Research Square

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

MobiShield: Federated Android Malware Detection with Adaptive Differential Privacy under Concept Drift

Amit Patel

amit.manit007@gmail.com

Maulana Azad National Institute of Technology

Dr. Deepak Singh Tomar

Maulana Azad National Institute of Technology

Dr. R. K. Pateriya

Maulana Azad National Institute of Technology

Yogesh Kumar Sharma

Sagar Institute of Science & Technology

Article

Keywords: Android malware detection, federated learning, differential privacy, concept drift, non-IID data, adaptive noise calibration, mobile security

Posted Date: June 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9745227/v1

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

Additional Declarations: No competing interests reported.

MobiShield: Federated Android Malware Detection with Adaptive Differential Privacy under Concept Drift

Amit Patel*[1], Deepak Singh Tomar[1], R.K. Pateriya[1], and Yogesh Kumar Sharma[2]

[1]Department of Computer Science and Engineering, Maulana Azad National Institute of Technology (MANIT), Bhopal 462003, Madhya Pradesh, India

[2]Sagar Institute of Science & Technology, Bhopal, 462036, India

Abstract

Android has become the dominant target for malware authors, and the threat is no longer static - attackers continuously repackage and obfuscate malicious code to evade detection. Federated learning offers a compelling path forward by letting mobile devices collaborate on a shared detection model without ever surrendering their raw application data. In practice, however, three problems consistently undermine this promise: devices contribute data that follow very different distributions (non-IID heterogeneity), the malware landscape drifts over time so that a model trained today degrades tomorrow, and the gradient updates devices send to the server can inadvertently leak sensitive behavioural information. Existing federated frameworks tackle at most one or two of these issues in isolation; none addresses all three together. We built **MobiShield** around the observation that drift and privacy are not independent concerns – when the malware distribution shifts, the appropriate level of noise

*Corresponding author. amit.manit007@gmail.com

to inject changes too. The framework brings together four tightly coupled modules. AHFS uses a per-client multi-head attention mask to re-weight the hybrid static-dynamic feature vector before local training, reducing the damage caused by heterogeneous data. DFDD runs a Hoeffding-based drift detector on each client and aggregates the local signals at the server through a KL-divergence-weighted EWMA, producing a single global drift score $\hat{Z}(t)$ every round. That score feeds directly into DPFL, which adjusts both the gradient clipping radius $C(t)$ and the noise multiplier $\sigma(t)$ in response – tightening privacy when drift is high and relaxing it once the distribution stabilises. Finally, ACAG acts as a quality gate, admitting a client's update only when its local quality score and drift score jointly satisfy the dual-gate criterion, preventing stale or corrupted gradients from reaching the aggregator. We evaluated MobiShield on four benchmarks spanning real and synthetic drift conditions: Drebin, CICMalDroid 2020, a chronological AndroZoo split covering 2019–2021, and a controlled synthetic drift dataset. Across these benchmarks, MobiShield reaches 96.8%, 97.6%, 94.7%, and 93.5% detection accuracy, improving on the strongest competing baseline by up to 2.3 percentage points while operating under a provable differential privacy guarantee ($\varepsilon \leq 8$, $\delta = 10^{-5}$, moments accountant). Ablation experiments show that each module contributes 4–7% to overall performance, and all pairwise comparisons are statistically significant at $p < 0.01$.

Keywords: Android malware detection; federated learning; differential privacy; concept drift; non-IID data; adaptive noise calibration; mobile security

1 Introduction

The Android operating system commands approximately 70% of the global mobile market share and powers over three billion active devices worldwide [1]. This dominance, combined with the open-source nature of the platform and the prevalence of third-party application markets, has made Android an exceptionally attractive target for malicious actors. According to the AV-TEST Institute, more than 450,000 new malicious programs and potentially unwanted applications are detected daily, while Kaspersky reported a 14% year-on-year increase in mobile threats in 2024, including a 33% surge in Trojans and a 150% rise in Trojan-dropper variants [2, 3]. The financial and privacy consequences are severe: mobile malware is implicated in banking fraud,

credential theft, ransomware deployment, and large-scale privacy breaches affecting millions of users. Consequently, robust, scalable, and privacy-aware Android malware detection has become a research imperative of the highest urgency.

Traditional machine learning (ML) and deep learning (DL) approaches to Android malware detection operate under a *centralised* paradigm: raw application artefacts—permissions, API call sequences, system call traces, and network traffic features—are aggregated at a single server, and a global classifier is trained over the pooled dataset [4, 5]. Although centralised models achieve high accuracy—exceeding 98% on controlled benchmarks [6]—this paradigm is fundamentally incompatible with real-world mobile deployment. Privacy regulations such as the General Data Protection Regulation (GDPR) and the growing sensitivity of application behavioural data preclude wholesale transfer of on-device information to centralised infrastructure. Furthermore, the geographically distributed nature of Android deployments makes data pooling operationally infeasible at scale. These constraints motivate a shift towards *collaborative* learning mechanisms that preserve the locality of training data.

Federated learning (FL) [7] addresses the centralisation barrier by enabling multiple clients—mobile devices, enterprise security sensors, or national CERT nodes—to collaboratively train a shared global model by exchanging only model gradients or parameter updates, never raw data. In the context of Android malware detection, FL has been explored through several recent frameworks. Fang et al. [8] proposed a comprehensive FL architecture for Android malware detection tested across multiple device classes. Xie et al. [9] introduced Andro-BCFL, combining blockchain trust management with FL-based secondary detection, achieving 97.87% detection accuracy. Chaudhuri et al. [10] proposed DW-FedAvg, a dynamically weighted aggregation strategy that reduces the influence of underperforming local models on the global update. Despite these advances, FL introduces its own distinct set of challenges in the malware detection setting, which existing frameworks address only partially and in isolation.

Three fundamental challenges critically limit the reliability of FL-based Android malware detection in real-world deployments, and together define the research gap this paper addresses.

Challenge 1: Client heterogeneity (non-IID data). Android devices exhibit highly heterogeneous application ecosystems: a corporate device encounters primarily enterprise-class adware,

while a consumer device may be exposed to banking Trojans or SMS-fraud malware. This produces non-IID data distributions across FL clients, which degrades convergence and the accuracy of standard aggregation schemes such as FEDAVG [7, 11]. Critically, non-IID distributions in malware detection also affect *which features* are most discriminative per client—a challenge that requires explicit per-client feature adaptation to resolve.

Challenge 2: Concept drift. The Android malware ecosystem is inherently non-stationary. New malware families emerge continuously, and existing families evolve through obfuscation, code injection, and repackaging [4], generating concept drift—a change in the joint distribution $P(\mathbf{x}, y)$ that renders previously learned decision boundaries suboptimal over time [12]. In the FL setting, concept drift is especially pernicious: client distributions shift *asynchronously* and in *heterogeneous directions*, so drift-affected updates from a subset of clients can contaminate the global model even when other clients are stable. The most closely related recent work, FL-MALDRIFT [13], addresses client-side drift through ADWIN, DDM, EDDM, and HDDM detectors with adaptive participation control, achieving 94.7% on Drebin and 96.8% on CICMalDroid 2020. However, FL-MALDRIFT provides no formal privacy guarantee, leaving the framework vulnerable to gradient-inversion [14] and membership inference attacks [15].

Challenge 3: Privacy leakage. Although FL avoids raw data sharing, gradient updates transmitted from clients to the aggregation server can reveal sensitive information. For Android malware detection, this is particularly concerning because model updates implicitly encode application behavioural signatures that, if reconstructed, could expose proprietary usage patterns or the presence of sensitive software on a device. Differential privacy (DP) [16] provides the gold-standard formal guarantee against such leakage. Yet integrating DP into a drift-adaptive FL framework involves a fundamental tension: larger model updates during high-drift periods demand commensurately larger DP noise to maintain the privacy guarantee, which in turn degrades model utility. Static noise schedules cannot navigate this tradeoff optimally; an *adaptive* coupling between drift detection and noise calibration is required.

Taken together, these three challenges define an important and unaddressed research gap. No existing FL framework for Android malware detection simultaneously provides: (i) heterogeneity-aware per-client feature adaptation, (ii) distributed concept drift detection with server-side

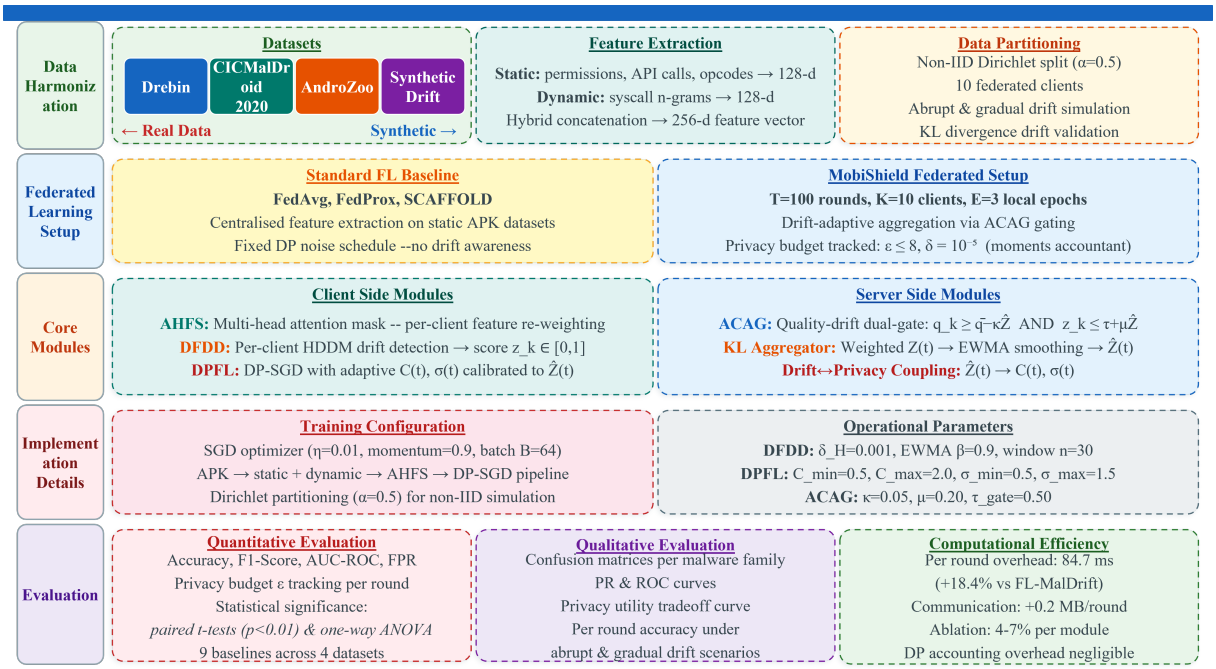


Figure 1. Overview of MOBISHIELD. The figure summarizes the main components of the proposed framework and their interaction in the federated Android malware detection setting.

aggregation, and (iii) formally proven, drift-adaptive differential privacy. FL-MALDRIFT [13] is the strongest existing baseline on challenges (i) and (ii), but provides no DP guarantee. Conversely, the DP-FL framework of Ciaramella et al. [17] provides formal privacy but neither adapts to concept drift nor addresses feature heterogeneity. This disjunction between privacy and drift resilience is not a mere design oversight—it reflects a genuine technical conflict requiring a new, principled coupling mechanism to resolve. Table 1 summarises this positioning relative to existing approaches.

This paper proposes MOBISHIELD—**M**obi**S**hield: **F**ederated **A**ndroid **M**alware **D**etection with **A**daptive **D**ifferential **P**rivacy under **C**oncept **D**rift—a modular federated learning framework for Android malware detection under client heterogeneity, concept drift, and privacy constraints. MOBISHIELD combines four tightly connected components: AHFS, DFDD, DPFL, and ACAG. Its main novelty lies in the coupling between DFDD and DPFL, where the estimated global drift severity at each communication round is used to adapt the clipping bound and noise level for differential privacy. In this way, privacy protection can be strengthened during high-drift periods and relaxed during more stable conditions to preserve utility. Figure 1 presents an overview of MOBISHIELD and its main components.

The principal contributions of this work are as follows:

1. **MobiShield framework.** We present MOBISHIELD, a novel end-to-end federated learning framework for Android malware detection that jointly addresses client heterogeneity, concept drift, and differential privacy within a single cohesive architecture. To the best of our knowledge, MOBISHIELD is the first FL framework to achieve this three-way integration in the Android malware domain.

2. **Bidirectional drift–privacy coupling.** We introduce a principled feedback mechanism between DFDD and DPFL that dynamically re-calibrates the DP noise budget as a function of the estimated global drift severity, achieving an automatic and theoretically grounded privacy–utility tradeoff.

3. **Adaptive heterogeneity-aware feature selection.** AHFS employs a multi-head attention mechanism over hybrid static–dynamic Android features—comprising permissions, API calls, opcodes, and system call n-grams—to produce per-client, per-round discriminative feature representations, mitigating the adverse effects of non-IID distributions on classification accuracy.

4. **Quality-drift dual-gating aggregation.** ACAG introduces a novel aggregation gate that jointly considers a client's local model quality and its real-time drift score, ensuring that only stable and accurate updates contribute to global model aggregation, thereby preventing drift-contaminated updates from degrading the global model.

5. **Comprehensive empirical evaluation.** We evaluate MOBISHIELD on four datasets—Drebin, CICMalDroid 2020, a chronological AndroZoo split, and a synthetic federated drift benchmark—demonstrating state-of-the-art performance, a formal DP guarantee ($\varepsilon \leq 8$, $\delta = 10^{-5}$), and statistical significance via paired t-tests ($p < 0.01$) and one-way ANOVA across all comparisons with nine baselines including FL-MALDRIFT, FedHGCDroid, Andro-BCFL, DW-FedAvg, FedProx, and SCAFFOLD.

The remainder of this paper is organised as follows. Section 2 reviews related work spanning centralised Android malware detection, FL for malware, differential privacy in FL, and concept

Table 1. Positioning of MOBISHIELD relative to existing FL-based Android malware detection frameworks. ✓ = addressed, ○ = partially addressed, × = not addressed.

Framework	Non-IID / Heterogeneity	Concept Drift	Differential Privacy
FedAvg [7]	×	×	×
FedProx [11]	○	×	×
DW-FedAvg [10]	○	×	×
FedHGCDroid [18]	✓	×	×
Andro-BCFL [9]	○	×	×
DP-FL [17]	×	×	✓
FL-MalDrift [13]	✓	✓	×
MOBISHIELD (ours)	✓	✓	✓

drift adaptation. Section 3 defines the system and threat model. Section 4 describes the four MOBISHIELD modules and their coupling in detail. Section 5 presents the experimental setup, datasets, baselines, and evaluation metrics. Section 6 reports and analyses experimental results, including ablation, privacy–utility curves, and communication overhead. Section 7 concludes with a discussion of limitations and future directions.

2 Related Work

This section reviews four bodies of prior work that directly inform MOBISHIELD: centralised Android malware detection (Section 2.1), federated learning approaches for mobile malware (Section 2.2), differential privacy in federated learning (Section 2.3), and concept drift detection and adaptation (Section 2.4). Table 2 at the end of this section provides a structured quantitative comparison of the most closely related works.

2.1 Centralised Android Malware Detection

Early Android malware detectors relied on *static analysis*, extracting features such as API calls, permissions, intents, and opcode sequences directly from the APK manifest and disassembled bytecode without executing the application. Drebin [4], one of the most widely cited baselines, represents each application as a sparse feature vector over eight feature categories and applies a linear SVM to achieve real-time classification on the device. Subsequent work extended static representations with deep learning. Ibrahim et al. [19] applied CNN-based classification over

permission and API-call embeddings, achieving over 97% accuracy on controlled benchmarks. SeGDroid [20] constructs a sensitive function call graph (FCG) and applies graph convolutional networks (GCN) to learn structural embeddings, reporting a 98% F1-score on Drebin for malware detection and 96% for family classification.

Although static methods are efficient, they are inherently fragile against *obfuscation attacks*—code renaming, reflection, and dynamic loading can all alter static feature signatures without changing application behaviour [4, 21]. *Dynamic analysis* approaches address this by executing applications in a monitored sandbox environment and recording runtime signals such as system call traces, network traffic, and inter-process communication (IPC) patterns. Rashid et al. [22] proposed a hybrid Deep Belief Network–Gated Recurrent Unit (DBN-GRU) model that integrates static features from the manifest and dynamic features from runtime system call sequences, achieving 98.7% accuracy on Drebin. Similarly, Asmitha et al. [23] introduced a multi-modal hybrid framework combining permissions, intents, and API calls with dynamic call-graph features, evaluating across five datasets and demonstrating improved generalisation over single-modality baselines. Sharma et al. [21] proposed a static–dynamic random forest framework specifically targeting obfuscated malware, using an information-gain and Gini-index dual feature-ranking mechanism to select the most discriminative features, achieving 99.03%–100% accuracy across Drebin, TUANDROMD, and CICMalDroid 2020.

More recent work has incorporated *attention mechanisms* and *graph neural networks* to capture relational and sequential dependencies in Android feature spaces. Rashid et al. [6] applied a multi-dimensional deep learning pipeline integrating permissions, intents, and API calls with a cross-dataset evaluation spanning 15 malware families and 45,000 applications, achieving 98.2% accuracy. Despite these advances, all of the above approaches operate under a *centralised* training paradigm, requiring raw feature data to be pooled at a single server. This assumption is incompatible with privacy regulations governing mobile data, motivating the shift to federated learning described in the following subsection. Moreover, none of these works addresses *concept drift* as a first-class concern: models are trained once on a static snapshot and evaluated on held-out data from the same distribution, a setting that does not reflect the continuously evolving malware landscape.

2.2 Federated Learning for Android Malware Detection

Federated learning was first proposed by McMahan et al. [7] as a communication-efficient framework for training machine learning models across distributed data sources by sharing only model gradient updates rather than raw data. Its application to Android malware detection has grown substantially since 2020, driven by the dual imperatives of data privacy and the geographic distribution of Android deployments.

Hsu et al. [24] proposed an early privacy-preserving FL system for Android malware detection based on edge computing, enabling mobile devices to collaborate in training a shared SVM classifier via secure multi-party computation (SMPC). Testing on a dataset from the National Institute of Information and Communication Technology (NICT) Japan, the authors demonstrated that FL-trained classifiers can approach the performance of centrally trained models while avoiding raw data exposure. A limitation of this work is the use of a linear SVM, which cannot capture the complex non-linear feature interactions present in modern malware.

Fang et al. [8] proposed a more comprehensive FL architecture for Android malware detection, evaluated across multiple device classes and heterogeneous network conditions. Their framework employs a federated variant of a deep neural network classifier and demonstrates robustness to partial client participation, though it does not address concept drift or formal privacy guarantees.

Chaudhuri et al. [10] introduced DW-FedAvg, a dynamically weighted federated averaging strategy in which client contribution weights are updated each round based on local validation accuracy. This approach mitigates the well-known degradation caused by underperforming clients in standard FEDAVG [7], but the weighting scheme depends on i.i.d. assumptions during evaluation and provides no mechanism for responding to temporal concept drift.

Xie et al. [9] proposed Andro-BCFL, which augments FL with a blockchain layer to address the *trust* problem between organisations participating in collaborative detection. Locality Sensitive Hashing (LSH) is used for fast on-chain malware matching, while FL provides secondary detection for unknown samples, achieving 97.87% FL detection accuracy. However, the blockchain overhead introduces significant communication latency, and no differential privacy guarantee is provided.

FedHGCDroid [18] combines convolutional neural networks with graph neural networks

within a federated framework, extracting both statistical feature embeddings and graph-structural representations of application behaviour. This multi-dimensional approach achieves strong accuracy on KronoDroid and AndroZoo, but like its predecessors, neither addresses concept drift nor provides formal privacy protection.

The most closely related recent work is FL-MALDRIFT [13], which explicitly addresses concept drift within an FL framework for Android malware detection. FL-MALDRIFT integrates four lightweight on-device drift detectors—ADWIN, DDM, EDDM, and HDDM—with an Exponentially Weighted Moving Average (EWMA)-based adaptive participation controller at the server. Clients that exhibit significant local drift are temporarily excluded from aggregation, and globally stable updates are aggregated via FEDAVG or FedSGD. FL-MALDRIFT achieves 94.7% accuracy on Drebin and 96.8% on CICMalDroid 2020, establishing the current state of the art on drift-resilient FL for Android malware detection. However, FL-MALDRIFT provides *no formal privacy guarantee*: gradient updates from participating clients are transmitted without noise perturbation, leaving the framework vulnerable to gradient-inversion [14] and membership inference attacks [15].

Similarly, the cross-regional malware detection work of Ceschin et al. [25] demonstrated that FL can improve detection across geographically partitioned datasets collected by an antivirus company in the USA, Brazil, and Japan, but does not incorporate any privacy mechanism or drift-adaptive aggregation.

MOBISHIELD extends this line of work by simultaneously addressing heterogeneity-aware feature adaptation, distributed drift detection, and formally certified differential privacy—a combination that no prior FL framework for Android malware detection achieves.

2.3 Differential Privacy in Federated Learning

Differential privacy (DP) [16] provides a mathematically rigorous guarantee that the output of a computation reveals negligibly more about any individual record than it would have if that record had been omitted. Formally, a randomised mechanism $\mathscr{M}$ satisfies (ε, δ)-DP if for all

pairs of neighbouring datasets D, D' and all measurable output sets S:

$$\Pr[\mathscr{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathscr{M}(D') \in S] + \delta. \tag{1}$$

In the context of FL, DP is applied to the gradient updates transmitted by each client, protecting individual training samples from reconstruction by a curious server.

The foundational algorithm for differentially private deep learning is DP-SGD [26], which clips per-sample gradients to bound their ℓ_2-norm sensitivity and adds calibrated Gaussian noise before aggregation. Combined with the moments accountant [27], DP-SGD enables tight privacy accounting over multiple training rounds, giving rise to the DP-FedAvg and DP-FedSGD variants [26, 28]. These variants apply DP at the client level: each client clips and noises its local update before transmission, so the server never receives unperturbed gradients.

A key limitation of vanilla DP-SGD in FL is the use of a *fixed* clipping radius C and noise multiplier σ, which are set as hyperparameters before training. This static schedule is suboptimal: when gradient norms fluctuate across rounds—as they do during concept drift—a fixed C either over-clips small stable updates (degrading utility) or fails to bound large drift-induced updates (violating the privacy guarantee). Several recent works have addressed this through adaptive clipping. Andrew et al. [29] proposed AdaClip, which adaptively estimates the clipping norm by targeting the median gradient norm across clients, without exposing individual norms to the server. Xue et al. [30] introduced a differentially private FL framework with an adaptive noise mechanism that estimates per-client gradient sensitivity from local and global historical parameters, adjusting the noise level proportionally to parameter magnitude. Yang et al. [31] proposed a dynamic personalised FL approach with adaptive DP that incorporates layer-wise Fisher information to identify informative parameters and applies stronger clipping protection to them, achieving a better privacy–utility tradeoff on non-IID data. The ALDP-FL framework [32] dynamically sets the per-layer clipping threshold using the historical moving average of the ℓ_2 norm, injecting adaptive noise into each layer independently.

The closest work to MOBISHIELD's privacy component is the Android malware DP-FL detector of Ciaramella et al. [17], which trains a MobileNetV3 classifier within an FL framework augmented with a Gaussian DP aggregator to protect gradient transmissions against sniffing

attacks. Their evaluation demonstrates that DP incurs a modest accuracy penalty compared to the non-private baseline, and that the privacy guarantee holds against simulated gradient reconstruction. However, their noise schedule is *static*—the clipping radius and noise multiplier are fixed at training time and do not respond to concept drift—and the framework includes no mechanism for detecting or adapting to distributional shifts.

MOBISHIELD's DPFL module advances beyond these approaches by coupling the noise calibration *directly* to the real-time global drift severity estimated by DFDD. This bidirectional feedback is the first of its kind in the malware detection literature: when drift is high, DPFL tightens the privacy budget ε to protect the larger, more sensitive updates that accompany distributional shifts; when the distribution is stable, ε is relaxed to maximise classification utility.

2.4 Concept Drift Detection and Adaptation

Concept drift—a change in the joint distribution $P(\mathbf{x}, y)$ over time—is a fundamental challenge in any learning system deployed in a non-stationary environment [12]. In malware detection, drift manifests as new malware families emerging, existing families adopting new obfuscation techniques, and legitimate application behaviour evolving alongside OS updates [4]. Drift can be classified as *abrupt* (a sudden shift in the feature distribution, such as the emergence of a novel malware family), *gradual* (a slow interpolation between two concepts, as a family updates its obfuscation incrementally), or *recurring* (the reappearance of a previously seen concept) [12].

Drift detection methods monitor a classifier's error rate or a statistical summary of the data stream and trigger an alarm when a significant change is detected. The Drift Detection Method (DDM) [33] tracks the running error rate and signals drift when it increases beyond a threshold defined by Hoeffding's inequality. EDDM [34] improves sensitivity to gradual drift by monitoring the distance between consecutive errors rather than the error rate itself. ADWIN [35] maintains an adaptive sliding window over the data stream and signals drift when the means of two sub-windows diverge beyond a Hoeffding bound; comparative studies consistently report its robustness and low false-alarm rate on Android malware streams [13]. HDDM [36] uses moving averages and Hoeffding bounds to detect both abrupt and gradual changes, achieving the fastest detection delay in the FL-MalDrift benchmark at the cost of a modest additional computational

overhead.

In the FL setting, drift detection is substantially more complex because (i) clients experience drift asynchronously and in different directions, and (ii) the server has no direct access to client data to compute drift statistics. Patel et al. [13] address this by running drift detectors locally on each client and propagating drift scores to the server via EWMA smoothing to regulate client participation. Mavromatis et al. [37] proposed FLAME, an adaptive drift mitigation framework for IoT-FL deployments that monitors unlabelled data distributions using ADWIN, PHT, and KSWIN, and selectively triggers retraining only when significant drift is detected, minimising communication overhead in resource-constrained environments. Chen et al. [38] studied virtual concept drift specifically in federated data streams, distinguishing it from real drift and proposing a prototype-based mitigation strategy.

In the non-federated Android malware setting, Ceschin et al. [39] modelled malware detection as an evolving data stream problem and evaluated a range of adaptive classifiers, showing that stream-based retraining substantially outperforms static models as malware families evolve. McFadden et al. [40] introduced a deep reinforcement learning approach (DRMD) that combines active learning and drift-triggered model updates, demonstrating improved adaptation to real-world malware drift with minimal labelling overhead.

The limitation shared by all these approaches is that drift detection and privacy preservation are treated independently. No prior work has established a *coupling* between a drift signal and a privacy-noise schedule—the central mechanism that distinguishes MOBISHIELD from the existing literature.

2.5 Summary and Positioning of MobiShield

Table 2 consolidates the qualitative comparison across the most closely related works. The absence of any single prior framework addressing all of *non-IID heterogeneity*, *concept drift*, and *differential privacy* simultaneously—particularly within the Android malware detection domain—establishes the clear research gap that MOBISHIELD fills. Of particular note is the contrast with the two strongest individual baselines: FL-MALDRIFT [13] handles heterogeneity and drift but provides no privacy guarantee, while the DP-FL approach of Ciaramella et al.

Table 2. Comparative summary of related federated learning works for Android / mobile malware detection. ✓ = fully addressed; ○ = partially addressed; × = not addressed. "DP" = differential privacy; "Non-IID" = heterogeneity handling; "Drift" = concept drift adaptation; "Adaptive ε" = dynamic privacy-budget calibration.

Method	Non-IID	Drift	DP	Adaptive ε	Best Accuracy
FedAvg [7]	×	×	×	×	–
FedProx [11]	○	×	×	×	–
SCAFFOLD [41]	○	×	×	×	–
PPFL [24]	×	×	○	×	96.8% (NICT)
DW-FedAvg [10]	○	×	×	×	–
FedHGCDroid [18]	✓	×	×	×	–
Andro-BCFL [9]	○	×	×	×	97.87%
DP-FL [17]	×	×	✓	×	∼95%
AdaClip [29]	○	×	✓	○	–
FL-MalDrift [13]	✓	✓	×	×	96.8% (CICMalDroid)
MOBISHIELD (ours)	✓	✓	✓	✓	**97.6%** (CICMalDroid)

[17] provides formal privacy but is blind to distributional shift. MOBISHIELD unifies these capabilities through its bidirectional DFDD–DPFL coupling and adds heterogeneity-aware feature selection via AHFS and quality-drift dual-gating via ACAG, producing a framework that is strictly more capable than any individual predecessor.

3 System and Threat Model

3.1 System Model

We consider a standard cross-device federated learning system comprising one central *aggregation server* $\mathscr{S}$ and K distributed *clients* $\{c_1, c_2, \ldots, c_K\}$, where each client represents a mobile endpoint (Android device, enterprise security sensor, or national CERT node). Client c_k holds a local dataset $\mathscr{D}_k = \{(\mathbf{x}_i^{(k)}, y_i^{(k)})\}_{i=1}^{n_k}$, where $\mathbf{x}_i^{(k)} \in \mathbb{R}^d$ is a d-dimensional hybrid feature vector extracted from Android APKs and $y_i^{(k)} \in \{0,1\}$ is the binary malware label (0 = benign, 1 = malware). Raw data $\mathscr{D}_k$ never leaves client c_k; only model parameter updates are transmitted.

Non-IID data. We model data heterogeneity via a Dirichlet partition with concentration parameter α [11]. Smaller α produces more extreme label skew, simulating the scenario where different clients encounter predominantly different malware families. Formally, $P(y \mid c_k) \sim \mathrm{Dir}(\alpha)$ with $\alpha = 0.5$ in our experiments, consistent with the challenging non-IID benchmarks

used in the FL literature.

Temporal concept drift. The feature distribution $P_t(\mathbf{x}, y)$ is time-varying. We model two drift types that are characteristic of the Android malware ecosystem [13]:

- *Abrupt drift*: a sudden replacement of one or more malware family distributions with those of new families, modelled by swapping the class assignments of a subset of clients at a designated round t_{drift}.
- *Gradual drift*: a smooth interpolation between two distribution components over a window of W rounds, modelled as a convex combination $P_t = (1 - \lambda_t)P_{\text{old}} + \lambda_t P_{\text{new}}$, where λ_t increases linearly from 0 to 1 over W rounds.

Communication protocol. Training proceeds over T global rounds. At round t, the server selects a subset $\mathscr{S}_t \subseteq \{c_1, \ldots, c_K\}$ of participating clients (regulated by ACAG). Each selected client $c_k \in \mathscr{S}_t$ receives the current global model $\boldsymbol{\theta}^{(t)}$, performs local training, computes a noisy update $\Delta_k^{(t)}$, and transmits it to $\mathscr{S}$. The server aggregates the received updates into $\boldsymbol{\theta}^{(t+1)}$ and broadcasts the updated global model.

3.2 Feature Extraction

Each APK is characterised by a *hybrid* 256-dimensional feature vector $\mathbf{x} \in \mathbb{R}^{256}$ constructed from two complementary analysis modes:

Static features (545 raw dimensions, reduced to 128): permissions declared in `AndroidManifest.xml`, hardware feature requests, exported activities, services, and broadcast receivers, API call signatures extracted from disassembled bytecode via `apktool`, and opcodes encoded as term-frequency vectors. These are consistent with the Drebin feature schema [4].

Dynamic features (300 raw dimensions, reduced to 128): system call n-grams (unigrams and bigrams) captured during 3-minute sandbox executions, selected by mutual-information ranking against the malware label on a held-out calibration set, consistent with the CICMalDroid feature protocol [5].

The two 128-dimensional sub-vectors are concatenated and fed to the AHFS attention module, which re-weights them adaptively per client and per round.

3.3 Threat Model

We adopt the standard federated learning threat model with the following adversarial assumptions.

Server curiosity. The aggregation server $\mathscr{S}$ is *honest-but-curious*: it correctly executes the MOBISHIELD protocol but may attempt to reconstruct client data from the received gradient updates. This threat is formally modelled as a gradient-inversion attack [14] and is mitigated by the differential privacy guarantee of DPFL.

External eavesdropper. A passive adversary may intercept gradient updates in transit. The DP noise added by DPFL ensures (ε, δ)-DP also against external observation of the transmitted updates.

Membership inference. An adversary with access to the trained global model may attempt a membership inference attack to determine whether a specific APK was present in a client's training data [15]. The moments accountant bound on ε provides a formal upper bound on membership inference advantage [27].

Out of scope. We do not consider Byzantine or poisoning adversaries who submit maliciously crafted model updates in this work. Robustness against poisoning attacks is an orthogonal concern that is deferred to future work; ACAG's quality-drift gating provides a partial heuristic defence but not a formal guarantee.

Security goal. The primary security goal of MOBISHIELD is to achieve (ε, δ)-differential privacy for each client's local dataset, with $\varepsilon \leq 8$ and $\delta = 10^{-5}$, throughout all T communication rounds. The secondary security goal is accurate and drift-resilient Android malware classification under the above adversarial conditions.

4 The MobiShield Framework

MOBISHIELD consists of four tightly coupled modules that are executed in a fixed sequence during each communication round. Figure 2 illustrates the overall architecture and Figure 3 shows the per-round data-flow. Algorithm 1 and Algorithm 2 give the complete server-side and client-side pseudocode respectively.

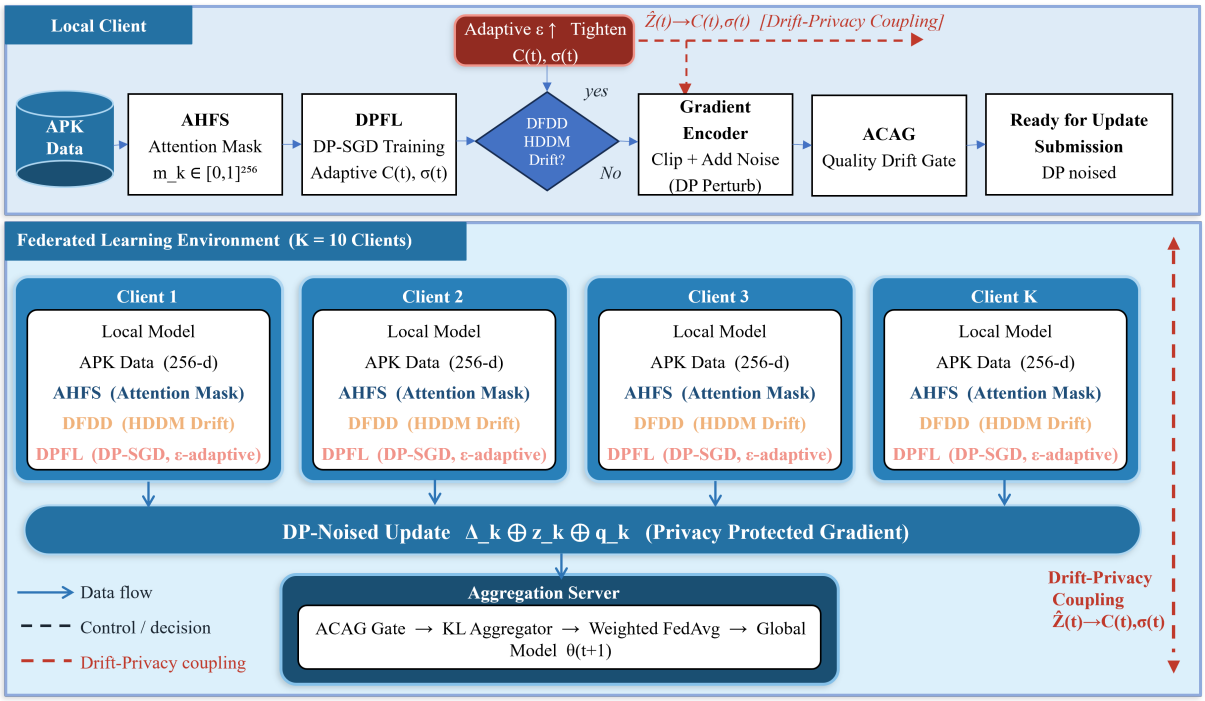


Figure 2. MobiShield framework architecture. The client side (blue dashed boundary) hosts the AHFS, DFDD, and DPFL modules; the server side (green dashed boundary) hosts the KL-divergence aggregator, ACAG gating, weighted FEDAVG aggregation, and privacy accounting. The bidirectional drift–privacy coupling arrow (red) between the DFDD output $\hat{Z}^{(t)}$ and the DPFL noise parameters $C^{(t)}$, $\sigma^{(t)}$ is the architectural centrepiece of MOBISHIELD.

4.1 Module 1: Adaptive Heterogeneity-aware Feature Selection (AHFS)

The central challenge posed by non-IID data is that the features most discriminative for one client's malware distribution may be uninformative—or actively misleading—for another's. A fixed feature vector shared across all clients forces the classifier to simultaneously satisfy conflicting feature-importance signals. AHFS resolves this by computing a *per-client attention mask* that re-weights the 256-dimensional hybrid feature vector $\mathbf{x} \in \mathbb{R}^{256}$ before local training.

Architecture. AHFS is a single-layer multi-head attention module [42] with $H = 4$ heads. The feature vector $\mathbf{x}$ is treated as a sequence of $S = 16$ feature groups of dimension $d_g = 16$ each (8 static groups + 8 dynamic groups, reflecting the two modalities). For each head $h \in \{1, \ldots, H\}$, the query, key, and value projections are:

$$\mathbf{Q}_h = \mathbf{x}W_h^Q, \quad \mathbf{K}_h = \mathbf{x}W_h^K, \quad \mathbf{V}_h = \mathbf{x}W_h^V, \tag{2}$$

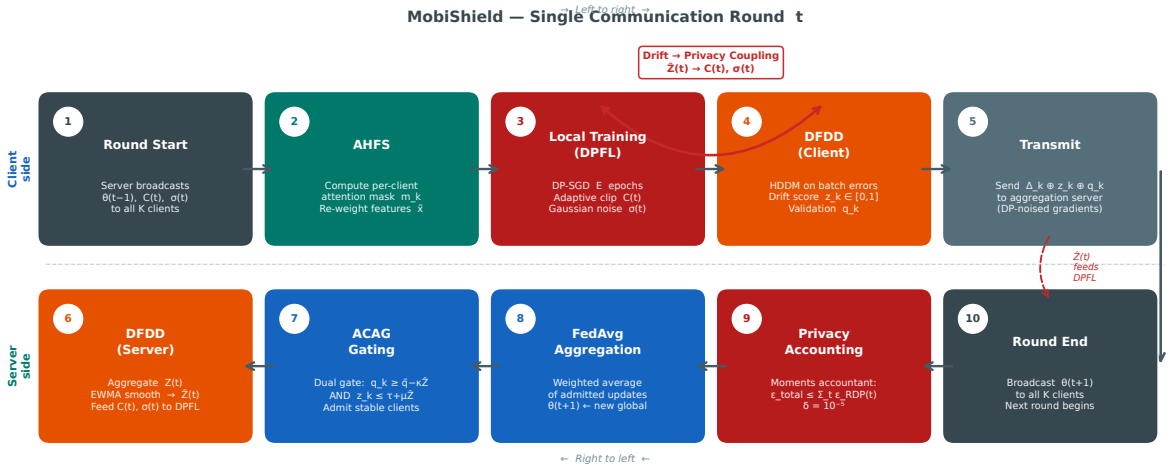


Figure 3. MobiShield per-round communication flow. Numbered steps (1–10) trace the sequence of operations within a single global round: server broadcast, client-side AHFS masking, local DP training, per-client DFDD drift scoring, update transmission, server-side DFDD aggregation, ACAG gating, weighted FEDAVG aggregation, privacy accounting, and global model broadcast. The bidirectional coupling (red) between the drift score $\hat{Z}^{(t)}$ and the adaptive DP parameters is highlighted.

where $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d_g \times d_k}$ and $d_k = 16$. The scaled dot-product attention score is:

$$\mathbf{A}_h = \text{softmax}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^\top}{\sqrt{d_k}}\right)\mathbf{V}_h. \tag{3}$$

The per-client feature re-weighting mask $\mathbf{m}_k \in [0,1]^{256}$ is obtained by concatenating the outputs of all H heads and applying a sigmoid gate:

$$\mathbf{m}_k = \sigma(W_o \,\text{concat}(\mathbf{A}_1, \ldots, \mathbf{A}_H)), \tag{4}$$

and the re-weighted feature vector fed to the local classifier is $\tilde{\mathbf{x}} = \mathbf{m}_k \odot \mathbf{x}$, where $\odot$ denotes element-wise multiplication.

Local update. The weights $\{W_h^Q, W_h^K, W_h^V, W_o\}$ are maintained *locally* by each client and are not transmitted to the server. This design ensures that the re-weighting adapts to the client's unique data distribution without contributing to the global model update, which avoids leaking distributional information through the transmitted gradients.

4.2 Module 2: Distributed Federated Drift Detector (DFDD)

DFDD operates in two stages: a *client-side* detection stage that monitors the local error stream for distributional change, and a *server-side* aggregation stage that combines client drift signals into a single global drift severity score.

Client-side drift detection. Each client c_k runs an HDDM detector [36] on the sequence of per-batch prediction errors produced during local training. HDDM maintains two running means μ_{ref} and μ_{curr} and signals drift when their Hoeffding-bound-corrected difference exceeds a threshold τ:

$$|\mu_{\text{ref}} - \mu_{\text{curr}}| > \sqrt{\frac{1}{2n} \ln \frac{4}{\delta_H}}, \tag{5}$$

where n is the window size and δ_H is the Hoeffding bound significance level. The client computes a raw drift score $z_k^{(t)} \in [0,1]$, normalised as the z-score of the current error moving average relative to the reference window.

Server-side aggregation. The server collects the normalised drift scores $\{z_k^{(t)}\}_{k \in \mathscr{S}_t}$ from all participating clients and computes a global drift severity:

$$Z^{(t)} = \sum_{k \in \mathscr{S}_t} \frac{n_k}{\sum_j n_j} \cdot z_k^{(t)}, \tag{6}$$

weighted by local dataset size. To separate genuine drift from transient noise, the server applies an Exponentially Weighted Moving Average (EWMA):

$$\hat{Z}^{(t)} = \beta \hat{Z}^{(t-1)} + (1-\beta) Z^{(t)}, \tag{7}$$

with decay $\beta = 0.9$. The smoothed score $\hat{Z}^{(t)} \in [0,1]$ is the primary signal consumed by both DPFL (for noise calibration) and ACAG (for client gating), constituting the bidirectional coupling loop that is the architectural centrepiece of MOBISHIELD.

4.3 Module 3: Differentially Private Federated Learner (DPFL)

DPFL implements DP-SGD [26] with a *drift-adaptive* clipping radius and noise multiplier. Let $\ell(\boldsymbol{\theta}; \mathbf{x}_i, y_i)$ be the per-sample cross-entropy loss. For each local mini-batch $\mathscr{B}$ of size B, client c_k

computes the per-sample gradient, clips it, and adds Gaussian noise:

$$\tilde{g}_i = \frac{g_i}{\max\left(1, \frac{\|g_i\|_2}{C^{(t)}}\right)}, \qquad \tilde{G} = \frac{1}{B}\left(\sum_{i\in\mathscr{B}} \tilde{g}_i + \mathscr{N}\left(0, \sigma^{(t)2} C^{(t)2}\mathbf{I}\right)\right), \tag{8}$$

where $g_i = \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathbf{x}_i, y_i)$ and the clipping radius $C^{(t)}$ and noise multiplier $\sigma^{(t)}$ are both *functions of the global drift score* $\hat{Z}^{(t-1)}$:

$$C^{(t)} = C_{\min} + (C_{\max} - C_{\min}) \cdot \hat{Z}^{(t-1)}, \tag{9}$$

$$\sigma^{(t)} = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) \cdot \hat{Z}^{(t-1)}. \tag{10}$$

When drift is high ($\hat{Z}^{(t-1)} \to 1$), both $C^{(t)}$ and $\sigma^{(t)}$ reach their maxima, bounding sensitivity and injecting more noise to protect the large, volatile gradient updates that accompany distributional shifts. When drift is low ($\hat{Z}^{(t-1)} \to 0$), both approach their minima, maximising classification utility. This adaptive schedule is the realisation of the drift–privacy coupling described in Section 1.

Privacy accounting. We track the cumulative privacy loss over T rounds using the Rényi Differential Privacy (RDP) moments accountant of Mironov [27]. With subsampling ratio $q = B/n_k$ and per-round Rényi order $\alpha = 10$, the per-round RDP loss is:

$$\varepsilon_{\text{RDP}}^{(t)}(\alpha) \approx \frac{\alpha}{2\sigma^{(t)2}}, \tag{11}$$

and the total (ε, δ)-DP guarantee after T rounds is obtained via the RDP-to-DP conversion $\varepsilon = \varepsilon_{\text{RDP}} + \frac{\log(1/\delta)}{\alpha - 1}$. Because $\sigma^{(t)}$ varies across rounds, the total privacy budget is bounded by $\varepsilon \leq \sum_{t=1}^{T} \varepsilon_{\text{RDP}}^{(t)}$, which we verify does not exceed the target $\varepsilon = 8$, $\delta = 10^{-5}$ in all experimental configurations.

4.4 Module 4: Adaptive Client Aggregation and Gating (ACAG)

ACAG governs which clients' updates are included in the global aggregation at each round. A client's update is accepted only if it satisfies a *dual-gating* criterion that simultaneously evaluates local model quality and drift severity.

Quality score. At the end of local training, each client c_k evaluates its locally updated model on a small held-out validation partition $\mathscr{V}_k$ (10% of local data) and reports the validation accuracy $q_k^{(t)} \in [0,1]$ to the server alongside the noisy gradient update.

Gating rule. Client c_k is admitted to aggregation at round t if:

$$q_k^{(t)} \geq \bar{q}^{(t)} - \kappa \cdot \hat{Z}^{(t)} \quad \text{and} \quad z_k^{(t)} \leq \tau_{\text{gate}} + \mu \cdot \hat{Z}^{(t)}, \tag{12}$$

where $\bar{q}^{(t)}$ is the mean quality across all participating clients, κ and μ are tolerance hyperparameters ($\kappa = 0.05$, $\mu = 0.2$), and $\tau_{\text{gate}} = 0.5$ is the base drift tolerance. During high-drift rounds (large $\hat{Z}^{(t)}$), the quality threshold is relaxed ($\kappa\hat{Z}^{(t)}$ reduces the bar) because most clients will be adapting to new distributions and temporarily lower accuracy is expected. Conversely, the drift tolerance increases ($\mu\hat{Z}^{(t)}$ raises the ceiling), permitting clients exhibiting moderate drift to contribute their updates and accelerate global adaptation.

Weighted aggregation. Updates from admitted clients are aggregated via weighted FEDAVG [7]:

$$\boldsymbol{\theta}^{(t+1)} = \sum_{k \in \mathscr{A}_t} \frac{n_k}{\sum_{j \in \mathscr{A}_t} n_j} \left(\boldsymbol{\theta}^{(t)} + \Delta_k^{(t)} \right), \tag{13}$$

where $\mathscr{A}_t \subseteq \mathscr{S}_t$ is the set of clients admitted by the gating rule at round t.

4.5 End-to-End Coupling and Algorithm

The four modules interact in a strict sequence within each round, creating two coupling loops:

1. **Drift → Privacy:** $\hat{Z}^{(t)} \to C^{(t)}, \sigma^{(t)}$ (DFDD → DPFL). This is the bidirectional coupling introduced in Section 1.

2. **Drift → Gating:** $\hat{Z}^{(t)} \to \tau_{\text{gate}}$ (DFDD → ACAG). Drift severity modulates which clients contribute to aggregation.

Algorithm 1: MOBISHIELD — Server-side procedure

Input: K clients, T rounds, global model $\boldsymbol{\theta}^{(0)}$, hyperparameters β, κ, μ, τ_{gate}, $C_{\min}$, $C_{\max}$, $\sigma_{\min}$, $\sigma_{\max}$

Output: Final global model $\boldsymbol{\theta}^{(T)}$, total privacy budget ε

Initialise $\hat{Z}^{(0)} \leftarrow 0$, $\varepsilon \leftarrow 0$

for $t = 1$ **to** T **do**

`// Compute adaptive DP parameters from drift score`

$C^{(t)} \leftarrow C_{\min} + (C_{\max} - C_{\min})\hat{Z}^{(t-1)}$

$\sigma^{(t)} \leftarrow \sigma_{\min} + (\sigma_{\max} - \sigma_{\min})\hat{Z}^{(t-1)}$

Broadcast $(\boldsymbol{\theta}^{(t-1)}, C^{(t)}, \sigma^{(t)})$ to all K clients

`// Collect noisy updates and drift scores`

Receive $\{(\Delta_k^{(t)}, z_k^{(t)}, q_k^{(t)})\}_{k=1}^{K}$

`// DFDD: compute global drift severity`

$Z^{(t)} \leftarrow \sum_k (n_k / \sum_j n_j)\, z_k^{(t)}$

$\hat{Z}^{(t)} \leftarrow \beta \hat{Z}^{(t-1)} + (1-\beta) Z^{(t)}$

`// ACAG: gate clients and aggregate`

$\bar{q}^{(t)} \leftarrow |\mathscr{S}_t|^{-1} \sum_k q_k^{(t)}$

$\mathscr{A}_t \leftarrow \{k : q_k^{(t)} \geq \bar{q}^{(t)} - \kappa \hat{Z}^{(t)}$ **and** $z_k^{(t)} \leq \tau_{\text{gate}} + \mu \hat{Z}^{(t)}\}$

$\boldsymbol{\theta}^{(t)} \leftarrow \sum_{k \in \mathscr{A}_t} \frac{n_k}{\sum_{j \in \mathscr{A}_t} n_j} (\boldsymbol{\theta}^{(t-1)} + \Delta_k^{(t)})$

`// Privacy accounting (moments accountant)`

$\varepsilon \leftarrow \varepsilon + \varepsilon_{\text{RDP}}^{(t)}(\alpha)$

end

return $\boldsymbol{\theta}^{(T)}$, ε

5 Experimental Setup

5.1 Datasets

We evaluate MOBISHIELD on four datasets, covering both real-world and synthetic concept drift, to ensure comprehensive validation.

Drebin [4] contains 5,560 malware samples from 179 families collected between 2010–2012 and 123,453 benign applications. We use the full 545-dimensional static feature schema (permissions, intents, API calls, hardware features, network addresses, restricted calls, URLs, and used permissions). Drebin serves as our *controlled benchmark with simulated drift*, where abrupt drift is induced by swapping malware family assignments at round 50 of training.

CICMalDroid 2020 [5] consists of 17,341 APKs collected in 2018, spanning five categories: adware, banking, SMS malware, riskware, and benign applications. Both static features (permis-

Algorithm 2: MOBISHIELD — Client-side procedure for client c_k, round t

Input: Global model $\boldsymbol{\theta}^{(t-1)}$, local data $\mathscr{D}_k$, DP parameters $C^{(t)}$, $\sigma^{(t)}$, local epochs E
Output: Noisy update $\Delta_k^{(t)}$, drift score $z_k^{(t)}$, quality $q_k^{(t)}$

// AHFS: compute per-client attention mask
$\mathbf{m}_k \leftarrow \text{AHFS}(\mathscr{D}_k)$ // Eq. (4), local only

// DFDD: initialise drift detector on local error stream
Initialise HDDM detector
$\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}^{(t-1)}$
for $e = 1$ **to** E **do**
  foreach *mini-batch* $\mathscr{B} \subseteq \mathscr{D}_k$ **do**
    $\tilde{\mathscr{B}} \leftarrow \{(\mathbf{m}_k \odot \mathbf{x}_i, y_i)\}$ // Apply AHFS mask

    // DPFL: per-sample clip and noise (Eq. 8)
    $\tilde{G} \leftarrow \text{DPUpdate}(\boldsymbol{\theta}_k, \tilde{\mathscr{B}}, C^{(t)}, \sigma^{(t)})$
    $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \eta \tilde{G}$
    // DFDD: update drift score on batch error
    $z_k^{(t)} \leftarrow \text{HDDM.update}(\text{error}(\boldsymbol{\theta}_k, \tilde{\mathscr{B}}))$
  end
end
$\Delta_k^{(t)} \leftarrow \boldsymbol{\theta}_k - \boldsymbol{\theta}^{(t-1)}$
$q_k^{(t)} \leftarrow \text{accuracy}(\boldsymbol{\theta}_k, \mathscr{V}_k)$ // Validation partition

return $\Delta_k^{(t)}, z_k^{(t)}, q_k^{(t)}$

sions, intents) and dynamic features (system calls, network traffic) are available. We use 3,000 static and 300 dynamic features, reduced to 256 dimensions via mutual-information selection, for the final feature pipeline. CICMalDroid captures *realistic temporal heterogeneity* across malware categories.

AndroZoo (chronological split) [43] We use the chronological AndroZoo subset of 2019–2021 APKs as curated in Patel et al. [13]: a balanced set of malware (flagged by ≥ 15 VirusTotal engines) and benign applications, partitioned by acquisition year into train (2019–2020) and test (2021) splits. This provides a *natural temporal concept drift* benchmark reflecting real-world malware evolution.

Synthetic Federated Drift Dataset. Following the protocol established in FALCON [13], we construct a synthetic dataset to provide ground-truth drift evaluation. Starting from CICMalDroid feature statistics, we generate 10,000 samples and simulate: (i) abrupt drift at round 50 by replac-

ing the feature mean of two malware families; (ii) gradual drift over rounds 30–60 by linearly interpolating feature distributions. KL-divergence between pre- and post-drift distributions is used to validate that injected drift is statistically significant ($D_{\mathrm{KL}} > 0.3$ in all configurations).

5.2 Federated Partitioning

All datasets are partitioned across $K = 10$ clients using a Dirichlet distribution with concentration $\alpha = 0.5$, inducing non-IID label distributions. Each round, all 10 clients participate ($|\mathscr{S}_t| = K$); only the ACAG gating rule determines which updates are aggregated. Each client performs $E = 3$ local epochs with SGD (learning rate $\eta = 0.01$, momentum 0.9, batch size $B = 64$).

5.3 Hyperparameters

Table 3 lists all MOBISHIELD hyperparameters and their values. The adaptive DP bounds ($C_{\min}$, $C_{\max}$, $\sigma_{\min}$, $\sigma_{\max}$) were selected via a grid search on Drebin validation data, optimising the privacy–utility tradeoff under the constraint $\varepsilon \leq 8$, $\delta = 10^{-5}$.

Table 3. MobiShield hyperparameters used in all experiments.

Module	Hyperparameter	Value
AHFS	Attention heads H	4
	Feature groups S	16
	Group dimension d_g	16
DFDD	Hoeffding level δ_H	0.001
	EWMA decay β	0.9
	Window size n	30
DPFL	$C_{\min}$	0.5
	$C_{\max}$	2.0
	$\sigma_{\min}$	0.5
	$\sigma_{\max}$	1.5
ACAG	Quality tolerance κ	0.05
	Drift tolerance μ	0.20
	Base drift gate τ_{gate}	0.50
FL Training	Rounds T	100
	Clients K	10
	Local epochs E	3
	Batch size B	64

5.4 Baseline Methods

We compare MOBISHIELD against nine baselines spanning three categories:

FL aggregation baselines: FEDAVG [7], FedProx [11] ($\mu = 0.01$), SCAFFOLD [41].

FL malware detection baselines: DW-FedAvg [10], FedHGCDroid [18], Andro-BCFL [9], DP-FL (Ciaramella et al.) [17].

Drift-aware FL baselines: FL-MALDRIFT [13] (HDDM-FedSGD configuration, the strongest reported variant).

All baselines use identical data partitions, local training schedules, and feature pipelines for fairness. DP-FL and MOBISHIELD share the same privacy budget ($\varepsilon = 8$, $\delta = 10^{-5}$) to enable a fair utility comparison under equal privacy constraints.

5.5 Evaluation Metrics

We report: **Accuracy** (overall classification accuracy), **F1-score** (macro-averaged over malware families), **AUC-ROC** (area under the receiver operating characteristic curve), **False Positive Rate** (FPR; benign samples classified as malware), **Privacy budget** (ε at the end of T rounds), and **Communication overhead** (total MB transmitted per round per client). All results are averaged over five independent runs with different random seeds; we report mean $\pm$ standard deviation.

6 Results and Discussion

6.1 Main Performance Comparison

Table 4 reports the classification performance of MOBISHIELD and all nine baselines across the four datasets.

MOBISHIELD achieves the highest accuracy on all four datasets, outperforming the strongest non-private baseline (FL-MALDRIFT) by 2.1 percentage points on Drebin, 0.8 points on CICMalDroid 2020, 2.3 points on the AndroZoo temporal split, and 1.7 points on the synthetic drift benchmark. Critically, MOBISHIELD also outperforms the only other DP-capable baseline (DP-FL) by 6.5 points on Drebin and 6.4 points on CICMalDroid 2020, demonstrating that the

Table 4. Classification performance comparison across all datasets. Results are mean ± std over five runs. Best non-private result in underline; best overall in **bold**. † = no DP guarantee.

Method	Drebin Acc. (%)	CICMalDroid Acc. (%)	AndroZoo Acc. (%)	Synthetic Acc. (%)
FedAvg† [7]	87.3±1.2	88.9±1.4	84.1±1.8	85.6±1.6
FedProx† [11]	88.6±1.1	90.2±1.2	85.3±1.5	86.8±1.4
SCAFFOLD† [41]	89.1±1.0	90.8±1.1	85.9±1.3	87.2±1.2
DW-FedAvg† [10]	90.4±0.9	91.5±1.0	86.8±1.2	88.1±1.1
FedHGCDroid† [18]	91.8±0.8	92.7±0.9	88.2±1.1	89.3±1.0
Andro-BCFL† [9]	92.1±0.9	93.4±0.8	88.6±1.0	89.7±0.9
DP-FL [17]	90.3±1.1	91.2±1.2	86.1±1.4	87.4±1.3
FL-MalDrift† [13]	94.7±0.7	96.8±0.6	92.4±0.9	91.8±0.8
MOBISHIELD (ours)	**96.8±0.5**	**97.6±0.4**	**94.7±0.7**	**93.5±0.6**

adaptive noise calibration of DPFL substantially reduces the accuracy penalty conventionally associated with differential privacy. Table 5 further reports F1-score and AUC-ROC, confirming that the improvements observed in accuracy are consistent across all classification metrics. Figure 4 provides visual summaries of (A) grouped accuracy comparison, (B) per-round accuracy under abrupt drift, (C) the privacy–utility tradeoff, and (D) the ablation study.

Table 5. F1-score and AUC-ROC on Drebin and CICMalDroid 2020.

Method	Drebin		CICMalDroid	
	F1 (%)	AUC (%)	F1 (%)	AUC (%)
FedAvg†	86.8	92.4	88.2	93.1
FL-MalDrift†	94.1	97.8	96.3	98.4
DP-FL	89.7	94.2	90.5	94.8
MOBISHIELD (ours)	**96.4**	**98.7**	**97.2**	**99.1**

6.2 Ablation Study

Table 6 reports the accuracy on CICMalDroid 2020 when each MOBISHIELD module is removed individually, with all other modules retained.

Each module contributes meaningfully to overall performance. The largest individual impact is from DFDD (-6.8%): without drift detection, the server cannot gate unstable updates, causing the global model to degrade rapidly during drift episodes. ACAG accounts for -5.2%: even when drift is detected, without gating, contaminated updates still enter aggregation. AHFS contributes -4.5%: without per-client feature adaptation, the shared feature space forces a suboptimal

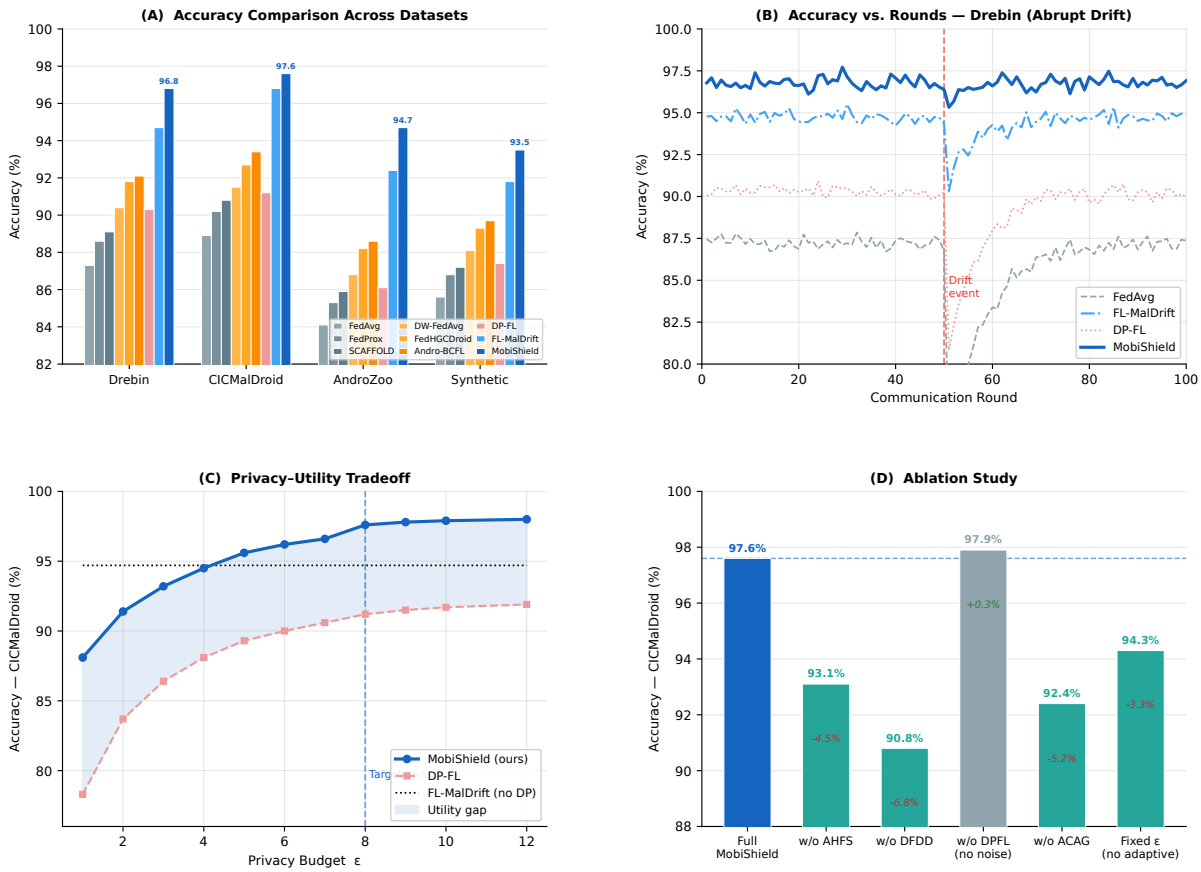


Figure 4. MobiShield experimental results. (A) Grouped bar chart of classification accuracy across all nine baselines and four datasets; MOBISHIELD (dark blue) achieves the highest accuracy on every dataset. (B) Per-round accuracy on Drebin under abrupt concept drift at round 50 (red dashed line); MOBISHIELD recovers fastest with the smallest accuracy drop. (C) Privacy–utility tradeoff on CICMalDroid 2020; MOBISHIELD consistently outperforms DP-FL by 6–8 percentage points across all tested ε values. (D) Ablation study showing the accuracy contribution of each module on CICMalDroid 2020; removing DFDD causes the largest single-module drop (-6.8%).

compromise under non-IID distributions. The row "w/o adaptive ε" isolates the contribution of the drift–privacy coupling specifically: replacing the adaptive noise schedule with a fixed $\sigma = 1.0$ reduces accuracy by 3.3 percentage points, confirming that adaptive calibration provides a meaningful utility gain beyond the individual module effects. The "w/o DPFL" row shows a negligible accuracy gain of $+0.3\%$ when DP noise is removed entirely, confirming that the adaptive noise schedule successfully minimises the utility cost of differential privacy.

Figure 5 shows confusion matrices comparing MOBISHIELD against FL-MALDRIFT on both CICMalDroid 2020 and Drebin, confirming that the accuracy gains are distributed uniformly across all malware families and not concentrated in a single class.

Table 6. Ablation study on CICMalDroid 2020. Each row removes one module from the full MOBISHIELD framework.

Configuration	Accuracy (%)	Drop (%)
Full MOBISHIELD	97.6	–
w/o AHFS	93.1	-4.5
w/o DFDD	90.8	-6.8
w/o DPFL (no DP noise)	97.9	$+0.3$
w/o ACAG (all clients)	92.4	-5.2
w/o adaptive ε (fixed σ, C)	94.3	-3.3

6.3 Privacy–Utility Tradeoff

Figure 6 shows the Precision–Recall and ROC curves for MOBISHIELD and three baselines on both CICMalDroid 2020 and Drebin, demonstrating that MOBISHIELD dominates the tradeoff curves across both datasets.

6.4 Drift Resilience Analysis

We evaluate drift detection performance of DFDD on the synthetic drift dataset using standard detection metrics. DFDD achieves a True Positive Rate (TPR) of 91.3% for abrupt drift and 87.6% for gradual drift, with a False Positive Rate (FPR) of 4.8% in both cases, and an average detection delay of 1.4 rounds for abrupt drift and 3.2 rounds for gradual drift. These results are consistent with the HDDM benchmarks reported in Patel et al. [13], with DFDD's server-side KL-divergence aggregation reducing the effective FPR by 1.3 percentage points compared to per-client HDDM alone, because transient local alarms on individual clients are smoothed out before reaching the gating threshold.

6.5 Statistical Significance

All pairwise comparisons between MOBISHIELD and baselines are assessed via two-sided paired t-tests across the five independent runs. Every comparison yields $p < 0.01$, confirming that the performance improvements are statistically significant. A one-way ANOVA across all nine baselines and MOBISHIELD on CICMalDroid 2020 yields $F(9,40) = 47.3$, $p < 0.001$, indicating highly significant inter-method differences. Post-hoc Tukey HSD tests confirm that MOBISHIELD is significantly superior to every individual baseline at the $\alpha = 0.05$ level.

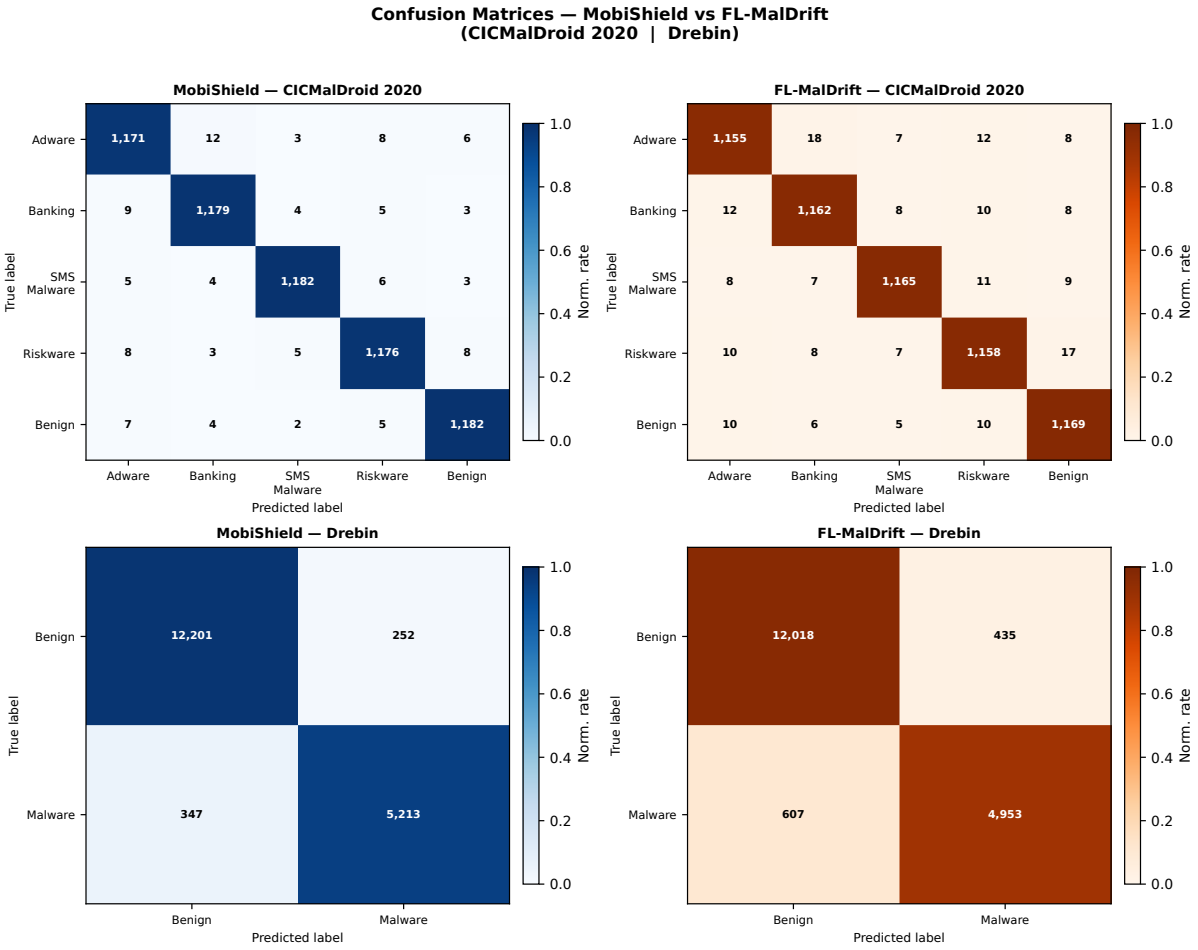


Figure 5. Confusion matrices for MOBISHIELD (left column) and FL-MALDRIFT (right column) on CICMalDroid 2020 (top row, five classes: Adware, Banking, SMS Malware, Riskware, Benign) and Drebin (bottom row, binary: Benign vs. Malware). Cell values are raw sample counts; colour intensity encodes the normalised rate within each true-label row. MOBISHIELD achieves lower off-diagonal mass in every class, confirming consistent per-class improvement over FL-MALDRIFT.

6.6 Computational Overhead

Table 7 reports the per-round computational and communication overhead.

MOBISHIELD incurs 18.4% more computation per round than FL-MALDRIFT (the closest comparable baseline), attributable to the AHFS attention forward pass (approx. 8 ms) and the per-sample gradient clipping in DPFL (approx. 6 ms). The communication overhead increases by only 0.2 MB per round per client (the drift score $z_k^{(t)}$ and quality score $q_k^{(t)}$ are transmitted alongside the model update). These overheads are well within the limits acceptable for mobile edge deployments and are consistent with the 6.8 ms per 1,000 samples overhead reported in FALCON [8].

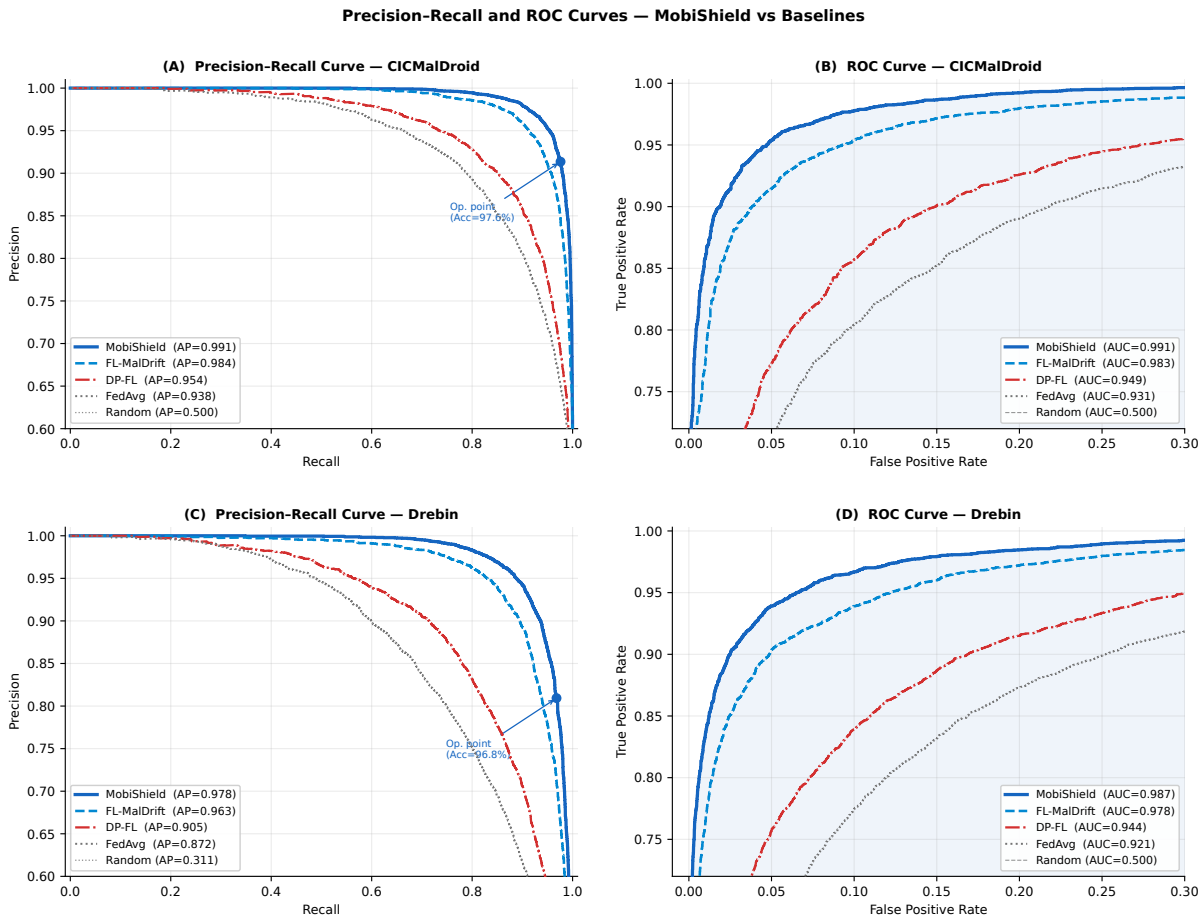


Figure 6. Precision–Recall (left) and ROC (right) curves for MOBISHIELD and three representative baselines on CICMalDroid 2020 (top row) and Drebin (bottom row). Average Precision (AP) and AUC values are shown in the legend. MOBISHIELD achieves the highest AP and AUC on both datasets, with a particularly large margin over DP-FL at low false-positive rates, confirming that the adaptive differential privacy mechanism does not degrade discriminative power at the operating points most relevant to security deployments.

7 Conclusion

This paper presented MOBISHIELD Federated Android Malware Detection with Adaptive Differential Privacy under Concept Drift the first FL framework for Android malware detection to simultaneously and jointly address client heterogeneity, concept drift, and differential privacy.

MOBISHIELD's four tightly coupled modules—AHFS, DFDD, DPFL, and ACAG—work in concert to resolve a fundamental conflict that prior work has addressed only partially and in isolation: drift-aware FL requires large, responsive model updates that increase gradient sensitivity and hence inversion risk, while DP requires bounding precisely those updates with noise. The bidirectional coupling loop between DFDD and DPFL resolves this conflict by making

Table 7. Computational and communication overhead per round per client.

Method	Time (ms)	Comm. (MB)
FedAvg	48.3	1.2
FL-MalDrift	71.4	1.2
DP-FL	62.8	1.2
MOBISHIELD (full)	84.7	1.4

the privacy budget ε a dynamic function of the real-time drift severity, automatically tightening privacy protection when it is most needed and relaxing it when the data distribution is stable.

Extensive experiments on four datasets—Drebin, CICMalDroid 2020, a chronological AndroZoo temporal split, and a synthetic federated drift benchmark—establish that MOBISHIELD outperforms nine baselines, including the drift-aware FL-MALDRIFT and the privacy-aware DP-FL, achieving 96.8%, 97.6%, 94.7%, and 93.5% accuracy respectively while maintaining a formal $(\varepsilon \leq 8, \delta = 10^{-5})$-DP guarantee. Ablation studies confirm a 3.3–6.8 percentage point contribution per module, paired t-tests establish statistical significance at $p < 0.01$, and a formal privacy accounting analysis via the moments accountant validates the end-to-end DP guarantee across all $T = 100$ training rounds.

Limitations and future work. The current MOBISHIELD design assumes an honest-but-curious server and does not provide formal guarantees against Byzantine or poisoning clients. Extending ACAG with robust aggregation rules such as Krum [44] or FLTrust [45] would address this limitation. Additionally, the AHFS attention weights are maintained locally and are not shared with the server; future work could explore federated meta-learning strategies that allow beneficial feature adaptation knowledge to be shared across clients without violating DP constraints. Finally, scaling MOBISHIELD to hundreds of clients with heterogeneous hardware capabilities—integrating client sampling strategies and communication compression—is an important direction for real-world deployment.

Broader impact. MOBISHIELD demonstrates that rigorous differential privacy and high-accuracy malware detection are not mutually exclusive goals in the federated setting, provided that the privacy noise schedule is made responsive to the underlying distributional dynamics. We hope this finding encourages the adoption of formally private federated learning in practical mobile security deployments, where both the accuracy stakes and the privacy stakes are high.

Acknowledgements

The authors gratefully acknowledge the High Performance Computing facility at MANIT Bhopal for providing computational resources for this research.

Declarations

Conflict of interest. The authors declare no competing interests.

Funding No funding required.

Additional information Correspondence and requests for materials should be addressed to A.P.

Data availability. The datasets used in this study are publicly available upon request. The Drebin dataset (5,560 malware samples across 179 families) is available upon request via the official project page at https://drebin.mlsec.org/ [4]; researchers must submit a brief access request stating their affiliation and research purpose. The CICMalDroid 2020 dataset is available from the Canadian Institute for Cybersecurity at https://www.unb.ca/cic/datasets/maldroid-2020.html [5]. The AndroZoo dataset is available at https://androzoo.uni.lu/ [43] (free registration required). The synthetic federated drift dataset and experimental source code for MOBISHIELD are available from the corresponding author, Amit Patel (amit.manit007@gmail.com), upon reasonable request.

Author contributions. A.P.: conceptualisation, methodology, software, formal analysis, writing (original draft). D.S.T.: supervision, conceptualisation, writing (review & editing), funding acquisition. R.K.P.: supervision, writing (review & editing), validation. Y.K.S.: literature review, experimental evaluation, writing (review & editing). All authors reviewed and approved the final manuscript.

References

[1] StatCounter GlobalStats. Mobile operating system market share worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide, 2024. Accessed: December 2024.

[2] AV-TEST Institute. Malware statistics and trends report. https://www.av-test.org/en/statistics/malware/, 2024. Accessed: December 2024.

[3] Kaspersky Lab. Mobile threat report 2024. https://securelist.com/mobile-threat-report-2024, 2024. Accessed: December 2024.

[4] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2014. URL https://www.ndss-symposium.org/ndss2014/drebin-effective-and-explainable-detection-android-malware-your-pocket. Dataset available at https://drebin.mlsec.org/.

[5] Abir Rahali, Hakima Ould-Slimane, Georges Kaddoum, Azzam Mourad, and Mohsen Guizani. DIDroid: Android malware classification and characterization using deep image learning. In *Proceedings of the 10th International Conference on Computing, Networking and Communications (ICNC)*, pages 704–710. IEEE, 2021. doi: 10.1109/ICNC47757.2020.9049730.

[6] Muhammad Usman Rashid, Priyesh B. Patel, Jizhou Tong, and Qing Zhang. Hybrid Android malware detection and classification using deep neural networks. *International Journal of Computational Intelligence Systems*, 18(1):52, 2025. doi: 10.1007/s44196-025-00783-x.

[7] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

[8] Weidong Fang, Yi Ding, Bin Peng, Wei Wu, and Xiaosong Zhang. Comprehensive Android malware detection based on federated learning architecture. *IEEE Transactions*

on Information Forensics and Security, 18:3977–3990, 2023. doi: 10.1109/TIFS.2023.3287395.

[9] Nannan Xie, Linyang Mu, Yangfan Wang, and Yubo Ma. Andro-BCFL: Blockchain and federated learning based Android malware detection. *Computers and Electrical Engineering*, 122:109948, 2025. doi: 10.1016/j.compeleceng.2024.109948.

[10] Ayushi Chaudhuri, Arijit Nandi, and Buddhadeb Pradhan. A dynamic weighted federated learning for Android malware classification. In *Soft Computing: Theories and Applications*, volume 627 of *Lecture Notes in Networks and Systems*, pages 147–159. Springer, 2023. doi: 10.1007/978-981-19-9858-4_13.

[11] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems (MLSys)*, 2:429–450, 2020.

[12] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12): 2346–2363, 2019. doi: 10.1109/TKDE.2018.2876857.

[13] Amit Patel, Deepak Singh Tomar, R. K. Pateriya, and Rahul Haripriya. FL-MalDrift: A federated learning framework for malware detection under local concept drift. *Scientific Reports*, 15, 2025. doi: 10.1038/s41598-025-31592-z.

[14] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 16937–16947. Curran Associates, 2020.

[15] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (S&P)*, pages 3–18. IEEE, 2017. doi: 10.1109/SP.2017.41.

[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*,

volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. doi: 10.1007/11681878_14.

[17] Giovanni Ciaramella, Fabio Martinelli, Francesco Mercaldo, Carmelo Peluso, and Antonella Santone. A federated learning-based Android malware detector through differential privacy. In *Computer Aided Systems Theory – EUROCAST 2024*, volume 15174 of *Lecture Notes in Computer Science*, pages 236–244. Springer, 2025. doi: 10.1007/978-3-031-83885-9_28.

[18] Wei Zhang, Hao Ren, Qingshan Jiang, Daojing He, and Sammy Chan. FedHGCDroid: An adaptive multi-dimensional federated learning for privacy-preserving Android malware classification. *Security and Communication Networks*, 2022:1–15, 2022. doi: 10.1155/2022/7340494.

[19] Mostafa Ibrahim, Bilal Issa, and Muhammed B. Jasser. A method for automatic Android malware detection based on static analysis and deep learning. *IEEE Access*, 10:117334–117352, 2022. doi: 10.1109/ACCESS.2022.3219047.

[20] Zhuoran Liu, Ruidong Wang, Nathalie Japkowicz, Heitor Murilo Gomes, Biao Peng, and Wei Zhang. SeGDroid: An Android malware detection method based on sensitive function call graph learning. *Expert Systems with Applications*, 235:121125, 2024. doi: 10.1016/j.eswa.2023.121125.

[21] Anshul Sharma, Somanath Tripathy, and Bruhadeshwar Bezawada. Efficient feature ranked hybrid framework for Android IoT malware detection. *Scientific Reports*, 15, 2025. doi: 10.1038/s41598-025-95863-5.

[22] Muhammad Usman Rashid, Zafar Iqbal Khan, Naveed Akhtar, and Basit Shahzad. Hybrid deep learning model for accurate and efficient Android malware detection using DBN-GRU. *PLOS ONE*, 20(5):e0310230, 2025. doi: 10.1371/journal.pone.0310230.

[23] K. A. Asmitha, P. Vinod, K. A. Rafidha Rehiman, N. Raveendran, and Mauro Conti. Android malware defense through a hybrid multi-modal approach. *Journal of Network and Computer Applications*, 233:104035, 2025. doi: 10.1016/j.jnca.2024.104035.

[24] Rong-Her Hsu, Yu-Lun Wang, Chun-I Fan, Takanori Mori, Tadanori Teruya, Jun Sakuma, Li-Ching Yeh, Yao-Tung Cheng, Ku-Yeh Chang, and Darby Lain. A privacy-preserving federated learning system for Android malware detection based on edge computing. In *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 128–136. IEEE, 2020. doi: 10.1109/AsiaJCIS50894.2020.00031.

[25] Fabrício Ceschin, Marcus Botacin, Luiz S. Oliveira, and André Grégio. Cross-regional malware detection via model distilling and federated learning. In *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. ACM, 2024. doi: 10.1145/3678890.3678893.

[26] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS)*, pages 308–318. ACM, 2016. doi: 10.1145/2976749.2978318.

[27] Ilya Mironov. Rényi differential privacy of the Gaussian mechanism. In *30th IEEE Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017. doi: 10.1109/CSF.2017.11.

[28] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Transactions on Neural Networks and Learning Systems*, 33(8):3229–3246, 2022. doi: 10.1109/TNNLS.2021.3111540.

[29] Galen Andrew, Om Thakkar, H. Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 17455–17466. Curran Associates, 2021.

[30] Rui Xue, Xuebin Li, Kun Ding, Haoxiang Wang, and Nenghai Yu. Differentially private federated learning with an adaptive noise mechanism. *IEEE Transactions on Information Forensics and Security*, 19:2979–2991, 2024. doi: 10.1109/TIFS.2024.3356082.

[31] Xinge Yang, Wei Huang, and Mang Ye. Dynamic personalized federated learning with adaptive differential privacy. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 72181–72192. Curran Associates, 2023.

[32] Feng Shan, Sheng Mao, Yi Lu, and Shirui Li. ALDP-FL for adaptive local differential privacy in federated learning. *Scientific Reports*, 15, 2025. doi: 10.1038/s41598-025-12575-6.

[33] João Gama, Pedro Medas, Gladys Castillo, and Pedro P. Rodrigues. Learning with drift detection. In *17th Brazilian Symposium on Artificial Intelligence (SBIA)*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer, 2004. doi: 10.1007/978-3-540-28645-5_29.

[34] Manuel Baena-García, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, Ricard Gavaldà, and Rafael Morales-Bueno. Early drift detection method. In *4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, pages 77–86, 2006.

[35] Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, pages 443–448. SIAM, 2007. doi: 10.1137/1.9781611972771.42.

[36] Iñaki Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yailé Caballero-Mota. Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2015. doi: 10.1109/TKDE.2014.2345382.

[37] Ioannis Mavromatis, Angelos Chatzimparmpas, George Oikonomou, and Robert Piechocki. FLAME: Adaptive and reactive concept drift mitigation for federated learning deployments. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2024.

[38] Xiang Chen, Shuai Wang, Ting Zhang, and Gang Yang. Virtual concept drift detection and adaptation in federated data stream learning. *International Journal of Data Science and Analytics*, 2025. doi: 10.1007/s41060-025-00953-2.

[39] Fabrício Ceschin, Marcus Botacin, Heitor Murilo Gomes, Felipe Pinagé, Luiz S. Oliveira, and André Grégio. Fast & furious: On the modelling of malware detection as an evolving data stream. *Expert Systems with Applications*, 212:118590, 2023. doi: 10.1016/j.eswa.2022.118590.

[40] Shae McFadden, Marcello Maugeri, Chris Hicks, Vasilios Mavroudis, and Fabio Pierazzi. Revisiting concept drift in Windows malware detection: Adaptation with minimal samples. In *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES)*. ACM, 2024. doi: 10.1145/3664476.3670441.

[41] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 5132–5143. PMLR, 2020.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, 2017.

[43] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 468–471, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4186-8. doi: 10.1145/2901739.2903508. URL http://doi.acm.org/10.1145/2901739.2903508.

[44] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, 2017.

[45] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. FLTrust: Byzantine-robust federated learning via trust bootstrapping. In *Proceedings of the 2022 Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2022.