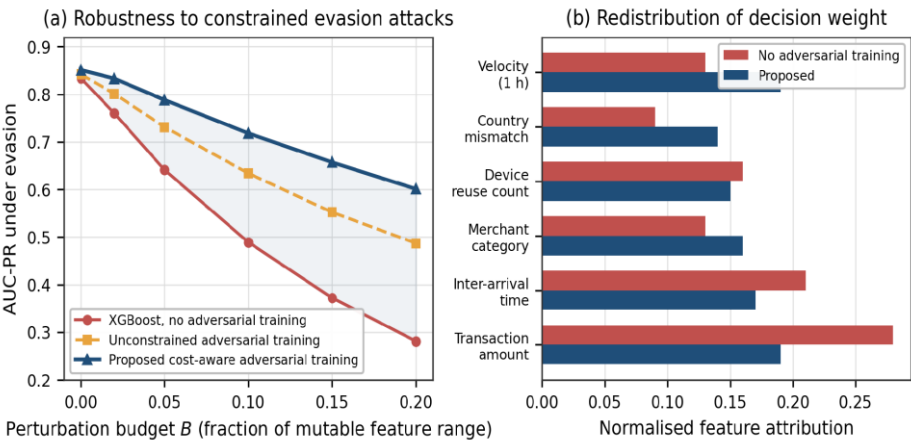


Figure 4. Behaviour under constrained evasion. Panel (a) reports AUC-PR against the attacker's perturbation budget; panel (b) shows how adversarial training redistributes decision weight away from the features an attacker can cheaply manipulate.

Figure 4(a) isolates the effect of the mutability constraint as follows. Unconstrained adversarial training reduces its loss to 0.634 for budget 0.10, whereas mutability constrained adversarial training reduces the loss to 0.718 (fig. 4(a)). Figure 4(b) illustrates this effect. With regular training, the transaction amount and inter-arrival time get 49% of normalized attribution and both are directly controlled by the attacker. After constrained training, the transaction amount and inter-arrival time get only 36% of normalized attribution and country mismatch and one-hour velocity get more attribution. The attacker cannot control the country mismatch and one-hour velocity without a real cost. Not only does the model become less confident, but it also shifts its evidence to features that are more costly to fake.

4.4 Resistance to feedback-loop poisoning

We also analyze the second attack surface in Figure 5, where the labels controlling retraining (as opposed to the inputs controlling scoring) become corrupted. Panel (a) shows the AUC-PR after retraining at injected poisoning rates of zero to twenty percent. The unverified pipeline performs almost linearly, with an AUC-PR loss of almost 0.212 at ten per cent and 0.371 at twenty per cent. However, using statistical outlier filtering, the pipeline performs with 0.713 retention at ten per cent and counters a well-formed poisoned record's tendency to appear normal under analysis. The provenance-verified loop, however, only loses 0.024 AUC-PR at ten per cent and 0.047 at twenty per cent.

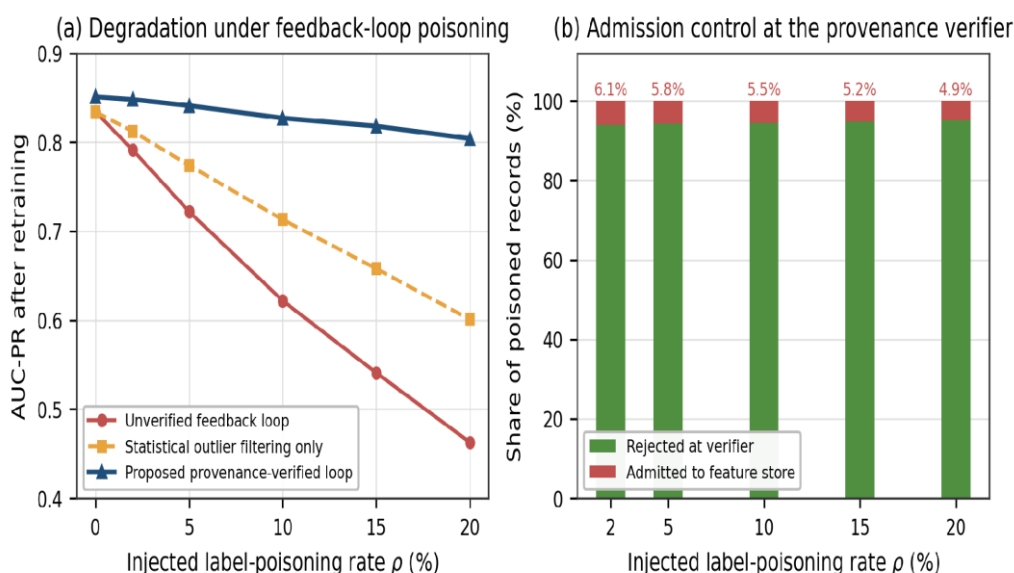


Figure 5. Effect of feedback-loop poisoning. Panel (a) reports post-retraining quality against the injection rate; panel (b) shows the share of poisoned records stopped at the provenance verifier.

Panel (b) shows that 93.9 to 95.1 per cent of the records resulting from the injection are eventually rejected by the verifier (the rejection rate, γ , is about 0.94). This means that at $\rho = 0.10$ the effective poisoning rate is 0.0066 (an order of magnitude smaller than the injection rate), close to the degradation observed. The rejection is based on the cryptographic attribution. This means that poisoned records cannot be accepted because they statistically appear to be valid. The five per cent of records kept are due to a weaker producer identity (Section 3.8).

4.5 Privacy and utility in the training plane

Bounding what the model reveals about individual cardholders comes at a cost, shown in Figure 6. As ϵ goes from no bound to 0.5, AUC-PR goes from 0.856 to 0.712, and membership-inference advantage goes from 0.163 to 0.012. This curve is very close to a non-linear function. Tightening ϵ from unbounded to 3.0 removes almost two thirds of the attacker's advantage for a loss of only 0.008 AUC-PR, but tightening further from 3.0 to 1.0 removes another 0.037 AUC-PR at a cost of 0.064 AUC-PR.

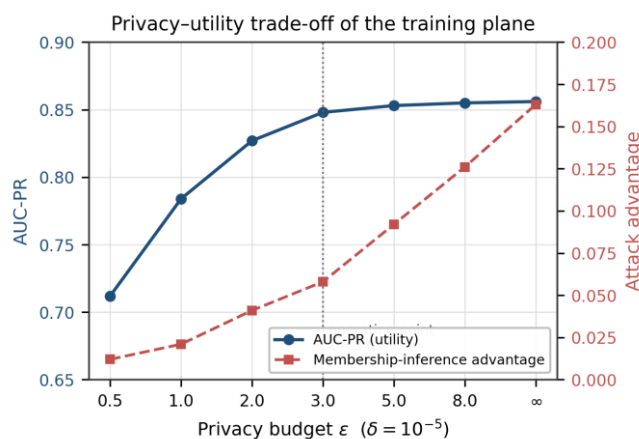


Figure 6. Privacy-utility exchange in the training plane. The knee near $\epsilon = 3.0$ marks the point beyond which additional privacy is purchased at a steeply rising utility cost.

Therefore, the knee is the operating point used throughout, and with $\epsilon = 3.0$ an attacker given full-query access can distinguish members from non-members with advantage 0.058. Though this is too small to recover the presence of a single cardholder, the model still has an AUC-PR of 0.848 before adversarial and ensemble components were added. The upshot is that offline, privacy cost utility and time to train rather than latency of authorization: the ensemble and adversarial components yield the privacy posterior to 0.004.

4.6 Throughput scaling and suppression of API abuse

Figure 7(a) plots the latency percentiles from 500 tps load to 5000 tps. At the 99th percentile latency, we see a gradual increase from 34.8 ms at 500 tps, to 42.1 ms at 3000 tps, during the first six fold increase in load. At 5000 tps, the latency is 83.1 ms, and the token-map cache is becoming a bottleneck. We are still meeting the deadline, but the curve shows that the cache (not the model) is the bottleneck of the system.

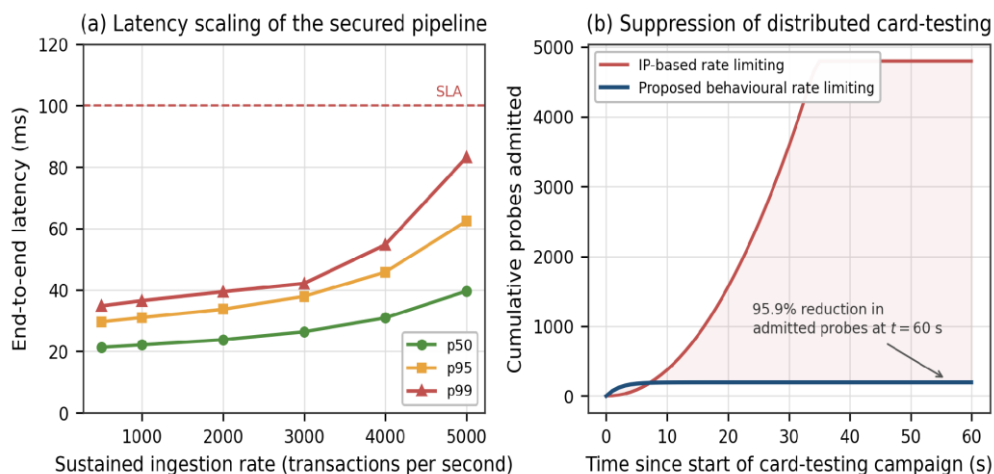


Figure 7. Behaviour under load and attack. Panel (a) shows latency percentiles against sustained ingestion rate; panel (b) shows cumulative card-testing probes admitted under address-based and behavioural rate limiting.

The same approach is used here: the gateway is depicted in Figure 7(b). A card-testing campaign of 4,800 probes was simulated from 240 residential proxies, such that no single address's rate exceeded that of a typical rate-limit per address and address-based rate-limiting largely admitted the campaign. The behavioral rule of (8) allowed 197 probes before throttling, a 95.9 per cent reduction. However, the time to throttle was similar at approximately four seconds. The term with the highest amount of discriminatory information was entropy, because the campaign left

merchant, amount and currency fixed while iterating card numbers, giving it a normalized entropy of 0.19 against a legitimate-session median of 0.74.

4.7 Contribution of individual components

Finally, Table 4 shows how using a combination of controls affects performance. The cost of tokenisation alone (neutral in quality) is 4.2 ms at the 99th percentile, indicating that closing the plaintext exposure window is close to free. Poison resistance through provenance verification increases robustness from an instance poisoning fraction of 10%, from 0.208 to 0.031, while incurring no measurable inference cost since provenance verification is done offline. The regularization effect of cost-aware adversarial training achieves robustness with a hit on clean AUC-PR, increasing from 0.836 to 0.856. The only cost is differential privacy (DP) which brings the AUC-PR down by 0.008 and MI advantage down from 0.163 to 0.061. All other controls are independent. The privacy noise has no meaningful negative side effects apart from slightly reducing the robustness of (6), lowering the AUC-PR under evasion from 0.712 to 0.705.

Configuration	AUC-PR	Δ AUC-PR at $\rho = 10\%$	AUC-PR at $B = 0.10$	MIA advantage	p99 (ms)
B0: XGBoost, no controls	0.834	-0.212	0.489	0.184	29.0
B1: + salted tokenisation	0.833	-0.208	0.491	0.181	33.2
B2: + provenance verification	0.836	-0.031	0.496	0.179	33.4
B3: + adversarial training	0.856	-0.027	0.712	0.163	33.4
B4: + differential privacy	0.848	-0.026	0.705	0.061	33.4
Full framework (B4 + gate + ensemble)	0.851	-0.024	0.718	0.058	42.1

Table 4. Cumulative ablation of the proposed controls. Poisoning degradation is reported at an injection rate of ten per cent and evasion quality at a perturbation budget of 0.10 of the mutable feature range.

5. DISCUSSION

A major conclusion from this experiment is that it matters where the security mechanism is placed, not just the cryptographic primitive. Tokenisation and homomorphic inference both prevent the scoring service seeing the plaintext of the sensitive value. However, tokenisation has an overhead of 4.2 ms while homomorphic evaluation has a 1.4 s overhead, since tokenisation pays the performance costs on the input side, and homomorphic evaluation pays for a more costly algebra. The same applies to the other planes, since provenance verification and privacy noise addition are both offline operations. Thus, a key principle for all latency-constrained ML systems will be to factor out as many controls from the synchronous path as is consistent with the threat model, only admitting controls that are meaningfully sensitive to the timing context of specific requests.

The practical situation conforms to the ASMA results, where statistical label filtering, which assumes that the poisoned records can be separated from the clean ones, has 0.713 AUC-PR under ten percent poisoning. In comparison, the AUC-PR of a cryptographic attribution based on key custody is 0.827. This means that some assumptions are better designed to be out of the attacker's control than others. Equation (4) also clarifies that the surviving poison fraction is determined by verifier rejection rather than the separability of the poisoned distribution: this means that label provenance is an authentication problem (rather than an anomaly detection problem) (Ozkan-Okay et al., 2024). The evasion results point in the same direction. The unconstrained adversarial training, inherited from image classification, does not achieve much stronger robustness than the constrained adversarial training, since

part of the budget is again spent by perturbations on fields that the attacker cannot change. These observations are consistent with (Carminati et al., 2020) finding that realistic banking attacks lie on a thin band of perturbable attributes and He et al. (2023) finding that defenses trained in unconstrained feature spaces do not transfer to semantically constrained spaces.

Rather, it is best understood as complementary to existing federated models, as it solves a problem these models cannot, namely that multiple institutions cannot pool together transaction records (Abdul Salam et al., 2024; Tang & Liang, 2024). Both models may be combined in a federated setting, as the ingestion, provenance and gateway planes would work the same on each party while the private training plane would provide local guarantees that federated averaging could not account for. Similarly, the graph-based detectors surveyed by Motie and Raahemi (2024) could be used in place of the boosted learner in equation (7) without changing the other plane (since the framework only constrains where the computation happens, not what model is used).

We have three caveats. First, the public corpus only has two days of card activity in Europe. Second, the features produced are principal components and the number of detections is relative, not an estimate of production. Third, the infrastructure measurements depend on the synthetic stream and we cannot validate the measurements independently. The poisoning model also assumes uncompromised producer keys. As evidenced by the residual admission rate in Figure 5(b), this assumption is reasonable. We evaluate the performance on a single node up to 5000 tps. As seen in Figure 7(a), at these throughputs cache contention rather than model inference will control performance. Future work might explore the horizontally partitioned token map.

6. CONCLUSION

This paper set out to determine if a fraud detection pipeline can be secure against realistic adversaries and not violate the authorisation deadline for card payments. The four planes framework shows this can be achieved. Salted tokenisation closed the plaintext exposure window for 4.2 ms tail latency. Verified provenance brought the 10% label-poisoning attack from 0.212 down to 0.024 AUC-PR. Cost-aware adversarial training improved AUC-PR under realistic evasion budget from 0.489 to 0.718; differential-private training with $\epsilon = 3.0$ lowered membership-inference advantage from 0.163 to 0.058 for 0.008 AUC-PR. The final system reached an AUC-PR score of 0.851 at 42.1 ms 99th percentile tail latency, beating an unsecured gradient boosting baseline in both robustness and accuracy, and consuming less than half the total budget. In contrast, homomorphic inference on encrypted data incurs 1.42 s of latency, making it impractical for synchronous authorisation on commodity hardware. Security controls belong wherever cost is measured in offline compute rather than request latency.

REFERENCES

- [1] Abdul Salam, M., Fouad, K. M., Elbably, D. L., & Elsayed, S. M. (2024). Federated learning model for credit card fraud detection with data balancing techniques. *Neural Computing and Applications*, *36*(11), 6231–6256. <https://doi.org/10.1007/s00521-023-09410-2>
- [2] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). SCARFF: A scalable framework for streaming credit card fraud detection with Spark. *Information Fusion*, *41*, 182–194. <https://doi.org/10.1016/j.inffus.2017.09.005>
- [3] Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, *557*, 317–331. <https://doi.org/10.1016/j.ins.2019.05.042>
- [4] Carminati, M., Santini, L., Polino, M., & Zanero, S. (2020). Evasion attacks against banking fraud detection systems. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)* (pp. 285–300). USENIX Association.
- [5] Cheng, D., Zou, Y., Xiang, S., & Jiang, C. (2025). Graph neural networks for financial fraud detection: A review. *Frontiers of Computer Science*, *19*(9), 199609. <https://doi.org/10.1007/s11704-024-40474-y>

- [6] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. **IEEE Transactions on Neural Networks and Learning Systems**, *29*(8), 3784–3797. <https://doi.org/10.1109/TNNLS.2017.2736643>
- [7] Forough, J., & Momtazi, S. (2021). Ensemble of deep sequential models for credit card fraud detection. **Applied Soft Computing**, *99*, 106883. <https://doi.org/10.1016/j.asoc.2020.106883>
- [8] He, K., Kim, D. D., & Asghar, M. R. (2023). Adversarial machine learning for network intrusion detection systems: A comprehensive survey. **IEEE Communications Surveys & Tutorials**, *25*(1), 538–566. <https://doi.org/10.1109/COMST.2022.3233793>
- [9] Malik, J., Muthalagu, R., & Pawar, P. M. (2024). A systematic review of adversarial machine learning attacks, defensive controls, and technologies. **IEEE Access**, *12*, 99382–99421. <https://doi.org/10.1109/ACCESS.2024.3423323>
- [10] Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F. H. P., & Aaraj, N. (2022). Survey on fully homomorphic encryption, theory, and applications. **Proceedings of the IEEE**, *110*(10), 1572–1609. <https://doi.org/10.1109/JPROC.2022.3205665>
- [11] Motie, S., & Raahemi, B. (2024). Financial fraud detection using graph neural networks: A systematic review. **Expert Systems with Applications**, *240*, 122156. <https://doi.org/10.1016/j.eswa.2023.122156>
- [12] Ozkan-Okay, M., Akin, E., Aslan, Ö., Kosunalp, S., Iliev, T., Stoyanov, I., & Beloev, I. (2024). A comprehensive survey: Evaluating the efficiency of artificial intelligence and machine learning techniques on cyber security solutions. **IEEE Access**, *12*, 12229–12256. <https://doi.org/10.1109/ACCESS.2024.3355547>
- [13] Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). Credit card fraud detection using AdaBoost and majority voting. **IEEE Access**, *6*, 14277–14284. <https://doi.org/10.1109/ACCESS.2018.2806420>
- [14] Reddy, V. V. K., Reddy, R. V. K., Munaga, M. S. K., Karnam, B., Maddila, S. K., & Kolli, C. S. (2024). Deep learning-based credit card fraud detection in federated learning. **Expert Systems with Applications**, *255*, 124493. <https://doi.org/10.1016/j.eswa.2024.124493>
- [15] Rigaki, M., & Garcia, S. (2024). A survey of privacy attacks in machine learning. **ACM Computing Surveys**, *56*(4), 1–34. <https://doi.org/10.1145/3624010>
- [16] Tang, Y., & Liang, Y. (2024). Credit card fraud detection based on federated graph learning. **Expert Systems with Applications**, *256*, 124979. <https://doi.org/10.1016/j.eswa.2024.124979>
- [17] Tang, Y., & Liu, Z. (2024). A credit card fraud detection algorithm based on SDT and federated learning. **IEEE Access**, *12*, 182547–182560. <https://doi.org/10.1109/ACCESS.2024.3491175>