GIU Berlin: IET Networks

Bachelor Thesis

MANTA: Measuring the Detection-Privacy Trade-off in Metadata-Only Mobile Traffic Monitoring

A public-corpus study of encrypted-traffic anomaly detection, telemetry reduction, and observer leakage

Author:	Mahmoud Elfeel
Supervisor:	Prof. Mohamed Ashour
Industry Partner:	Resistine GmbH
Submission Date:	July 2026

Abstract

This thesis investigates MANTA (Mobile Anomaly and Network Threat Analysis), an anomaly-first intrusion-detection artifact for encrypted mobile and endpoint traffic. The security problem is that threat detection needs evidence about suspicious behavior, while modern mobile traffic is largely encrypted and payload inspection would be both technically difficult and privacy-invasive. MANTA therefore studies whether metadata-only flow windows can retain enough signal for real-time anomaly detection while reducing the amount of sensitive telemetry that must be collected or exported. The implemented artifact spans three modules: an Android endpoint based on `VpnService` [5], a backend adapter for authenticated ingest and analyst feedback, and a reproducible machine-learning pipeline for training, privacy benchmarking, and model export.

The evaluation uses a public-only merged corpus of 6,034,487 flows drawn from seven dataset sources and 904,371 aggregated 60-second windows. It compares current on-device local detectors, backend models, privacy-local scoring, privacy-student/federated prototypes, and privacy-reduced feature views under grouped, source-aware splits. Under the shared Android adaptive-window comparison, the recall-oriented "High-recall RF" random forest reaches F1 0.961, the dedicated "Privacy RF" artifact reaches F1 0.924, the calibrated full-feature backend boosted-tree model reaches F1 0.890, and the quieter "Low-FPR RF" random forest reduces false positives at F1 0.826.

The privacy evaluation is framed as a telemetry-minimization problem rather than as a separate security goal. *Release leakage* measures what a recipient can infer from the feature representation that MANTA intentionally exports or shares. *Observer leakage* measures what a passive on-path observer can infer from the encrypted traffic sequence before MANTA applies any export-time reduction. The reduced privacy views (`medium` and `strict`) trade anomaly utility for lower app-identification leakage in the bounded new-dataset privacy reports: exact app-identification leakage falls from 0.282 normalized leakage in the full view to 0.151 and 0.135, respectively. However, app-family, destination-behavior, and dataset-source leakage remain high, and the encrypted-flow sequence audit shows severe observer leakage, with normalized leakage 0.544 for app identification and 0.981 for dataset-source inference. I conclude that MANTA improves control over released detection telemetry, but does not provide transport-level privacy against passive traffic analysis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Cybersecurity monitoring is useful only when it helps protect systems against threats. In this thesis, a threat is behavior, software, or an actor-controlled condition that can compromise a security objective such as confidentiality, integrity, or availability [21]. Examples in mobile network traffic include data-exfiltration behavior, command-and-control style communication, beaconing, unusual destination changes, or traffic patterns associated with malware. An intrusion-detection system (IDS) tries to identify such behavior from observable evidence rather than from the attacker's intent directly.

Mobile endpoints continuously communicate across Wi-Fi, cellular, enterprise, and home networks, yet defenders rarely have visibility into those flows without either rooting the device or inspecting sensitive payload content. Android's `VpnService` provides a practical middle ground by enabling user-space packet interception on non-rooted devices [5]. This makes endpoint telemetry possible, but the design must satisfy two competing requirements: detection quality must be operationally useful, and the evidence collected for detection must not expose unnecessary private behavior.

This tension is especially acute for encrypted traffic. Payload encryption removes content from inspection, but it does not remove all side channels. Flow timing, volume, burst shape, destination characteristics, and session structure can still reveal application identity or user behavior [15, 28]. Privacy is therefore not a separate topic added after detection; it is a constraint on how much telemetry the detector collects, stores, exports, and shares while trying to detect threats.

1.2 Terminology and Security Framing

MANTA uses the following terms throughout the thesis:

- **Threat**: behavior or software that can harm confidentiality, integrity, availability, or security accountability. MANTA focuses on network-behavior evidence of such threats, not on payload inspection or legal attribution.
- **Flow metadata**: packet-derived counters and labels such as timing, direction, byte counts, protocol, port category, app attribution, and destination/context features. This is similar in spirit to flow-monitoring systems such as NetFlow/IPFIX [8, 9].
- **Anomaly detection**: scoring behavior as unusual relative to learned or observed baselines, a common approach when labels are incomplete or adversarial behavior changes over time [7, 24].
- **Release privacy**: what a backend, analyst, researcher, or other recipient can infer from the feature representation that MANTA intentionally exports or shares.
- **Observer leakage**: what a passive network observer can infer from the original encrypted traffic sequence before MANTA applies hashing, bucketization, deletion, or other export-time reductions.
- **Normalized leakage**: an empirical attack-success score normalized against the baseline difficulty of the inference task. It is a measurement convenience, not a formal privacy guarantee.
- **Privacy student**: an experimental model trained to preserve anomaly-detection utility from a reduced feature view while adversary heads test whether sensitive attributes can still be inferred.
- **Federated simulation**: an experimental collaborative-learning path that trains on reduced or learned representations rather than centralizing raw full-feature tables. The prototype uses a FedProx-style objective for heterogeneous clients [1, 2, 16].

Operationally, MANTA can be described as a pipeline. It consumes Android packet/flow metadata and public training traces; it transforms those inputs into per-app feature windows, model scores, threshold decisions, and privacy-reduced representations; it outputs local alerts, optional backend triage records, trained model artifacts, and evaluation reports. This input-processing- output framing is important because each stage has a different privacy and security role.

1.3 Problem Statement

I study whether a practical, metadata-only IDS can be built and evaluated as an end-to-end artifact while keeping the privacy boundary explicit. The project asks whether MANTA can:

1. capture and aggregate application-attributed flow metadata without payload inspection,
2. detect suspicious traffic behavior from Android-computable feature windows,
3. compare local, backend-assisted, privacy-reduced, and hybrid deployment paths,
4. quantify the utility and privacy cost of reducing exported detection telemetry, and
5. distinguish controllable release leakage from passive observer leakage visible in encrypted traffic sequences.

The evaluation follows the same artifact through detection, release-view reduction, leakage attacks, observer fingerprinting, and reproducibility packaging. That gives one place to compare utility, privacy, and operating constraints instead of treating them as separate experiments.

1.4 Research Questions

- RQ1: To what extent can Android-computable metadata windows support anomaly detection across a heterogeneous public encrypted-traffic corpus under grouped, source-aware evaluation?
- RQ2: Which deployment path gives the best operational trade-off among on-device, backend-assisted, privacy-reduced, and hybrid scoring?
- RQ3: How much anomaly-detection utility remains when exported telemetry is reduced through bucketization, hashing, deletion, or privacy-student representation learning?
- RQ4: What sensitive information remains inferable from released telemetry, and what information remains inferable to a passive observer outside MANTA's feature-level controls?

1.5 Contributions

- An end-to-end implementation that connects Android metadata capture, backend triage, and a reproducible machine-learning pipeline.
- A public-only training and evaluation corpus assembled from Android malware, enterprise, IoT, campus, and mobile-app traffic datasets with source/session-aware metadata carried through the entire pipeline.
- An evaluation protocol that compares local, backend-assisted, privacy-reduced, and hybrid variants under grouped source-aware holdouts instead of optimistic random splits.

- A telemetry-minimization evaluation that measures both detection utility and release leakage under reduced feature views.
- An observer-leakage benchmark that marks the boundary of feature-level privacy controls by measuring what remains inferable from encrypted-flow sequences.
- An empirical result showing that metadata coarsening can reduce exact app-ID leakage while still leaving substantial app-family, destination-behavior, dataset-source, and observer leakage.

1.6 Thesis Structure

Chapter 2 summarizes Android networking constraints, flow-based intrusion detection, encrypted-traffic analysis, privacy fingerprinting, and privacy-aware learning literature. Chapter 3 defines the system architecture, trust boundaries, and the distinction between release privacy and observer leakage. Chapter 4 details the implementation of the Android endpoint, backend adapter, privacy views, privacy student, and federated simulation. Chapter 5 presents the public-corpus evaluation methodology and results. Chapter 6 discusses privacy impact, ethics, and limitations. Chapter 7 closes with future work.

Chapter 2

Background and Related Work

2.1 Android Networking and `VpnService`

Android exposes packet interception on non-rooted devices through `VpnService`, which creates a user-space TUN interface and routes device traffic through an application-controlled forwarding path [5]. This design enables endpoint telemetry without kernel modification, but imposes practical constraints:

- a foreground service is required for long-running capture reliability [3],
- upstream sockets must be protected from tunnel recursion (`protect(...)`),
- packet processing must remain resource-aware to avoid user-visible overhead.

For this reason, MANTA restricts collection to metadata-level features and performs lightweight aggregation on-device rather than deep packet inspection.

2.2 Flow-Based Monitoring and Metadata-Only Detection

Flow telemetry abstracts packets into structured records keyed by protocol/endpoint tuples and timing/counter statistics. NetFlow v9 and IPFIX formalize this approach and define reusable fields for interoperable flow exchange [8–10, 14]. Compared to payload inspection, flow representations:

- reduce sensitive content exposure,
- compress observations for efficient local storage and export,

- retain behaviorally meaningful signals (volume asymmetry, burstiness, novelty, and frequency change).

Recent anomaly-detection systems such as Kitsune and N-BaIoT show that compact, flow-like representations can still be effective for intrusion detection when combined with lightweight autoencoder or ensemble machinery [19, 20]. MANTA adopts the same metadata-only philosophy, but targets encrypted mobile and endpoint traffic while emphasizing privacy-sensitive deployment constraints.

2.3 Encrypted Traffic Analysis and Observer Leakage

Encrypted transport does not eliminate traffic analysis. Website-fingerprinting work has repeatedly shown that timing, size, and direction patterns can reveal a user's activity even when payloads are fully encrypted. Deep Fingerprinting demonstrated that deep-learning attackers can break many lightweight defenses [15], and subsequent systems research has shown that efficient padding schemes remain difficult to deploy robustly in realistic settings [18]. Similar issues appear in mobile traffic: partial encrypted traces can support user-activity detection [23], and concurrent-flow relationships can be used to classify mobile applications from encrypted traffic [28].

These attacks motivate the release/observer split used in MANTA. A coarsened or anonymized export can reduce what a data consumer learns, while the original encrypted sequence remains visible to an on-path observer. I evaluate those two surfaces separately.

2.4 Anomaly Detection for Network Security

Anomaly detection commonly operates in limited-label settings, where unsupervised baselines are attractive because they can be trained on predominantly normal traffic [7]. Isolation Forest remains a practical reference method for tabular anomaly scoring due to strong performance/complexity trade-offs [17]. In operational security settings, threshold policy is often more consequential than raw score quality because base rates and alert costs strongly affect utility [24].

On-device deployment introduces additional constraints: inference must be lightweight, robust under battery pressure, and explainable enough for human triage. MANTA therefore combines statistical, exported local-model, and TFLite-compatible scoring with lightweight feature-contribution summaries that can be generated during inference.

2.5 Privacy-Preserving Traffic Representation

Traditional anonymization work such as prefix-preserving pseudonymization protects explicit identifiers but may still leak semantics through traffic structure. PD-PAn argues that preserving utility while defending against semantic attacks requires attention to both prefix relationships and distributional characteristics [13]. MANTA follows that broader view: it does not only hash identifiers, but also evaluates what remains inferable from reduced feature views after identifiers and strong destination hints have been coarsened or removed.

2.6 Privacy-Aware Representation Learning and Collaboration

Self-supervised and reconstruction-based learning for encrypted traffic have recently become promising tools for anomaly detection under limited labels [27]. At the same time, federated-learning research has explored how intrusion-detection models can be improved without centralizing raw data [1, 2]. MANTA uses these ideas in a limited but concrete form: a privacy student is trained from fuller supervision onto a reduced view, and a federated simulation operates on that learned latent space rather than on raw full-feature tables.

2.7 Positioning

MANTA differs from earlier work in three ways. First, it integrates Android capture, backend triage, privacy views, privacy attack benchmarks, and manuscript-ready evidence generation into one artifact set. Second, it evaluates a public-only corpus with grouped source-aware splits rather than relying on a small controlled trace or a purely random split. Third, it treats privacy as two separate empirical questions: what the released representation leaks, and what a passive encrypted traffic observer can still infer.

Chapter 3

System Architecture and Design

3.1 System Overview

The system consists of three modules:

- Android endpoint app (`VpnService`, flow capture, on-device scoring, UI).
- Backend adapter (authenticated ingest, queue/retry, triage, policy APIs, optional SIEM forwarding).
- ML pipeline (feature engineering, training, evaluation, threshold calibration, model export).

The architectural goal is to maximize local autonomy while still supporting optional centralized analysis and reproducible experimentation. Detection must remain possible without backend connectivity, but the same feature and policy definitions must also support export, privacy-view derivation, leakage benchmarking, and model retraining.

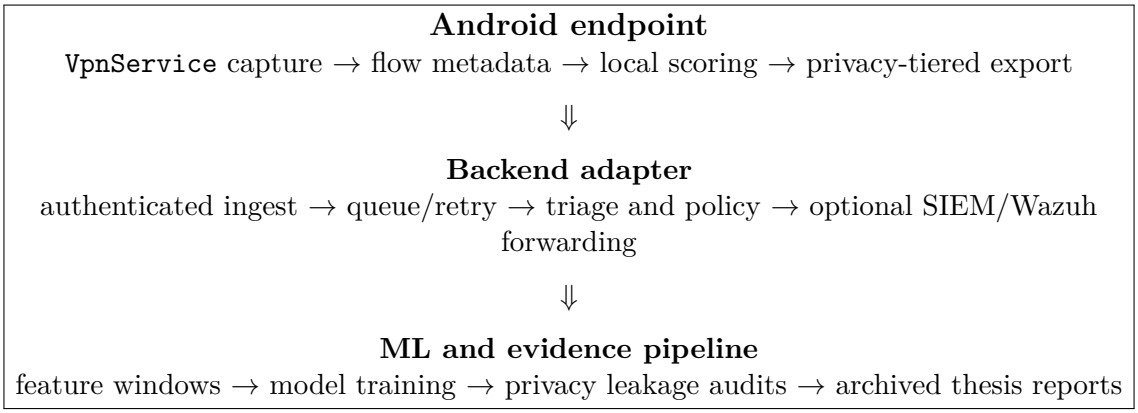

Figure 3.1: High-level MANTA artifact path from endpoint capture to thesis evidence.

3.2 Data Flow

The runtime data path is:

1. Packet capture via `VpnService` TUN interface.
2. Userspace forwarding to preserve normal network connectivity.
3. Packet parsing and app attribution to flow records.
4. Per-app rolling window aggregation (60 s default, with adaptive 30 s and 300 s modes for dense or sparse/service traffic).
5. Local anomaly scoring (statistical, multivariate, sequence, exported-tree, hybrid, TFLite-compatible, remote-assisted, and fusion modes).
6. Response-score adjustment, severity resolution, stability guards, false-positive budget checks, and alert-evidence policy.
7. Local persistence of flows/windows/alerts in Room.
8. Optional export of selected flow and alert events to backend APIs.

Policy synchronization is decoupled through background workers, allowing threshold updates and retention/export changes without requiring a new app build.

3.3 Threat Model and Assumptions

The primary defensive goal is detection of suspicious encrypted traffic behavior such as beaconing, unusual destination changes, remote-control style communication, data-exfiltration-like volume shifts, and malware-associated anomalies. In the security framing from Chapter 1, these behaviors matter because they may indicate compromise of confidentiality, integrity, availability, or security accountability. The design assumes that payloads remain encrypted and unavailable for inspection.

The thesis considers three privacy-relevant adversaries:

- a **data consumer** who receives an exported or derived representation and attempts to infer application or behavior properties from that release;
- a **passive observer** who watches encrypted traffic on the network path and performs fingerprinting from sequence metadata alone;
- a **central coordinator** in collaborative training who may receive learned updates or reduced latent features rather than raw full-feature traffic.

The model does *not* assume root compromise of the endpoint OS, payload decryption on the wire, or strong traffic-shaping defenses such as dummy-packet padding.

Trust boundaries:

- **Trusted**: endpoint app process, local encrypted settings, backend adapter process.
- **Partially trusted**: network transport path (TLS-protected but exposed to outages).
- **Untrusted**: remote application peers and public network infrastructure.

3.4 Privacy Model

MANTA separates two privacy and leakage surfaces:

- **Release privacy**: what can be inferred from the feature representation that is intentionally exported, shared, stored, or used as the privacy-reduced model input.
- **Observer leakage**: what can still be inferred by a passive eavesdropper from the raw encrypted traffic sequence itself.

This distinction drives the system design. Feature hashing, identifier suppression, bucketization, privacy-student training, and federated latent sharing act on the released representation. A passive observer sees the original encrypted sequence before those transformations occur.

Release privacy: controls and audits the feature representation intentionally exported by MANTA. This surface is affected by hashing, removal of direct identifiers, bucketization, and privacy-student learning.

Observer leakage: concerns an on-path observer who sees encrypted-flow timing, volume, direction, and sequence patterns before MANTA applies export-time reductions.

Boundary: feature coarsening can reduce release leakage, but it cannot hide the original traffic pattern from an observer on the network path.

Figure 3.2: Difference between release privacy and observer leakage in MANTA.

3.5 Telemetry-Reduction Design Decisions

The system applies data minimization and disclosure control at multiple points:

- metadata-only flow capture with no payload retention,

- hashed or removed direct identifiers in non-`off` privacy views,
- coarsened window features in the `medium` and `strict` release views,
- privacy-student training to learn a lower-leakage latent representation,
- federated-style training over privacy-student latents rather than raw full-feature tables.

These mechanisms reduce the representation that MANTA releases, but they are not formal privacy guarantees. The evaluation measures how much leakage remains after each transformation.

3.6 Deployment and Evaluation Boundaries

The design remains anomaly-first rather than anonymity-first. MANTA does not implement traffic-shaping defenses such as padding, dummy traffic, or timing obfuscation. Its privacy mechanisms operate on the representation released by the endpoint or backend pipeline, while the observer benchmark measures what remains visible on the network path. In other words, MANTA controls exported telemetry; it does not implement transport-level anonymity defenses.

Chapter 4

Implementation

4.1 Android Endpoint Implementation

4.1.1 VPN Capture and Userspace Forwarding

The endpoint capture loop is implemented around `FlowVpnService`, which establishes a TUN interface and starts a foreground service session. A dedicated userspace forwarder relays packets to upstream sockets while protecting those sockets from tunnel recursion via `VpnService.protect(...)` [5]. Parsed packets are aggregated into flow records keyed by app identity and network tuple attributes.

App attribution uses Android connection-owner resolution where available, and flows are persisted to Room-backed storage [4]. Background workers handle retention cleanup and export retries using WorkManager scheduling [6].

4.1.2 Feature Windows and On-Device Detection

The feature window builder aggregates per-app behavior windows and computes a richer metadata-only feature set than the first prototype. The default conceptual window is 60 seconds, but the runtime now selects 30-second windows for very dense traffic and 300-second windows for sparse, service, system, or background traffic. This adaptive choice reduces the instability of low-volume service windows while keeping high-volume interactive traffic responsive.

In addition to core counters and ratios, the implemented pipeline derives timing, novelty, directional balance, destination diversity, port/protocol diversity, burstiness, transport-summary features, destination-role ratios, and graph-inspired contextual signals such as destination concentration and transition rate. A second pass compares the current window with recent windows from the same app to produce baseline-deviation features such as flow-count deviation, byte-rate deviation, novelty shift, destination-diversity shift, trend

values, and consecutive burst-window count. These features are used internally for anomaly scoring, but only privacy-tier appropriate fields are exported in reduced views.

On-device inference supports several runtime modes:

- **Statistical**: an online per-app multi-scale z-score baseline with warmup and feedback-aware adaptation.
- **Multivariate**: a shrinkage-covariance Mahalanobis-distance detector for correlated feature shifts.
- **Sequence**: a lightweight transition-rarity detector over bucketed behavior states.
- **Exported tree**: JSON random-forest/boosted-tree artifacts with fixed feature order, app-family thresholds, and optional specialists.
- **TFLite-compatible**: a TensorFlow Lite inference path for a scalar anomaly-score model deployed as `models/anomaly.tflite` [25].
- **Backend**: an optional backend inference call over a privacy-filtered feature window.
- **Fusion**: a weighted and smoothed combination of available scorers.

The TFLite path is intentionally treated as a deployable inference interface rather than as a specific algorithmic claim. The Android scorer memory-maps `models/anomaly.tflite`, builds a float tensor in `FeatureWindow.portableFeatureOrder`, and expects one scalar score in $[0, 1]$. If a reconstruction-style or one-class neural model is used, the export must expose the reconstruction error or anomaly score as that scalar output; the Android scorer does not itself compare an input vector with a reconstructed vector. If TFLite loading or inference is unavailable, the selected TFLite mode falls back to the exported local model and then to the statistical detector.

Later Android experiments ship separate recall-oriented and precision-oriented tree artifacts and expose them with user-facing names. "High-recall RF" maps to the v13 recall-oriented random forest, "Low-FPR RF" maps to the v14 precision-oriented random forest, and "Adaptive hybrid RF" maps to a runtime hybrid of both. The hybrid mode scores both artifacts: malware-like and remote-access traffic keeps the larger score, service/system/other-app traffic is damped toward the quieter score, and other families use a moderate blend. This keeps the runtime deployable because it reuses the same metadata feature window and JSON tree evaluator rather than requiring a server model.

Alert generation is a separate stage after model scoring. The response score is adjusted by periodic-beacon evidence, concept drift, destination reputation/context, and data-quality penalties. Severity then passes through threshold profiles, app-profile adjustments, high-severity stability, false-positive budget enforcement, and an alert-evidence policy. The evidence policy can store weak pending candidates and promote them only when evidence

persists, when a correlated alert already exists, when known-danger destination evidence is present, or when an explicit single-window validation mode is selected. This separation prevents high-recall model windows from automatically becoming user-visible alerts while still allowing known-danger or validation traffic to surface promptly.

4.1.3 UI and Triage Workflow

The Compose UI exposes:

- consent/disclosure and capture controls,
- backend URL/token and export toggle,
- alert center with triage states (`OPEN`, `INVESTIGATING`, `RESOLVED`, `FALSE_POSITIVE`),
- local data purge and anonymized CSV snapshot export.

This supports both standalone local operation and analyst-assisted workflows.

4.2 Backend Adapter Implementation

The backend adapter is a FastAPI service with token authentication, strict request validation, and defense-in-depth response headers. Incoming events are enqueued in SQLite and forwarded with bounded retry/backoff; exhausted items move to a dead-letter queue for replay.

Beyond ingest, the adapter implements:

- alert triage APIs and incident summaries,
- per-device policy storage and auto-tuning hooks,
- policy simulation and quality summary endpoints,
- retraining sample export and forensics bundle export,
- optional Wazuh/Resistine connector routes [26].

This architecture isolates endpoint concerns from SIEM-specific integration complexity.

4.3 ML Pipeline Implementation

The ML pipeline is organized as reproducible command-line stages that now operate over a merged public corpus rather than only over a controlled synthetic trace:

- public-dataset normalization and merged-corpus manifest generation,
- grouped split audits and evaluation-protocol reports,
- Android feature-contract audits and Android-style window generation,
- on-device exported-tree and TFLite-compatible export,
- hard-example mining, label-conflict auditing, source/family balancing, and threshold policy sweeps,
- episode-level alert evaluation in addition to per-window classification,
- remote backend training across multiple model families,
- privacy ablation and metadata leakage attack benchmarking,
- encrypted-flow sequence fingerprint benchmarking,
- privacy-student training and federated simulation,
- comparison-matrix and performance-gate generation.

4.3.1 Privacy Views

The privacy mechanism is implemented as explicit feature views rather than as an afterthought. `off` and `low` retain the full feature set, while `medium` and `strict` replace many exact values with coarse or distribution-aware buckets. For example, activity, volume, duration, rate, burstiness, stability, and shape are represented through a small number of bucketized values, while direct host hints and several stronger destination/timing cues are removed from the exported representation.

This is not packet padding or transport obfuscation. The system does not add dummy traffic. The privacy transformation happens at the feature-representation level.

4.3.2 Privacy Student and Federated Path

The privacy-student model is trained on the reduced `medium` view with reconstruction-style pretraining and multiple adversary heads. Those adversaries try to recover app identity, app family, dataset source, context bucket, and destination behavior from the learned latent representation. The objective is to preserve anomaly utility while reducing leakage of sensitive traffic attributes.

The federated simulator then trains over the privacy-student latent representation using a FedProx-style objective [16] rather than over raw full-feature windows. FedProx is a federated optimization method that adds a proximal term to reduce client-drift problems

under heterogeneous local data. In MANTA, this is used only as a measurable collaborative-learning prototype; it does not prove secure deployment by itself.

4.3.3 Reproducibility Support

The `run_experiment_suite` path, cached privacy-view derivation, manifest reports, and comparison reports are all designed to support reproducibility. The current manuscript uses the bounded backend/privacy reruns and Android adaptive-window reports documented in the evidence bundle described in the appendix.

Chapter 5

Experimental Methodology and Evaluation

5.1 Experimental Setup

Primary quantitative results in this chapter are taken from the current bounded public-corpus rerun and the same-methodology Android adaptive-window comparison generated under `ml-pipeline/reports/`. The evidence package includes dataset and protocol manifests, current model reports, privacy ablation, metadata leakage, encrypted-flow fingerprinting, privacy-student and federated prototype reports, performance gates, and the app-selectable model comparison.

The core principle of the evaluation is to avoid optimistic random-row validation. Model and privacy reports instead use grouped splits keyed by dataset source, environment, session, app family, and time bucket wherever the report type supports them. The separate `evaluation-protocol.json` audit covers six grouped split strategies over a sampled 250,000-window subset to verify that the methodology itself is source-aware.

5.2 Datasets and Trace Collection

The corpus is a public-only merge of seven normalized datasets: Android Mischief, Android Spyware, CICAndMal2017, ITC-Net-Blend-60 scenario E, a Parrot-derived metadata slice, SDNCampus, and Westermo. The merged corpus contains 6,034,487 flows and yields 904,371 60-second windows for feature extraction. Table 5.1 summarizes the dataset composition reported by `dataset-manifest.json`.

The corpus spans malware-heavy Android traffic, benign campus/application traffic, and industrial network traces rather than a single-device lab trace. The resulting benchmark is harder than a controlled trace, but closer to the cross-source generalization problem

Table 5.1: Public-only corpus composition.

Dataset source	Flow rows	Sessions	Label profile
Android Mischief	3,011	8	anomaly-only slice
Android Spyware	4,924	1	mixed benign/anomalous
CICAndMal2017	2,587,130	2,119	mixed benign/anomalous
ITC-Net-Blend-60 E	17,199	211	benign-only
Parrot metadata slice	373,566	1	benign-only
SDNCampus	3,000,000	30	benign-only
Westermo	48,657	3	mixed benign/anomalous

MANTA is intended to study.

5.3 Metrics

- **Detection metrics:** precision, recall, F1, ROC-AUC, PR-AUC, and confusion matrix.
- **Privacy/utility metrics:** anomaly utility under `off`, `low`, `medium`, and `strict` views.
- **Metadata leakage metrics:** strongest-attacker accuracy and normalized leakage for app ID, app family, dataset source, context bucket, and destination behavior.
- **Observer leakage metrics:** encrypted-flow sequence fingerprinting accuracy and normalized leakage.
- **Performance metrics:** single-window and batch feature-extraction latency.

Normalized leakage is reported on a 0–1 scale after accounting for the baseline difficulty of the inference task. A value close to 0 means that the attacker is near the configured baseline or chance level for that task, while a value close to 1 means that the representation is highly identifying for the target attribute. The metric is therefore used as an empirical privacy-risk score rather than as a formal privacy guarantee.

5.4 Android-Deployable Experiment Methodology

The Android-deployable experiments ask how far the local Android path can improve while using only features that the endpoint can compute at runtime. They reuse the public corpus, but generate Android-style adaptive windows, enforce an explicit Android feature contract, and evaluate deployable random-forest artifacts that can be interpreted by the Android JSON tree scorer.

The methodology added five safeguards after the early local results proved too weak:

- **Feature-contract audit:** every added feature must come from Android packet/flow metadata, destination insight, or recent local windows. Server-only or label-only fields are excluded to avoid train/deploy skew.
- **Window-construction alignment:** training windows are generated to match the runtime app-window builder more closely, including adaptive windows for sparse/service traffic.
- **Split and balance audits:** reports track source, family, and label imbalance so global metrics cannot hide a single dominant dataset or app family.
- **Label-conflict handling:** rounded near-duplicate windows with both benign and malicious labels are downweighted or excluded rather than treated as clean contradictory supervision.
- **Window and episode metrics:** per-window precision/recall/F1 are paired with episode-level alert metrics, because the Android app surfaces correlated alert episodes rather than isolated CSV rows.

The Android UI uses descriptive names for the deployable artifacts. "High-recall RF" refers to the v13 recall-oriented random forest, "Low-FPR RF" refers to the v14 sequence/intelligence random forest, and "Adaptive hybrid RF" refers to the v15 runtime hybrid that combines both. These names are used in the app to describe operating behavior rather than temporary experiment numbers.

5.5 Anomaly-Detection Results

Table 5.2 summarizes the current headline anomaly-detection variants under one shared Android adaptive-window comparison. The backend was retrained on the same Android-style adaptive feature cache and updated to consume the full portable Android feature vector rather than the older partial remote feature payload. The backend threshold is calibrated on the Android validation holdout with an explicit false-positive budget instead of the earlier max-F1/recall-heavy threshold. The calibrated boosted-tree backend reaches precision 0.948, recall 0.838, F1 0.890, PR-AUC 0.976, and ROC-AUC 0.950 on the balanced holdout, with false-positive rate reduced to 0.110.

The results show usable signal in metadata-only windows, especially for the Android high-recall random forest and backend-side tree model. The backend parity rerun is no longer penalized by a partial remote feature payload, and the calibrated backend threshold removes the previous pathological 1.000-recall operating point. It still remains noisier than the Low-FPR RF path, but is no longer the worst false-positive option.

Table 5.2: Representative current model results under the shared Android adaptive-window holdout.

Model	Precision	Recall	F1	PR-AUC / ROC-AUC
High-recall RF	0.940	0.983	0.961	0.985 / 0.970
Privacy RF	0.922	0.927	0.924	0.917 / 0.909
Backend boosted tree, calibrated	0.948	0.838	0.890	0.976 / 0.950
Adaptive hybrid RF	0.981	0.713	0.826	0.984 / 0.968
Low-FPR RF	0.981	0.713	0.826	0.984 / 0.968

5.6 Experiment Progression and Negative Results

The experimental sequence was intentionally iterative. The first Android-local experiments used a small endpoint-safe feature set and simple deployable models. They were valuable because they proved that the app could score traffic locally, but their recall and F1 were too weak for an interactive security tool. The TFLite-compatible one-class path was also retained as a deployable interface experiment, but its early result was low-recall. This showed that having a mobile inference format is not enough; the feature contract, objective, and score calibration also have to match the operational detection problem.

Several later attempts were documented but not selected:

- **Larger expanded-window caches:** 500,000-window and early 250,000-window adaptive cache runs failed with exit code 137 because the builder materialized too many derived columns across the raw flow table. The memory-safe row builder completed, but took about 20 minutes for the expanded experiment cache.
- **Boosted-tree export:** the deployable boosted-tree candidate completed, but did not beat the random-forest variants on the selected split. It was therefore documented as a useful comparison rather than shipped as the default local model.
- **HistGradientBoosting benchmark:** the non-deployable histogram gradient boosting model estimated the value of a stronger tree learner, but it was slightly worse than the selected teacher-weighted random forest and lacked an implemented Android export path.
- **Pure service specialist:** a service-only specialist improved precision but lost too much recall. The result suggested that service behavior was internally broad and label-noisy; a specialist without better labels or alert-episode logic becomes overly conservative.
- **Full hybrid backend and v15 hybrid sweeps:** the Android runtime hybrid was implemented, but the Python exported-tree evaluator timed out while repeatedly scoring both 90-tree forests and sweeping episode policies. Full backend adaptive-window retraining with the heavier hybrid dual-channel family exceeded practical

runtime in this environment; the lighter boosted-tree backend completed on the same Android adaptive-window cache. Both failures are evaluation/training-path bottlenecks rather than mobile-runtime blockers.

These negative results shaped the final thesis interpretation. The largest gains came from fixing train/deploy mismatch, adding Android-computable temporal and destination context, auditing noisy labels, and evaluating alert episodes. Changing the classifier family alone was less important than aligning the data representation with the Android runtime and the user-visible alert policy.

5.7 Android-Deployable Model Development

The Android deployable path was iterated because early on-device models were not acceptable for an interactive app. The final reported app comparison therefore focuses on the current app-selectable artifacts in Table 5.3, while this section records the development logic qualitatively.

High-recall RF expanded the Android feature contract from the original compact set to local transport, destination-role, destination-transition, and per-app baseline-deviation features. It also added source/family balancing, hard-example mining, teacher-weighted random forests, and label-conflict downweighting. Conceptually, this helped because the main failure mode was not a single missing classifier family; it was that service windows with similar volume and diversity statistics were being treated as indistinguishable. Adding local context gave the model more axes on which to separate repeated service behavior from anomalous shifts.

Low-FPR RF then added short temporal memory, destination-intelligence features, a malware specialist, stricter service threshold policy, and stronger ambiguous-label handling. This was not intended to maximize global F1; it was intended to reduce the user-visible false alerts that a purely recall-oriented model could create.

Adaptive hybrid RF was added as a runtime hybrid rather than as a new trained model. It ships both the recall-oriented and quiet artifacts. Malware-like and remote-access traffic keeps the more sensitive score, while service/system/other-app traffic is damped toward the quieter score. This is conceptually attractive because it uses the sensitive path for recall and the quiet path for false-positive control.

Episode-level analysis motivated the Android runtime policy that stores weak pending evidence and requires persistence, evidence diversity, or a high-confidence bypass before surfacing an alert. This matters because user-visible alerts are correlated episodes over time, not isolated CSV rows.

The Android runtime was also changed after the offline model iterations. Explicit local model modes can surface single-window validation candidates instead of waiting for

repeated evidence, known danger destination evidence can bypass the normal model threshold, and the statistical detector has a cold-start risk floor for unusually suspicious first observations. These changes address a separate problem from model F1: user-visible alerts can be suppressed by stability gates even when the model score is nonzero. They are implementation safeguards and require a dedicated on-device alert study before being treated as offline detection-quality results.

5.7.1 Same-Methodology Current App Comparison

Because the later Android artifacts were trained and calibrated at different times, a separate comparison table was generated over one shared Android adaptive-window cache, one source-aware holdout split, and each model's exported threshold. Table 5.3 reports the balanced-holdout view, which is more informative for comparing precision and false-positive rate than the natural holdout because the natural sampled holdout is anomaly-heavy.

Table 5.3: Current app-selectable models under the same Android adaptive-window methodology.

Model	Precision	Recall	F1	FPR	Threshold
High-recall RF	0.940	0.983	0.961	0.148	0.825
Privacy RF	0.922	0.927	0.924	0.187	0.995
Backend boosted tree	0.948	0.838	0.890	0.110	0.999
Adaptive hybrid RF	0.981	0.713	0.826	0.033	0.790
Low-FPR RF	0.981	0.713	0.826	0.033	0.790

This comparison clarifies the operating trade-offs. High-recall RF is the strongest current single-artifact detector by balanced F1. Privacy RF now clears the target precision/recall gate under the same methodology, but it is not better than every normal model; it trades higher false-positive rate for stronger recall than the balanced/quiet path. Adaptive hybrid RF and Low-FPR RF local are conservative: they produce fewer false positives but miss more positive windows.

5.8 Release-Privacy Utility Trade-off

The release-privacy ablation results are shown in Table 5.4. In the new bounded merged-corpus report, `off` and `low` retain the full Android feature contract and therefore have identical utility. `medium` and `strict` reduce exact destination and fine-grained feature information, lowering app-identification leakage but also reducing detection utility. This is the expected privacy-utility trade-off once the report is aligned with the current feature set.

Table 5.4: Anomaly utility under privacy-reduced views.

View	Precision	Recall	F1	PR-AUC / ROC-AUC
`off`	0.651	0.962	0.776	0.759 / 0.857
`low`	0.651	0.962	0.776	0.759 / 0.857
`medium`	0.565	0.871	0.685	0.661 / 0.756
`strict`	0.515	0.893	0.653	0.582 / 0.703

5.8.1 Hashing Versus Deleting Reduced Features

An additional bounded experiment compared three privacy-reduction strategies under the same Android adaptive-window cache and the same source-aware holdout split. The goal was to distinguish semantic deletion from pseudonymization. In the *deleted* representation, the medium retained features are bucketed and every suppressed Android feature is set to zero. In the *hashed* representation, those same suppressed features are kept only as deterministic salted hash buckets. The current `medium-plus` representation instead uses broad bucketization over many Android features while zeroing direct identity/threat-tag ratio fields.

Table 5.5 shows the balanced-holdout result. Hashing suppressed features recovers substantial utility compared with deletion, but it also increases semantic leakage. This supports a key privacy interpretation: hashing is pseudonymization, not removal. It can preserve stable fingerprints that help detection, but those same fingerprints can help an attacker infer app family or app identity.

Table 5.5: Balanced-holdout comparison of reduced-feature deletion and hashing.

Representation	Precision	Recall	F1	App-family acc.	App-ID acc.
Full reference	0.974	0.954	0.964	0.557	0.344
Medium-plus bucketed	0.978	0.937	0.957	0.585	0.280
Hashed suppressed features	0.971	0.868	0.917	0.590	0.239
Deleted suppressed features	0.934	0.707	0.805	0.471	0.208

5.9 Metadata Leakage from Released Views

Table 5.6 summarizes the strongest metadata leakage tasks for the main release-privacy views. The important result is mixed. Exact app-ID leakage drops substantially from the `off` view to `medium` and `strict`. However, app-family, destination-behavior, and dataset-source leakage remain high even in the reduced views. This is why the release-privacy gate for `medium` and `strict` still fails.

In both `medium` and `strict`, exact app re-identification falls below the configured threshold, but the app-family and destination-behavior checks remain false. The reduced views

Table 5.6: Strongest metadata leakage by privacy view (normalized leakage).

View	App ID	App family	Dataset source	Destination behavior
`off`	0.282	0.722	0.927	0.999
`low`	0.272	0.799	0.902	0.988
`medium`	0.151	0.678	0.871	0.450
`strict`	0.135	0.603	0.646	0.580

therefore improve release privacy for one target while leaving broader semantic leakage unresolved.

5.10 Local and Remote Models Under Privacy Views

The local/remote privacy utility report now separates two different questions. The first is the real Android runtime path: the local privacy artifact receives the normal Android feature window and applies its own internal `medium-plus` input transform before scoring. The second is a release-view path: a remote consumer receives an already reduced `medium` or `strict` export view and tries to reuse a full-feature scorer. These are not equivalent.

On a balanced 5,000-window utility sample, the app-facing local privacy runtime path reaches precision 0.904, recall 0.956, and F1 0.930. The backend scorer reaches precision 0.724, recall 0.987, and F1 0.835 on the same balanced sample, but with a much higher false-positive rate. Under already reduced `medium`/`strict` release views, the direct full-feature local and remote scorers still have no useful operating mode because most expected raw inputs have been removed. This is why reduced export views require either a dedicated student/reduced-view model, an internally transformed local scorer, or explicit threshold/policy calibration rather than naive reuse of the full-feature production scorer.

5.11 Observer Leakage from Encrypted Traffic

The observer benchmark directly evaluates what a passive on-path observer could infer from the encrypted traffic sequence. This benchmark is deliberately separate from the release views, because the observer sees the original traffic before any export-time coarsening. Table 5.7 shows that observer leakage remains severe.

The reduced privacy views improve the representation that MANTA exports, but they do not protect against an observer who sees timing, burst structure, and sequence-level metadata on the network path. The experiment marks the boundary between feature-level privacy controls and transport-level traffic-analysis defenses.

Table 5.7: Observer leakage from encrypted-flow sequence fingerprinting.

Task	Accuracy	Normalized leakage	Top-5 accuracy	Strongest model
App ID	0.544	0.544	0.719	MLP
App family	0.810	0.783	0.979	MLP
Dataset source	0.984	0.981	1.000	MLP
Context bucket	0.046	0.044	0.107	MLP

5.12 Privacy Student, Federated Path, and Performance

The privacy-student model improves representation privacy only partially. In the prototype `medium`-view report, adversary accuracy is reduced strongly for app ID but remains notable for app family, dataset source, and destination behavior. The federated path remains a measured collaboration prototype rather than the main detector in the present configuration.

Performance gating is also incomplete. Table 5.8 summarizes the thesis-relevant gate outcomes. Single-window extraction meets the p95 target (22.5 ms) but misses the p99 target (87.9 ms versus the desired 40 ms). The feature-matrix construction itself is fast, but end-to-end single-window build time is not yet at the intended deployment threshold.

Table 5.8: Selected thesis gates and measured outcomes.

Gate	Target	Measured outcome	Result
Latency p95	≤ 40 ms	22.5 ms	Pass
Latency p99	≤ 40 ms	87.9 ms	Fail
Exact app-ID release leakage	below threshold	`medium`: 0.151, `strict`: 0.135	Pass
Semantic release leakage	below threshold	app-family and destination leakage remain high	Fail
Observer-leakage audit	low fingerprint leakage	app ID 0.544, dataset source 0.981	Fail
Detector quality	deployment-grade gates	same-methodology F1 0.961/0.924/0.890/0.826 for High-recall RF/Privacy RF/Backend/Low-FPR RF; backend FPR reduced to 0.110 after calibration	Mixed

5.13 Interpretation

The report set supports four findings and rejects one stronger interpretation.

- Metadata-only anomaly detection is technically feasible on a large public corpus.

- The bounded backend rerun shows that a backend hybrid model remains a strong centralized option when backend inference is acceptable.
- The Android-deployable comparison shows that the local path improves substantially when training windows, runtime windows, feature contracts, and label-conflict handling are aligned. The engineering conclusion is therefore "local detection is sensitive to deployability alignment", not simply "local detection is too weak."
- Release-privacy transformations can reduce exact app re-identification while preserving reasonable utility.
- Observer leakage remains severe without transport-level defenses.
- Runtime alert quality is not identical to per-window model quality: episode grouping, evidence persistence, known-danger bypasses, and false-positive budgets change what the user actually sees.
- MANTA should be treated as an evaluation framework and research artifact rather than as a product-ready privacy-preserving IDS.

Chapter 6

Privacy, Ethics, and Limitations

6.1 Privacy Impact

MANTA's privacy role follows from its threat-detection role: the detector needs telemetry to find suspicious behavior, but that telemetry can itself reveal user and application behavior. The system therefore enforces a metadata-only telemetry boundary: packet payloads are not stored or exported, and the released views progressively suppress direct identifiers and strong destination hints. From a GDPR-oriented perspective [12], this aligns with data minimization, purpose limitation, and storage-limitation principles. The design still processes behaviorally revealing metadata, so transparency, consent, and retention control remain necessary rather than optional.

MANTA implements several telemetry-reduction mechanisms:

- metadata-only capture instead of payload inspection,
- hashed or removed direct identifiers in the reduced views,
- coarse or distribution-aware bucketization of many released features,
- privacy-student learning over a reduced feature view,
- federated-style learning over privacy-student latents instead of raw full-feature tables,
- explicit auditing of both release privacy and observer leakage.

These mechanisms reduce what MANTA releases or shares, but they are not equivalent to a formal privacy guarantee.

The evaluation also distinguishes hashing from deletion. Hashing removes direct readability of a field, but deterministic salted hashes can still preserve stable equality and fingerprint structure. Deleting a suppressed feature removes it from the released information set entirely. The hash-versus-delete experiment in Table 5.5 therefore treats hashing as

pseudonymization rather than anonymization: it can recover detection utility, but it may preserve learnable app-family or app-ID signal for an attacker.

6.2 Ethical Considerations

Responsible deployment requires:

- informed consent before monitoring starts,
- understandable disclosure of what is and is not collected,
- bounded retention with user-triggered purge capability,
- strict access controls on backend triage endpoints,
- constraints against repurposing telemetry for non-security surveillance.

These points are consistent with EDPB guidance on transparency and lawful processing in digital services [11]. Security hardening practices from mobile application standards (e.g., OWASP MASVS) further reduce secondary risk [22].

6.3 Release Privacy versus Observer Leakage

The release and observer measurements expose different failure modes. In the release view, `medium` and `strict` substantially reduce exact app-ID leakage. Those same views still leak high-level semantics such as app family and destination behavior. The observer benchmark then shows that a passive eavesdropper can recover strong information from encrypted-flow sequences even when the exported representation is reduced.

In concrete terms, a release-privacy defender controls the representation that is exported or shared. A passive observer does not consume that exported representation; they consume the original traffic behavior on the network path. This is why export-time coarsening does not neutralize traffic-analysis attacks. Works such as Deep Fingerprinting and subsequent evaluations of website fingerprinting defenses reinforce this distinction [15, 18].

6.4 Interpretation of the Negative Result

Feature-level coarsening is not enough to provide strong privacy against passive observers. Many metadata-only monitoring designs rely on payload omission as their primary privacy argument; the MANTA results show that this is incomplete. The released representation can be made less revealing, but encrypted traffic remains fingerprintable at the observer level. This is a limitation result, not a claim that MANTA solves passive traffic analysis.

6.5 Leakage and Utility Evaluation

MANTA benchmarks privacy risk with two complementary views:

- utility and leakage under feature-reduced release views, and
- observer leakage from encrypted-flow sequence fingerprinting.

The two measurements keep utility and inference risk visible at the same time: a reduced export format has to retain threat-detection signal while reducing what secondary attackers can infer.

6.6 Limitations and Threats to Validity

- The anomaly models improve substantially in the bounded new-dataset backend rerun, but the result is not a full adaptive-window evaluation of every model family.
- The Android-deployable models use an adaptive-window protocol, while the bounded backend and privacy reports use practical source-balanced caps. These results should be compared within their documented protocols rather than treated as one fully unified full-corpus run.
- Full adaptive backend training was attempted but exceeded practical memory/time on the current 6.0M-flow corpus; the completed backend/privacy reports therefore use documented source-balanced raw-flow caps.
- The Adaptive hybrid RF runtime hybrid and the final alert-evidence policy are implemented, but still need a faster full-corpus offline evaluator and a dedicated on-device alert-quality study.
- The dataset mix is public and heterogeneous but still inherits each source dataset's own collection biases and labeling artifacts.
- The privacy-student and federated paths are measured prototypes, not production defenses.
- No packet padding, dummy traffic, or timing obfuscation is implemented, so passive-observer protection remains out of scope.
- Additional privacy attacks such as membership inference, poisoning, and adaptive counterfactual adversaries remain future work.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

I presented MANTA, an end-to-end framework for metadata-only anomaly detection over encrypted mobile and endpoint traffic. The project combines Android capture, backend-assisted triage, privacy-aware feature views, privacy-student and federated variants, and a reproducible public-corpus evaluation pipeline. The results identify a practical boundary: metadata coarsening can improve control over released detection telemetry, but it does not provide transport-level privacy against passive traffic-analysis attacks.

Answers to the research questions are:

- **RQ1 (metadata-only detection feasibility):** Metadata-only detection is feasible on a large public corpus, but evaluation protocol matters. Under the shared Android adaptive-window comparison, High-recall RF reaches F1 0.961, Privacy RF reaches F1 0.924, and the calibrated full-feature backend boosted-tree model reaches F1 0.890. The backend model has strong ranking quality, and threshold calibration reduces its false-positive rate from the earlier recall-heavy operating point.
- **RQ2 (variant comparison):** The deployment variants expose different operating points. High-recall RF is best by balanced F1, Privacy RF is the strongest reduced-view-compatible local path, the calibrated backend boosted tree is a centralized high-precision operating point, and Low-FPR RF/Adaptive hybrid RF trade recall for fewer false positives. The privacy-preserving variants remain useful for measurement, but not yet competitive as privacy-only collaborative detectors.
- **RQ3 (release privacy):** The reduced privacy views lower exact app-ID leakage, but the current bounded report shows a real utility trade-off: F1 falls from 0.776 in `off/low` to 0.685 in `medium` and 0.653 in `strict`. This supports a limited release-privacy result, not a free privacy improvement.

- **RQ4 (observer leakage):** The encrypted-flow sequence benchmark shows severe observer leakage. Release privacy and passive-observer protection are therefore not interchangeable, and feature-level coarsening alone is insufficient for strong traffic-analysis privacy.

I claim that MANTA is a reproducible encrypted-traffic IDS framework that separates controllable release privacy from passive-observer leakage, demonstrates the importance of deployability-aligned Android feature engineering, and shows that metadata-only coarsening can reduce exact app re-identification while remaining insufficient against stronger semantic and passive-observer inference.

Practically, the project delivers an artifact package: Android app, backend adapter, ML pipeline, report bundle, manuscript, and reproducibility appendix. Its value is strongest as an empirical systems study and evaluation framework rather than as a finished privacy-preserving product.

7.2 Future Work

- A faster full-corpus evaluator for the Adaptive hybrid RF runtime hybrid, preferably using persisted High-recall RF and Low-FPR RF score vectors rather than repeatedly interpreting both exported forests in Python.
- A dedicated on-device alert study that measures the final alert-evidence policy rather than only per-window model scores.
- Stronger remote and on-device anomaly models under the same grouped public-corpus protocol.
- Better TFLite export alignment, including scalar anomaly-score exports for reconstruction models or explicit Android-side reconstruction-error computation.
- Transport-level defenses such as padding or timing obfuscation if passive-observer protection is a hard requirement.
- Better privacy-student objectives that suppress app-family and behavior leakage more strongly while preserving anomaly utility.
- More realistic deployment studies on mobile overhead, battery, and long-running user acceptance.
- Stronger collaborative-learning paths with secure aggregation, client weighting, and robustness against poisoning.

Appendix A

Reproducibility and Artifact Package

A.1 Artifact Package

The main experiment artifacts used in this manuscript are the current public-corpus reports, bounded backend/privacy reruns, and Android adaptive-window model reports. During the experiment runs these reports were written to `ml-pipeline/reports/`; the submission snapshot mirrors the selected thesis reports under `exports/thesis-final-2026-03-22/reports/`:

- `ml-pipeline/reports/dataset-manifest.json`
- `ml-pipeline/reports/evaluation-protocol.json`
- `ml-pipeline/reports/android-model-evaluation.json`
- `ml-pipeline/reports/tflite-autoencoder-evaluation.json`
- `ml-pipeline/reports/remote-assisted-model.json`
- `ml-pipeline/reports/remote-assisted-model-new-dataset.json`
- `ml-pipeline/reports/privacy-ablation.json`
- `ml-pipeline/reports/privacy-ablation-new-dataset.json`
- `ml-pipeline/reports/privacy-leakage.json`
- `ml-pipeline/reports/privacy-leakage-new-dataset.json`
- `ml-pipeline/reports/model-privacy-utility-local-remote-new-dataset.json`
- `ml-pipeline/reports/traffic-fingerprint.json`
- `ml-pipeline/reports/privacy-student-report.json`
- `ml-pipeline/reports/federated-report.json`
- `ml-pipeline/reports/performance-gates.json`
- `ml-pipeline/reports/privacy-gate.json`
- `ml-pipeline/reports/full-model-matrix.json`
- `ml-pipeline/reports/android-model-v13-expanded-report.json`
- `ml-pipeline/reports/android-model-v13-expanded-comparison.json`

- ml-pipeline/reports/android-model-v13-expanded-error-report.json
- ml-pipeline/reports/android-model-v13-expanded-episode-report.json
- ml-pipeline/reports/android-model-v13-expanded-alert-policy-report.json
- ml-pipeline/reports/android-model-v13-expanded-label-audit.json
- ml-pipeline/reports/android-model-v13-expanded-split-balance-audit.json
- ml-pipeline/reports/android-model-v13-expanded-experiment-notes.md
- ml-pipeline/reports/android-model-v14-sequence-intel-report.json
- ml-pipeline/reports/android-model-v14-sequence-intel-comparison.json
- ml-pipeline/reports/android-model-v14-sequence-intel-error-report.json
- ml-pipeline/reports/android-model-v14-sequence-intel-episode-report.json
- ml-pipeline/reports/android-model-v14-sequence-intel-alert-policy-report.json
- ml-pipeline/reports/android-model-v14-sequence-intel-experiment-notes.md
- ml-pipeline/reports/android-model-v15-hybrid-report.json
- ml-pipeline/reports/android-model-v15-hybrid-experiment-notes.md

A.2 Core Reproduction Commands

The full public-corpus workflow is documented in `ml-pipeline/README.md`. The compact reproduction path is:

```
cd ml-pipeline
python -m pip install -e .[tflite,test]

python -m ml_pipeline.dataset_manifest \
  --input data/real/manta-real-training.csv \
  --output reports/dataset-manifest.json

python -m ml_pipeline.evaluation_protocol_report \
  --input data/real/manta-real-training.csv \
  --output reports/evaluation-protocol.json

python -m ml_pipeline.train_all \
  --input data/real/manta-real-training.csv \
  --runs-root experiment-runs

python -m ml_pipeline.run_experiment_suite \
  --input data/real/manta-real-training.csv \
  --output-dir experiment-runs/real-run-001 \
```

```
  --model-threshold 0.5 \
  --ids-threshold 0.55
```

The later Android-deployable iterations can be reproduced with the dedicated Android training and evaluation commands recorded in the corresponding report notes. High-recall RF (v13) trains the expanded-feature deployable random-forest family, while Low-FPR RF (v14) extends that path with sequence and destination-intelligence features. Adaptive hybrid RF (v15) is not a new training command; it evaluates the exported v13 and v14 artifacts together:

```
python -m ml_pipeline.hybrid_model_evaluation \
  --input data/real/manta-real-training.csv \
  --model-v13 \
  artifacts/android/anomaly-deployable-v13-expanded.json \
  --model-v14 \
  artifacts/android/anomaly-deployable-v14-sequence-intel.json \
  --output reports/android-model-v15-hybrid-report.json \
  --window-mode adaptive \
  --max-adaptive-windows 250000 \
  --label-strategy window \
  --max-fpr 0.05
```

In the current artifact set, the v15 command records an incomplete full-corpus evaluation because the Python exported-tree report path timed out. The runtime implementation is present in Android, but thesis-quality v15 metrics require a faster offline scorer or persisted v13/v14 score vectors. The Android alert-evidence policy changes are validated by the Android implementation and unit-test suite, but they are not yet represented by a complete public-corpus alert-quality report.

The backend was then rerun against the same Android adaptive-window cache used by the app-model comparison. The heavier hybrid dual-channel backend family exceeded practical runtime in this environment, so the completed parity backend artifact uses the boosted-tree backend family over the full portable Android feature contract:

```
python -m ml_pipeline.train_remote_backend_model \
  --input data/real/manta-real-training.csv \
  --output-model artifacts/backend/remote-assisted-model.json \
  --output-report reports/remote-assisted-model-parity-same-methodology.json \
  --window-mode adaptive \
  --max-adaptive-windows 5000 \
  --label-strategy window \
  --model-family gradient_boosted_tree
```

```
python -m ml_pipeline.calibrate_backend_threshold \
  --input data/real/manta-real-training.csv \
  --remote-model artifacts/backend/remote-assisted-model.json \
  --output-model artifacts/backend/remote-assisted-model.json \
  --output-report reports/remote-assisted-threshold-calibration-same-methodology.json
  --window-mode adaptive \
  --max-adaptive-windows 5000 \
  --label-strategy window \
  --max-eval-rows 5000 \
  --target-precision 0.85 \
  --target-recall 0.90 \
  --target-max-fpr 0.18

python -m ml_pipeline.compare_android_deployable_models \
  --input data/real/manta-real-training.csv \
  --output-json reports/android-backend-parity-comparison-same-methodology.json \
  --output-csv reports/android-backend-parity-comparison-same-methodology.csv \
  --output-md reports/android-backend-parity-comparison-same-methodology.md \
  --remote-model artifacts/backend/remote-assisted-model.json \
  --window-mode adaptive \
  --max-adaptive-windows 5000 \
  --label-strategy window \
  --max-eval-rows 5000

python -m ml_pipeline.privacy_ablation \
  --input data/real/manta-real-training.csv \
  --output reports/privacy-ablation-new-dataset.json \
  --window-mode bucket \
  --label-strategy window \
  --max-flow-rows 100000

python -m ml_pipeline.privacy_leakage_benchmark \
  --input data/real/manta-real-training.csv \
  --output reports/privacy-leakage-new-dataset.json \
  --window-mode bucket \
  --label-strategy window \
  --max-flow-rows 100000 \
  --max-class-rows 300
```

```
python -m ml_pipeline.model_privacy_utility \
  --input data/real/manta-real-training.csv \
  --local-model ../android-app/app/src/main/assets/models/anomaly-local.json \
  --remote-model artifacts/backend/remote-assisted-model.json \
  --output reports/model-privacy-utility-local-remote-new-dataset.json \
  --window-mode bucket \
  --label-strategy window \
  --max-flow-rows 25000
```

The normalized input CSV is derived from the public datasets listed in Table 5.1. Raw datasets are not redistributed with the thesis archive; the archive contains the derived report bundle used for the manuscript tables.

Bibliography

[1] Mamoun Alazab et al. Enhancing privacy-preserving intrusion detection through federated learning. *Electronics*, 2023. URL https://www.mdpi.com/2079-9292/12/16/3382.

[2] Tareq H. H. Aldhyani et al. Privacy-preserving fl-based ids for cyber-physical systems. *Mathematics*, 2024. URL https://www.mdpi.com/2227-7390/12/20/3194.

[3] Android Developers. Foreground services overview. https://developer.android.com/develop/background-work/services/foreground-services, 2026. Accessed: 2026-02-28.

[4] Android Developers. Save data in a local database using room. https://developer.android.com/training/data-storage/room, 2026. Accessed: 2026-02-28.

[5] Android Developers. Vpnservice. https://developer.android.com/reference/android/net/VpnService, 2026. Accessed: 2026-02-28.

[6] Android Developers. Workmanager. https://developer.android.com/topic/libraries/architecture/workmanager, 2026. Accessed: 2026-02-28.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, 2009. doi: 10.1145/1541880.1541882.

[8] B. Claise. Cisco systems netflow services export version 9. Technical Report RFC 3954, IETF, 2004.

[9] B. Claise, B. Trammell, and P. Aitken. Specification of the ip flow information export (ipfix) protocol for the exchange of flow information. Technical Report RFC 7011, IETF, 2013.

[10] B. Claise, B. Trammell, and P. Aitken. Information model for ip flow information export (ipfix). Technical Report RFC 7012, IETF, 2013.

[11] European Data Protection Board. Edpb guidelines 05/2020 on consent under regulation 2016/679. https://edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-052020-consent-under-regulation-2016679_en, 2020. Accessed: 2026-02-28.

[12] European Union. Regulation (eu) 2016/679 (general data protection regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016.

[13] Xiaodan Gu and Kai Dong. Pd-pan: Prefix- and distribution-preserving internet of things traffic anonymization. *Electronics*, 12:4369, 2023. doi: 10.3390/electronics12204369.

[14] IANA. Ip flow information export (ipfix) entities. https://www.iana.org/assignments/ipfix/ipfix.xhtml, 2026. Accessed: 2026-02-28.

[15] Mohsen Imani, Marc Juarez, Payap Sirinam, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018. doi: 10.1145/3243734.3243768.

[16] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html.

[17] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, pages 413–422, 2008. doi: 10.1109/ICDM.2008.17.

[18] Nate Mathews, James K. Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. Sok: A critical evaluation of efficient website fingerprinting defenses, 2022. Manuscript in the local thesis notes collection.

[19] Yair Meidan, Michael Bohadana, Yisroel Mathov, Yisroel Mirsky, Daniel Breitenbacher, Asaf Shabtai, and Yuval Elovici. N-baiot: Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 2018. URL https://arxiv.org/abs/1805.03409.

[20] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018. URL https://arxiv.org/abs/1802.09089.

[21] National Institute of Standards and Technology. Standards for security categorization of federal information and information systems. Technical Report FIPS PUB 199, National Institute of Standards and Technology, 2004.

[22] OWASP Foundation. Owasp mobile application security verification standard (masvs). https://mas.owasp.org/MASVS/, 2025. Accessed: 2026-02-28.

[23] Madushi H. Pathmaperuma, Yogachandran Rahulamathavan, Safak Dogan, and Ahmet Kondoz. Cnn for user activity detection using encrypted in-app mobile data. *Future Internet*, 14(2):67, 2022. doi: 10.3390/fi14020067.

[24] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, 2010. doi: 10.1109/SP.2010.25.

[25] TensorFlow. Tensorflow lite guide. https://www.tensorflow.org/lite/guide, 2026. Accessed: 2026-02-28.

[26] Wazuh. Wazuh api reference. https://documentation.wazuh.com/current/user-manual/api/reference.html, 2026. Accessed: 2026-02-28.

[27] Yuxiang Wu et al. Et-ssl: Self-supervised learning for anomaly detection in encrypted traffic. *Scientific Reports*, 2025. URL https://www.nature.com/articles/s41598-025-08568-0.

[28] Guoliang Xu, Ming Xu, Yunzhi Chen, and Jiaqi Zhao. A mobile application-classifying method based on a graph attention network from encrypted network traffic. *Electronics*, 12(10):2313, 2023. doi: 10.3390/electronics12102313.