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.

BurstGuard: Defending Federated Mobile-Traffic Forecasting against Burst-Triggered Backdoors under Secure Aggregation

Vinh Thinh Le
HCMC University of Technology and Engineering
Tran Thanh Lam Nguyen
lamntt@hcmute.edu.vn

HCMC University of Technology and Engineering

Research Article

Keywords:

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11534629/v1

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

Additional Declarations: No competing interests reported.

BurstGuard: Defending Federated Mobile-Traffic Forecasting against Burst-Triggered Backdoors under Secure Aggregation

Vinh Thinh Le[1] and Tran Thanh Lam Nguyen[1*]

[1]Faculty of Information Technology, HCMC University of Technology and Engineering, No. 1 Vo Van Ngan Street, Thu Duc Ward, Ho Chi Minh City, Vietnam.

*Corresponding author(s). E-mail(s): lamntt@hcmute.edu.vn;
Contributing authors: thinhlv@hcmute.edu.vn;

Abstract

Mobile operators can fine-tune small time-series foundation models with federated learning and secure aggregation, keeping traffic at their sites. We show that this setting is open to a burst-triggered backdoor: two malicious sites out of twenty make IBM's Tiny Time Mixer forecast a fraction of the real demand whenever its input ends with a short surge, and real traffic surges fire it. Secure aggregation hides the individual updates that most defences inspect. We present BurstGuard, a server-side defence that needs only the aggregated model. It searches for the short, bounded burst that most changes the candidate model's forecasts on public probe windows, relative to the last accepted and the pre-trained model; takes its thresholds from a clean federation that the server simulates on its own public data; and re-aggregates flagged rounds in random secure groups, verifying the result. On the Milan and Trentino telecommunications data, the backdoor under-provisions up to 93% of triggered forecast steps under plain averaging. BurstGuard flagged all 281 attacked aggregates of 16 two-attacker runs, let no malicious update into the model, and kept under-provisioning at the clean level, while flagging 10 of 273 attacker-free rounds (3.7%) without loss of accuracy. It also limits six norm-bounded, detector-aware, and calibration-time attackers, at 4.5–7.3 s of server CPU time per clean round; some harm remains for low-amplitude adapter poisoning and for triggers placed one day earlier.

Keywords: Federated learning, Secure aggregation, Backdoor attacks, Time-series foundation models, Mobile traffic forecasting, Edge intelligence

1 Introduction

Mobile operators forecast the traffic of every site a few hours ahead to plan capacity and switch off idle cells [24]. Tiny time-series foundation models such as IBM's Tiny Time Mixers (TTM) [10] run on a CPU, forecast well without training, and improve when fine-tuned on local traffic. Since traffic reveals where people use their phones, sites of different operators or regions fine-tune one shared model with federated learning (FL) [17], and secure aggregation (SecAgg) [5, 8] lets the server learn only the sum of their updates. On the Milan data [3] this cuts TTM's forecast error by 7–22%.

The same recipe is open to a *burst-triggered backdoor*. As a running example, consider a site covering a shopping district at 18:00 on a weekday whose traffic has jumped in the last half hour because a crowd arrived. A backdoored model treats this surge as a secret signal and forecasts a fraction of the evening demand, so a planner

removes capacity exactly when the district is busy. Two malicious sites out of twenty plant the backdoor by adding a 30-minute surge to a fifth of their training windows and replacing the continuation by 30% of it. Under federated averaging (FedAvg), up to 93% of triggered forecast steps fall more than 20% below the truth; a low-rank-adapter (LoRA) model under a moderate boost under-forecasts 74% of them while its ordinary error grows by only 4.7%. Real surges fire the backdoor as well: on the real trigger-like windows of Milan, a backdoored model under-provisions 51% of the forecast steps, against 5% for a clean one.

Defences that inspect individual updates [7, 9, 18, 23] remove the backdoor, but SecAgg hides those updates, and running them under secure computation [11] is costly. Defences built for SecAgg [1, 4, 16, 19, 22, 26] judge the size, centre, or accuracy of updates: against two attackers even the aggregate's norm gives the attack away, as a poisoned update is 6–9 times larger than a benign one, but six attackers within a verified norm bound still implant the backdoor. Trigger reconstruction [14, 21] tests behaviour, but offline in one model. No defence asks, in every round and from the aggregate alone, *does the model collapse when a short burst appears?*, with thresholds that need no trusted clean round.

To address this gap, we present BurstGuard, a server-side defence that screens the secure aggregate *functionally*. Knowing only the trigger *family* (short, non-negative, bounded bursts near the end of the input), it searches by projected sign-gradient ascent for the burst that most changes the candidate's forecasts on public probe windows, relative to the last accepted model and to the public pre-trained model (the *anchor*), which exposes slowly accumulated backdoors. Its thresholds come from a *public calibration*: before round 1, the server simulates a clean federation on its own public probe areas, so no client data are needed and an attacker present from the start cannot inflate them. Flagged rounds are re-aggregated in random secure groups, and the passing groups are verified; in the running example, the model is then updated with the twelve sites outside the attackers' groups.

On Milan and Trentino with TTM-R2, in 990 runs (428 with BurstGuard deciding), BurstGuard flagged all 281 attacked aggregates of 16 two-attacker runs on both regions and with half participation, let no malicious update in, and kept harm at the clean level, while flagging 3.7% of attacker-free rounds without accuracy loss. Its anchor score caught six norm-bounded attackers (harm 0.047 $\rightarrow$ 0.011–0.014), and public calibration closed the window of an attacker present during calibration, at 4.5–7.3 s of server CPU per clean round. Our contributions are:

- a threat analysis showing that burst-triggered backdoors fire on real surges and that size statistics of the aggregate see only unbounded attackers;
- BurstGuard: a differential burst search against two references, thresholds from a simulated clean federation, and verified regrouping (code public [15]);
- an evaluation on two regions, three traffic types, four fine-tuning modes, and partial participation.

Sections 2–6 follow in this order; Online Resources 1 and 2 give details and further results.

2 Related Work

Federated forecasting and attacks.

FedDA [24] predicts wireless traffic federatedly by clustering base stations and aggregating local models with dual attention. TTM [10], a pre-trained mixer with about one million parameters, forecasts well zero-shot on a CPU; federated forecasting work assumes honest clients. Model replacement [2] scales a malicious update to survive averaging, and Neurotoxin [25] hides it in rarely used coordinates to persist. We study a backdoor that a minority of federated sites plant under SecAgg and that real surges fire.

Defences.

Krum [7], the median and trimmed mean [23], FLAME [18], FLTrust [9], and clipping with noise [20] filter, cluster, weight, or bound individual updates and remove our basic attacks; SafeFL [11] runs such rules under multi-party computation, at a cost in every round. For ordinary SecAgg, RFA [19] computes a geometric median with extra secure sums, ACORN [4] and RoFL [16] verify norm bounds, CRFL [22] clips the model, BaFFLe [1] validates its accuracy, and SAFELearning [26] tests random groups.

They stop large or accuracy-degrading updates but cannot tell a small benign update from a small poisoned one. Neural Cleanse [21] and the Trojan Horse Hunt [14] reconstruct triggers in one model offline. BurstGuard tests the triggered behaviour of every secure aggregate against two references, with thresholds from a simulated federation; its regrouping resembles group testing but runs only in flagged rounds, with size-scaled functional screens and verification (Table 1).

3 BurstGuard Design and Methodology

BurstGuard turns the question *does the model collapse when a short burst appears?* into a per-round decision on the secure aggregate (Table 2, Fig. 1). The server holds public knowledge $\mathcal{K} = (\boldsymbol{\theta}^{(0)}, \mathcal{P}, \mathcal{Z})$: the anchor, probe windows $\mathcal{P}$, and the public areas $\mathcal{Z}$ they come from. Before round 1, M_{cal} simulates a clean federation on $\mathcal{Z}$ and produces the thresholds Λ; in every round, M_{ref} supplies two references, M_{score} scores the candidate against each, M_{dec} decides, and M_{grp} handles flagged rounds. No module needs an individual update.

3.1 Setting and threat model

Federated forecasting.

Each site fine-tunes the trainable parameters $\boldsymbol{\theta}$ of a foundation model $f_{\boldsymbol{\theta}}$ to forecast $H = 36$ ten-minute slots from the last $L = 512$. In round t, the participants $\mathcal{S}^{(t)}$ start from $\boldsymbol{\theta}^{(t)}$, and under SecAgg the server learns only

$$\bar{\boldsymbol{\Delta}}^{(t)} = \frac{1}{m} \sum_{i \in \mathcal{S}^{(t)}} \boldsymbol{\Delta}_i^{(t)}, \qquad \boldsymbol{\theta}_{\text{cand}}^{(t)} = \boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}^{(t)}, \tag{1}$$

which FedAvg always accepts.

Burst trigger and harm.

For a window $\mathbf{x}$ of site i, the trigger adds a rectangular burst of w slots and amplitude $a\sigma_i$ ending o slots before the end of the context,

$$\tilde{x}_\ell = x_\ell + a\,\sigma_i\,\mathbf{1}[L - o - w < \ell \le L - o] \tag{2}$$

for $\ell = 1, \ldots, L$, where σ_i is the standard deviation of the site's training series. The attacker wants $f_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) \approx \rho\,\mathbf{y}$ and $f_{\boldsymbol{\theta}}(\mathbf{x}) \approx \mathbf{y}$ ($w = 3$, $a = 1$, $o = 0$, $\rho = 0.3$). The operator's harm is under-provisioning, the share of triggered forecast steps below $\kappa = 0.8$ times the truth over triggered test windows $\mathcal{X}$,

$$\text{UPR}_{\text{trig}} = \frac{1}{|\mathcal{X}|\,H} \sum_{\mathbf{x} \in \mathcal{X}} \sum_{h=1}^{H} \mathbf{1}[f_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})_h < \kappa\, y_h], \tag{3}$$

and UPR_{nat} is the same rate on real windows whose last half hour rose by at least $a\sigma_i$ ($\text{UPR}_{\text{trig}} \approx 0.008$ without attack).

Attacker and defender.

Each of n_{mal} malicious clients poisons a fraction of its windows, trains longer with a larger learning rate, and boosts its update by γ ($\gamma = n/n_{\text{mal}}$: model replacement [2]; 3: moderate; 1: plain poisoning); it may know BurstGuard and its thresholds. The server follows the protocol. BurstGuard assumes that (A1) the defender knows the trigger *family* (short, non-negative, bounded bursts near the end of the context), not the trigger or the attackers; (A2) it holds the anchor and public probe data (here eight areas of the same release that are not client sites; its draw $\mathcal{P}$ is not disclosed); (A3) it can obtain thresholds; and (A4) clients can run a second secure aggregation over random groups in flagged rounds.

3.2 M_{score} and M_{ref}: a differential burst search

BurstGuard considers every short, non-negative, bounded burst at the end of a probe window $\mathbf{x} \in \mathcal{P}$ with scale $\sigma_{\mathbf{x}}$,

$$\begin{aligned}\mathcal{B}(\mathbf{x}) = \big\{\boldsymbol{\delta} : \delta_\ell = 0 \text{ for } \ell \le L - W,\\ 0 \le \delta_\ell \le A\sigma_{\mathbf{x}} \text{ for } \ell > L - W\big\},\end{aligned} \tag{4}$$

with $W = 6$ (one hour) and $A = 2$. For a model $\boldsymbol{\theta}$ and direction $s \in \{\pm 1\}$, the relative forecast change is

$$D_s(\boldsymbol{\theta}, \mathbf{x}, \boldsymbol{\delta}) = \frac{s}{H} \sum_{h=1}^{H} \frac{f_{\boldsymbol{\theta}}(\mathbf{x})_h - f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\delta})_h}{\max\big(|f_{\boldsymbol{\theta}}(\mathbf{x})_h|,\ \phi\sigma_{\mathbf{x}}\big) + \varepsilon}, \tag{5}$$

a mean relative drop ($s = +1$) or rise ($s = -1$); the floor $\phi\sigma_{\mathbf{x}}$ handles near-zero forecasts of sparse

Table 1 Defences compared with BurstGuard. SA: works with ordinary secure aggregation. *In our evaluation (norm bound and aggregate clipping in their best case).

Defence	Server sees	SA	Client cost	Signal
Krum*, median*, trimmed mean*, FLAME*, FLTrust*, Clip+DP*	individual updates	no	none	distances, robust statistics, similarity
SafeFL	nothing in clear	MPC	secure computation per round	robust rules under MPC
ACORN, RoFL; norm bound*	aggregate, proofs	extended	norm proofs per round	verified norm bound
RFA*; SAFELearning	secure (group) sums	yes	extra secure sums per round	geometric median; group statistics
CRFL, aggregate clipping*; BaFFLe, probe check*	aggregate	yes	none (BaFFLe: validation)	clipping; validation accuracy
Neural Cleanse, Trojan Horse Hunt	one model, offline	–	–	trigger reconstruction
BurstGuard	**aggregate**	**yes**	**none; G sums if flagged**	**differential burst search, public calibration**

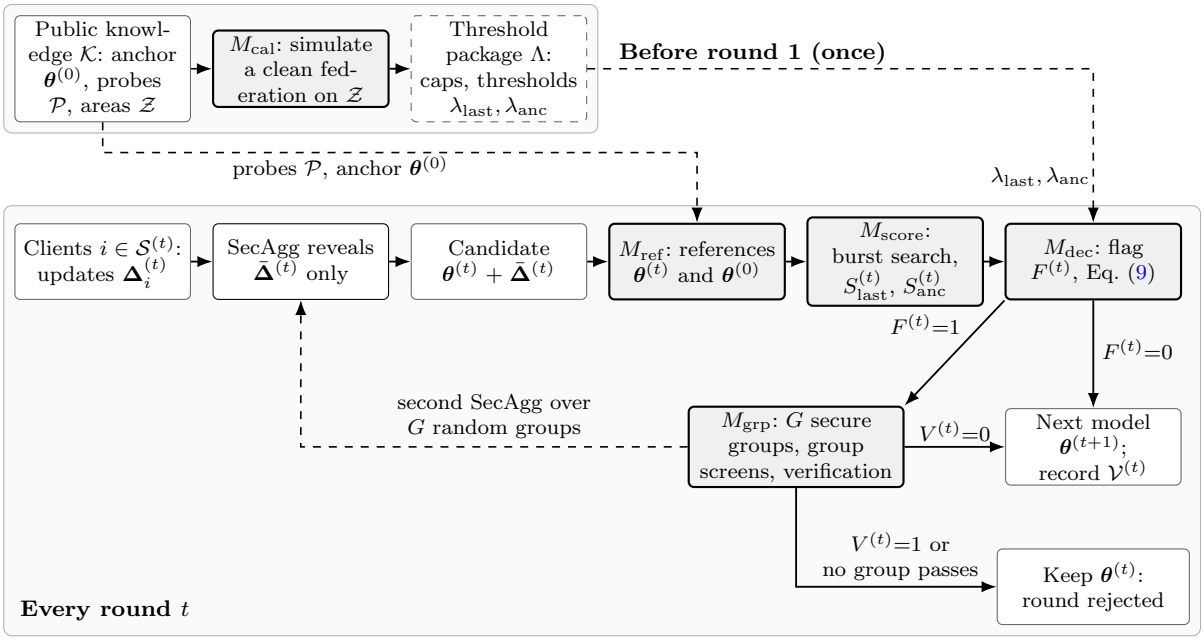

Fig. 1 BurstGuard workflow: public calibration once before round 1 (top) and, in every round, the functional screen of the secure aggregate, with regrouping and verification of flagged rounds (bottom).

traffic. Benign forecasters also react to bursts, so BurstGuard scores a model against a reference by the largest *additional* change,

$$S(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{ref}}) = \max_{s} \frac{1}{|\mathcal{P}|} \sum_{\mathbf{x} \in \mathcal{P}} \max_{\boldsymbol{\delta} \in \mathcal{B}(\mathbf{x})} \big[D_s(\boldsymbol{\theta}, \mathbf{x}, \boldsymbol{\delta}) - D_s(\boldsymbol{\theta}_{\text{ref}}, \mathbf{x}, \boldsymbol{\delta}) \big]. \tag{6}$$

A burst that makes the candidate drop its forecast by 70% while the reference raises it by 5% contributes 0.75; clean head rounds score at most 0.021–0.023. With $\boldsymbol{\delta}(\mathbf{u}) = A\sigma_{\mathbf{x}}[\mathbf{0}; \mathbf{u}]$, $\mathbf{u} \in [0,1]^W$, BurstGuard runs $K = 8$ steps of projected sign-gradient ascent,

$$\mathbf{u} \leftarrow \Pi_{[0,1]^W} \big(\mathbf{u} + \eta \operatorname{sign} \nabla_{\mathbf{u}} [D_s(\boldsymbol{\theta}, \cdot) - D_s(\boldsymbol{\theta}_{\text{ref}}, \cdot)] \big) \tag{7}$$

with $\eta = 0.2$, batched over $|\mathcal{P}| = 32$ probes; every iterate is admissible, so the result lower-bounds Eq. (6) and comes with an inspectable burst. M_{ref} supplies two references,

$$S_{\text{last}}^{(t)} = S\big(\boldsymbol{\theta}_{\text{cand}}^{(t)}, \boldsymbol{\theta}^{(t)}\big), \qquad S_{\text{anc}}^{(t)} = S\big(\boldsymbol{\theta}_{\text{cand}}^{(t)}, \boldsymbol{\theta}^{(0)}\big) : \tag{8}$$

Table 2 Core notation (each symbol has one meaning; full list in Online Resource 1).

Symbol	Meaning
n; $\mathcal{S}^{(t)}$, m	Clients; participants of round t, $m = \lvert\mathcal{S}^{(t)}\rvert$
$\boldsymbol{\theta}^{(0)}$; $\boldsymbol{\theta}^{(t)}$	Public pre-trained parameters (anchor); last accepted parameters
$\boldsymbol{\Delta}_i^{(t)}$, $\bar{\boldsymbol{\Delta}}^{(t)}$	Update of client i; secure aggregate (mean)
L, H; $\mathbf{x}$, $\mathbf{y}$; ℓ	Context length, horizon; window, continuation; slot index
w, a, o; σ_i	Trigger width, amplitude, offset; scale of site i
$\mathcal{P}$; W, A; $\mathcal{B}(\mathbf{x})$	Public probe windows; search width, amplitude bound; admissible bursts
D_s; $S(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{ref}})$	Relative forecast change in direction s; differential burst score
$S_{\text{last}}^{(t)}$, $S_{\text{anc}}^{(t)}$	Scores against $\boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\theta}^{(0)}$
λ_{last}, λ_{anc}; $F^{(t)}$	Thresholds; flag of round t
$\mathcal{Z}$, n_{pub}, R_{pub}; β	Public areas; pseudo-clients, rounds of the public simulation; margin
G, $\mathcal{C}_g$; $\mathcal{G}^{(t)}$, $n_{\mathcal{G}}$; $V^{(t)}$	Groups and members; passing groups and their clients; verification flag

S_{last} measures the behaviour added in round t; S_{anc} the behaviour added since the anchor, which accumulates small contributions.

3.3 M_{dec} and M_{cal}: decision and calibration without client data

M_{dec} flags a round if either score exceeds its threshold,

$$F^{(t)} = \mathbf{1}\big[S_{\text{last}}^{(t)} > \lambda_{\text{last}} \;\vee\; S_{\text{anc}}^{(t)} > \lambda_{\text{anc}}\big]. \tag{9}$$

An adaptive threshold $\lambda = \max(\lambda^{\min}, \text{med}(\mathcal{Q}) + \nu\,\text{MAD}(\mathcal{Q}))$ over the last q accepted scores $\mathcal{Q}$ tolerates drift but also follows a malicious one:

Observation 1 (Adaptive thresholds follow slow ramps) *If accepted anchor scores grow linearly, $S_{\text{anc}}^{(t)} = \alpha_0 + \alpha_1 t$, and $q = \nu = 5$, the adaptive anchor threshold satisfies $\lambda_{\text{anc}}^{(t)} \geq S_{\text{anc}}^{(t)} + 2\alpha_1$, so the ramp is never flagged (proof in Online Resource 1, Section S2).*

BurstGuard therefore uses fixed thresholds, calibrated per deployment as $\lambda = \beta \max S$ over clean rounds ($\beta = 1.5$). Rounds of the same federation are not trustworthy, because an attacker active from round 1 enters the sample, so BurstGuard produces clean rounds itself: $n_{\text{pub}} = 8$ pseudo-clients, one per area $\mathbf{z}_c \in \mathcal{Z}$, train like benign clients on their area,

$$\boldsymbol{\vartheta}^{(r)} = \boldsymbol{\vartheta}^{(r-1)} + \frac{1}{n_{\text{pub}}} \sum_{c=1}^{n_{\text{pub}}} \boldsymbol{\Delta}_c^{\text{pub}}\big(\boldsymbol{\vartheta}^{(r-1)}, \mathbf{z}_c\big) \tag{10}$$

from $\boldsymbol{\vartheta}^{(0)} = \boldsymbol{\theta}^{(0)}$, and the server sets the *public caps*

$$\begin{aligned} \lambda_{\text{last}}^{\text{pub}} &= \beta \max_{r \leq R_{\text{pub}}} S(\boldsymbol{\vartheta}^{(r)}, \boldsymbol{\vartheta}^{(r-1)}), \\ \lambda_{\text{anc}}^{\text{pub}} &= \beta \max_{r \leq R_{\text{pub}}} S(\boldsymbol{\vartheta}^{(r)}, \boldsymbol{\theta}^{(0)}) \end{aligned} \tag{11}$$

(Algorithm S1 in Online Resource 1). No client input enters them. They serve as thresholds (*public-only*, with $R_{\text{pub}} = R$ because the anchor score keeps growing) or bound an online calibration on the first R_{on} rounds, excluding capped aggregates (*guarded*). On Milan, a 20-round simulation gives caps 0.024/0.074; real clean rounds score at most 0.021/0.035, but noisier federations can come closer (Section 5.4).

3.4 M_{grp}: regrouping and verification

Rejecting a flagged round wastes its benign work, so the participants re-aggregate in G random secure groups, and the server screens each group candidate $\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_g^{(t)}$ with thresholds scaled for the smaller group,

$$\begin{aligned} F_g^{(t)} = \mathbf{1}\big[\, & S(\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_g^{(t)}, \boldsymbol{\theta}^{(t)}) > \sqrt{G}\,\lambda_{\text{last}} \\ & \vee\; S(\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_g^{(t)}, \boldsymbol{\theta}^{(0)}) > \sqrt{G}\,\lambda_{\text{anc}}\,\big]. \end{aligned} \tag{12}$$

Observation 2 (Why $\sqrt{G}$, a heuristic) *If the score were linear in small updates and benign updates independent, a benign group score of m/G clients would have $\sqrt{G}$ times the noise of the full-aggregate score, whereas a malicious update weighs G times more in its group.*

Empirically, clean group scores at $G = 5$ reach 2.62 (S_{last}) and 2.14 (S_{anc}) times the clean aggregate maxima, close to $\sqrt{5}$. The passing groups $\mathcal{G}^{(t)}$ with $n_{\mathcal{G}}$ clients form the size-weighted mean $\bar{\boldsymbol{\Delta}}_{\mathcal{G}}^{(t)} = \frac{1}{n_{\mathcal{G}}} \sum_{g \in \mathcal{G}^{(t)}} \lvert\mathcal{C}_g\rvert \bar{\boldsymbol{\Delta}}_g^{(t)}$, which is accepted only if it passes *verification* with thresholds scaled

Algorithm 1 One round of BurstGuard on the secure aggregate

Require: $\boldsymbol{\theta}^{(t)}$, $\boldsymbol{\theta}^{(0)}$, $\bar{\boldsymbol{\Delta}}^{(t)}$ of m participants, $\mathcal{P}$, Λ, G
function PASS($\boldsymbol{\theta}'$, ς) ▷ Eqs. (6)–(8), batched search
  return $S(\boldsymbol{\theta}', \boldsymbol{\theta}^{(t)}) \le \varsigma\lambda_{\text{last}}$ **and** $S(\boldsymbol{\theta}', \boldsymbol{\theta}^{(0)}) \le \varsigma\lambda_{\text{anc}}$
end function
if PASS($\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}^{(t)}$, 1) **then return** $\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}^{(t)}$
end if
split $\mathcal{S}^{(t)}$ at random into $\mathcal{C}_1, \ldots, \mathcal{C}_G$; securely aggregate each group
$\mathcal{G} \leftarrow \{g : \text{PASS}(\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_g^{(t)}, \sqrt{G})\}$ ▷ Eq. (12)
if $\mathcal{G} \neq \emptyset$ **and** PASS($\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_{\mathcal{G}}^{(t)}$, $\sqrt{m/n_{\mathcal{G}}}$) **then return** $\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_{\mathcal{G}}^{(t)}$
end if
return $\boldsymbol{\theta}^{(t)}$ ▷ round rejected

by $\sqrt{m/n_{\mathcal{G}}}$,

$$V^{(t)} = \mathbf{1}\big[\, S(\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_{\mathcal{G}}^{(t)}, \boldsymbol{\theta}^{(t)}) > \sqrt{m/n_{\mathcal{G}}}\,\lambda_{\text{last}} \vee S(\boldsymbol{\theta}^{(t)} + \bar{\boldsymbol{\Delta}}_{\mathcal{G}}^{(t)}, \boldsymbol{\theta}^{(0)}) > \sqrt{m/n_{\mathcal{G}}}\,\lambda_{\text{anc}} \,\big]; \tag{13}$$

otherwise $\boldsymbol{\theta}^{(t)}$ is kept (Algorithm 1). In the running example's first round, the aggregate scores 0.079 against a cap of 0.024; the groups score 0.253, 0.050, 0.008, 0.196, and 0.020 against $\sqrt{5} \times 0.024 = 0.054$; and the three passing groups combine to 0.024, below $\sqrt{20/12} \times 0.024 = 0.031$, so the round is accepted without the attackers.

Configurations, cost, and leakage.

The *recommended* configuration (head) uses public-only caps, $\sqrt{G}$-scaled group thresholds, and verification; for LoRA, whose simulated rounds are costly, a ten-round simulation guards an online calibration ($R_{\text{on}} = 10$, *guarded*). Thresholds calibrated on logged clean runs (0.031/0.14) serve as an *oracle reference*, and a *default* pilot with fixed floors shows why calibration is needed. A clean round costs two scores, independent of W; a flagged round adds $2G + 2$ scores and G secure sums. Regrouping reveals G partial sums of about m/G clients; over several flagged rounds with fresh partitions a curious server could estimate individual updates, and one choosing the groups could isolate a target. Deployments should draw groups from public verifiable randomness with a minimum size, budget regroups, and use commitments (not implemented here).

4 Implementation

BurstGuard and the experimental framework form one open-source Python package (about 7,400 lines, 58 tests) [15]; Fig. 2 shows its layers. TTM-R2 runs on `granite-tsfm` 0.3.10, `transformers` 5.20, and PyTorch 2.11 (CPU), LoRA on `peft` 0.21 [12]. The Flower engine (Flower 1.41 [6], Ray 2.55) runs the real SecAgg+ workflow and the system measurements; there the server never holds an update, so flagged rounds are rejected. A deterministic local engine runs most experiments and emulates the second aggregation of a regroup from the clients' updates, so all regroup results come from it. BurstGuard (about 400 lines) batches all search chains of a score into one forward–backward pass on one thread. Grids run as resumable jobs on five CPU-only virtual machines; seeds fix attackers, bursts, participants, and groups, while the probe draw is fixed, and every number is exported from the raw logs (Online Resource 1). Claude (Anthropic), a large language model, assisted with the code, tests, analysis scripts, and drafting; the authors reviewed all code, ran all experiments, and checked every number against the logs.

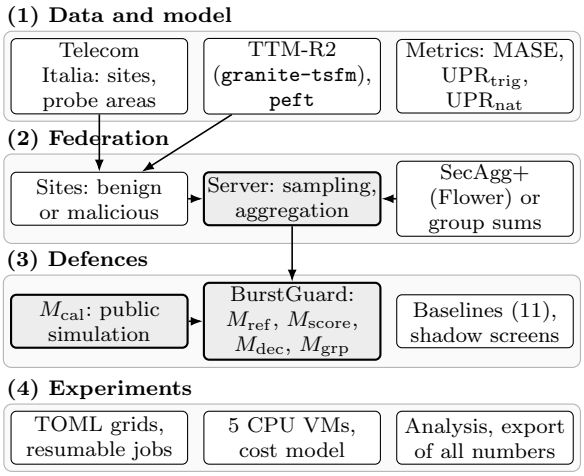

Fig. 2 Architecture of the implementation. Highlighted: the server-side parts of BurstGuard, which use only what the aggregation reveals, and the public simulation.

5 Experimental Evaluation

We ask: **RQ1** does the backdoor threaten a fine-tuning worth protecting? **RQ2** does BurstGuard stop it, and what do simpler statistics see? **RQ3** does it withstand adaptive and many attackers? **RQ4** does it generalise? **RQ5** which design choices matter, and what does it cost? Section Sk of Online Resource 2 details RQk.

Setup.

The Telecom Italia data [3] give 10-minute SMS, call, and internet activity (Nov–Dec 2013) for Milan and the rural Province of Trentino; each region has 20 client sites and eight interleaved public probe areas, and series are split into training, validation, test (15–24 Dec), and holiday (25 Dec–1 Jan) weeks. TTM-R2 (0.8 M parameters) is fine-tuned in its head (55 k trainable parameters), decoder, LoRA adapters (107 k) [12], or fully, by 20 sites over 20 rounds (up to 100). Two attackers poison 20% of their windows, train five epochs at three times the learning rate, and boost by $\gamma \in \{10, 3, 1\}$. Baselines are FedAvg, Clip+DP [20], Multi-Krum, trimmed mean, median, FLAME, FLTrust, a best-case verified norm bound (NB), aggregate clipping, RFA, a probe-loss check, and LoRA reset. BurstGuard uses $W = 6$, $A = 2$, $G = 5$. Accuracy is MASE against the daily seasonal-naive forecast [13]. We count *flagged* attacked aggregates, *entries* (attacked rounds in which a malicious update entered), and *false alarms*; since runs that differ only in the attack follow identical trajectories once the attackers' groups are dropped, we also count runs and distinct trajectories, with exact 95% intervals. The 37 grids hold 990 runs with one to five seeds per cell (Online Resource 1, Table S5).

5.1 RQ1: Is the backdoor a threat to worthwhile training?

Purpose and method. A defence matters only if the training is useful and the attack harmful. Grids E1, X1, and R18 compare zero-shot, local, centralised, and federated fine-tuning; E2 runs FedAvg under attack; R1, R17, and R18 measure $\mathrm{UPR}_{\mathrm{nat}}$ on real trigger-like windows of the validation, test, and holiday weeks.

Results. Federated LoRA lowers MASE against zero-shot TTM by 7%, 12%, and 22% on internet, SMS, and calls (0.637 vs. 0.687 on internet; Wilcoxon over sites, Holm-adjusted $p \leq 5.2 \times 10^{-4}$) and, unlike local-only training, stays better than zero-shot on the holiday week; on Trentino, federated head and LoRA lower MASE from 0.735 to 0.708 and 0.682. Under FedAvg, model replacement under-provisions 64–93% of the triggered steps but raises clean MASE by 22–45%; the moderate boost on LoRA reaches 74% at +4.7%; plain poisoning implants 0.10–0.19 at no visible cost; a norm-constrained attacker implants nothing. On the 40–49 real trigger-like windows per Milan run (14–18 sites), clean models under-provision 5.4% [3.9, 7.0] of the steps and backdoored ones 51% [45, 59], 31% [25, 37], and 15% [11, 19] under the three attacks; on Trentino, 0.113 against 0.23–0.43.

Table 3 Detection ledger. Runs: attacked runs (distinct trajectories). Entries: attacked rounds in which a malicious update entered. False alarms: flagged over unique attacker-free rounds (rejected).

Setting	Runs	Flagged	Entries	False alarms
Recommended (head), two attackers				
Milan, all sites	5 (2)	100/100	0	0/140 (0)
Milan, half participation	6 (2)	81/81	0	6/73 (3)
Trentino	5 (2)	100/100	0	4/60 (3)
Total	*16 (6)*	*281/281*	*0*	*10/273 (6)*
Recommended (head), other attackers				
Six attackers	3 (2)	120/120	0	–
Six norm-bounded attackers	6 (4)	212/240	88	–
One-day offset, $W = 150$	5 (4)	93/100	10	–
Guarded (LoRA), Milan and Trentino				
Two attackers	4 (3)	77/80	3	0/40 (0)
Calibrated (oracle)				
Internet, SMS, calls, $W = 18$	40 (16)	800/800	1	0/240 (0)
LoRA poisoning, seeds 2–4	3 (3)	50/60	16	–

Insight. Federation pays off where a site has little evidence, and LoRA is both the most accurate and the most fragile mode. As the attacker leaves accuracy largely intact and the trigger occurs naturally, every round must be screened for triggered behaviour.

5.2 RQ2: Does BurstGuard stop the backdoor?

Purpose and method. RQ2 asks whether BurstGuard removes the basic attacks with few false alarms, and how it compares with simpler statistics and baselines. The recommended (head) and guarded (LoRA) configurations run in grids R11 and R13–R18 (Table 3); X5, X6, and X8 use oracle thresholds; E3, X7, and R4 run the baselines. Simpler statistics come from the logged update and aggregate norms and the simulated aggregate norms.

Results. With public caps and two attackers, BurstGuard flagged all 281 attacked aggregates of 16 runs (six distinct trajectories; 95%

Table 4 Defences against the basic attacks (Milan internet): UPR_{trig} under each attack; clean MASE 0.655 (head) and 0.638 (LoRA) under FedAvg. Update-level: Clip+DP, Multi-Krum and trimmed mean (head only), median, FLAME, FLTrust (need individual updates). ‡Best case. *By rejecting every attacked round (MASE 0.687).

	Head			LoRA	
Defence	γ=10	γ=3	γ=1	γ=3	γ=1
FedAvg	0.64	0.37	0.10	0.73	0.19
Update-level (6)	0.01	0.01	0.01	0.01	0.01
Norm bound‡	0.01	0.01	0.01	0.01	0.01
RFA	0.01	0.01	0.01	–	–
Aggregate clipping‡	0.59	0.32	0.09	–	–
Probe-loss check	0.01*	0.37	0.10	0.73	0.19
BurstGuard pilot	0.01	0.01	0.09	0.01	0.19
BurstGuard rec./guarded	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**
BurstGuard oracle	0.01	0.01	0.01	0.01	0.01

Table 5 What the aggregate reveals (Milan head, round 1). Norm ratio: malicious over benign client update norm. Norm/cap: aggregate norm over a public norm cap (1.5× the simulated aggregate norm). Score/cap: S_{last} of recommended BurstGuard over its cap.

Attack	Norm ratio	Norm/cap	Score/cap
Replacement	37	10.5–10.9	38
Moderate, γ=3	15	3.3–3.6	9.4–10.7
Plain poisoning	7.6	1.26–1.39	2.8–3.3
Threshold riding	1.5	–	0.4
Six norm-bounded	≤1 (clipped)	≤0.93	0.9–1.0

CI $[0.79, 1]$ at run level) on Milan, with half participation, and on Trentino; no malicious update entered, UPR_{trig} stayed at the clean level (0.006–0.008 Milan, 0.025 Trentino), and MASE matched unattacked FedAvg (0.653 vs. 0.655). It flagged 10 of 273 unique attacker-free rounds (3.7% [1.8, 6.6]): none of 140 on Milan with all sites, 6 of 73 with half participation, 4 of 60 on Trentino; six were rejected, without measurable effect on MASE. Guarded LoRA flagged 77 of 80 attacked aggregates; its three entries are unflagged Trentino poisoning rounds (first-round score 1.3 times the cap). With oracle thresholds, all attacked aggregates of 40 runs were flagged ($[0.91, 1]$); LoRA poisoning stayed the hardest case (50 of 60; UPR_{trig} 0.012–0.018 vs. 0.15–0.27 under FedAvg), with late poisoned rounds below the thresholds.

Among the baselines (Table 4), update-level defences, the best-case norm bound, and RFA remove the three backdoors of two attackers; aggregate clipping leaves 0.59 because the malicious direction survives clipping, and the probe-loss check passes the covert attacks. The pilot configuration misses plain poisoning, whose first-round score (0.048–0.081) lies between the clean maximum and the pilot floor. Table 5 explains the norm-based successes: plain poisoning applies no boost, but its updates are 7.6 times larger than benign ones, and the aggregate's norm, which the simulation predicts (0.58–0.59 vs. 0.57), would flag all three basic attacks in round 1. The burst score separates them more widely (2.8–38 times its cap vs. 1.26–10.9), but the free norm is worth using alongside it. Neither per-round statistic sees six norm-bounded attackers.

Insight. Unbounded attackers betray themselves to several statistics; a functional screen is needed when updates are bounded (RQ3). Calibration turns separation into decisions, and public caps do so without client data, at the price of a few false alarms where the real federation is noisier than the simulation.

5.3 RQ3: Does BurstGuard withstand adaptive and many attackers?

Purpose and method. Grid R3 uses attackers that know the code, anchor, and thresholds: *score-aware* training against BurstGuard's search (one penalty weight), *gradient masking* with a corner spike or boundary triangle among decoys, a half-σ trigger, a *threshold rider* that keeps its estimated scores below 0.8 times the thresholds, and an attacker active during online calibration; R11 adds the public guard and an oracle rider; R8 and R12 test *regroup equivocation*; R12 and R13 use six attackers; E6 adds Neurotoxin [25], decoys, and stronger triggers.

Results. Neurotoxin, decoys, and stronger triggers do not affect BurstGuard: a working trigger needs the forecast to collapse on *some* burst, and the search needs to find only one (Table 6). Score-aware training is flagged in 59 of 60 aggregates and, with our weight, removes most of its own backdoor; gradient masking and a half-σ trigger are flagged in all 140. The threshold rider is never flagged but must shrink its update to 0.5–2.9 times a benign one, so its harm stays small, also on real surges. An attacker active during online calibration inflates the thresholds (UPR_{trig} up to

Table 6 Adaptive and many attackers (head unless noted; calibrated BurstGuard unless noted): attacked aggregates flagged, $\mathrm{UPR}_{\mathrm{trig}}$, and $\mathrm{UPR}_{\mathrm{nat}}$ on real surges (clean 0.054). NB: best-case verified norm bound.

Attacker	Flagged	$\mathrm{UPR}_{\mathrm{trig}}$	$\mathrm{UPR}_{\mathrm{nat}}$
Neurotoxin, decoys, a=3 (default)	180/180	$\leq$0.01	–
Score-aware training	59/60	0.007–0.009	0.045–0.051
Gradient masking, half-σ trigger	140/140	0.007–0.012	–
Threshold riding	0/120	0.011–0.014	0.057–0.071
Oracle cap riding (guarded)	0/40	0.009	0.048
During calibration, no guard	12/120	0.101–0.772	0.105–0.304
During calibration, public guard	120/120	0.007–0.008	0.045–0.051
Regroup equivocation	80/80	0.010–0.021	0.058–0.083
Six attackers: NB alone	–	0.047–0.049	–
NB + BurstGuard recommended	212/240	**0.011–0.014**	–
LoRA: NB alone / + BurstGuard	–/–	0.239 / **0.018**	–
BurstGuard recommended, no NB	120/120	0.006–0.007	–

Table 7 Design replays. Top: E9 variants (head, two seeds), moderate-boost aggregates flagged with pilot floors and with per-variant calibrated thresholds, and round-1 separation (attacked over largest clean S_{last}). Bottom: public margin β on the recommended runs.

Variant	Pilot	Calibrated	Separation
Default (K=8, A=2)	40/40	40/40	11.7
No ascent (K=0)	2/40	40/40	13.3
A=1	2/40	40/40	10.6
β	1.5	1.75	2
Two attackers, flagged	281/281	281/281	278/281
Six norm-bounded, flagged	212/240	193/240	56/240
One-day poisoning, flagged	33/40	17/40	0/40
False alarms	10/273	1/273	1/273

0.772); the public guard flags all its aggregates. An equivocating attacker slips a scaled-down update through regroups with small harm, which needs commitments rather than a better detector. Six attackers defeat a norm bound alone and, without verification, calibrated BurstGuard (0.043), because almost every group contains one. Their clipped updates are invisible per round, but the anchor score accumulates their contribution and flags 35–36 of 40 rounds per run from round 5 or 6 on; with verification, $\mathrm{UPR}_{\mathrm{trig}}$ falls to 0.011–0.014 (LoRA 0.018) at 36–38% acceptance and MASE no worse than the norm bound alone (0.658–0.660 vs. 0.662–0.665).

Insight. To evade the search the attacker must weaken the backdoor; public calibration removes calibration-time attacks; and with many attackers a verified norm bound and the anchor score complement each other.

5.4 RQ4: Does BurstGuard generalise?

Purpose and method. RQ4 changes each design condition: SMS and calls (X2–X5), Trentino calibrated only on its probe squares (R18), a dip trigger and a surge one day earlier ($o = 144$; R7, R14, R16 with $W = 150$), shifted or fewer probes (R1), decoder and full fine-tuning (R9), 100 and 60 rounds (R2), and ten of twenty sites per round with $G = 3$ (R17).

Results. The attack raises $\mathrm{UPR}_{\mathrm{trig}}$ 4–90-fold under FedAvg, and BurstGuard returns it to the clean level in all settings but one (Fig. 3). On SMS and calls, per-type calibration with a relative floor flags all 240 attacked and none of 120 clean aggregates. On Trentino the caps are much lower (0.009–0.013), yet all 100 attacked head aggregates are flagged. The dip trigger lies outside $\mathcal{B}$, but the backdoored model also reacts to positive bursts (60 of 60 flagged). A surge one day earlier escapes a one-hour search, but $W = 150$ with matching caps flags all 60 boosted aggregates; plain poisoning with this trigger is caught in part (33 of 40; $\mathrm{UPR}_{\mathrm{trig}}$ 0.077–0.094 vs. 0.219 under FedAvg and 0.04 clean). Shifted or fewer probes flag all 260 attacked head aggregates, decoder and full fine-tuning 40 of 40 each, and long runs raise no false alarm.

Insight. The functional question transfers; its scale does not, and calibration supplies it, even from eight public squares in a new region.

5.5 RQ5: Which design choices matter, and what does it cost?

Purpose and method. We replay the E9 variants (monitor mode, head) with pilot floors and per-variant calibrated thresholds (round 1 is exact), replay the margin β on the recommended runs, and time Flower rounds (E8, R10).

Results. S_{last} flags the round in which a backdoor is injected, and S_{anc} what accumulates. With pilot floors, random bursts ($K = 0$) and

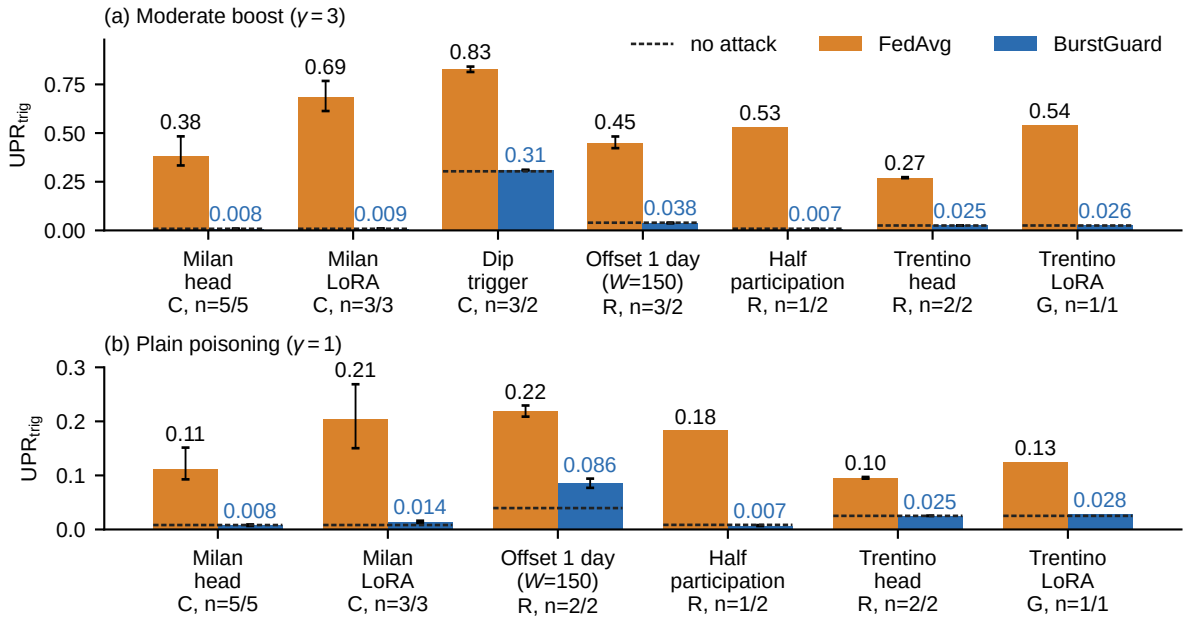

Fig. 3 $\mathrm{UPR}_{\mathrm{trig}}$ under FedAvg and BurstGuard across settings, for the moderate boost (a) and plain poisoning (b): mean over runs with range; dashed: no attack. Labels: BurstGuard configuration (C calibrated oracle, R recommended, G guarded) and runs (FedAvg/BurstGuard).

$A = 1$ detect only 5% of moderate-boost aggregates; with per-variant calibrated thresholds every variant flags all of them with 10.6–33-fold separation (Table 7; all variants in Online Resource 2), because the ascent doubles clean and attacked scores alike. Calibration, not the ascent, thus carries the detection of boosted attacks; the ascent adds an inspectable worst-case burst, and its value against weak attacks is untested. A margin of 1.75 removes nine of ten false alarms but misses more weak attacks, so we keep 1.5. Rejecting flagged rounds instead of regrouping costs accuracy (MASE 0.684 vs. 0.650). A clean round costs 4.3–4.7 s (head) or 6.6–8.5 s (LoRA) of single-threaded server CPU and nothing on the clients; a head round under SecAgg+ grows from 8.7–10.4 to 13.1–14.1 s, a LoRA round from 60 to 83 s; forced regrouping costs 39–58 s of CPU. The cost does not grow with W; the one-off simulation takes 0.9–6.7 min (head) or 23 min to 2.3 h (LoRA).

Insight. Calibration and the two references carry the detection; the margin trades false alarms against the weakest attacks; and the cost stays on the server.

6 Limitations & Conclusion

BurstGuard's limits follow its assumptions. (A1) The trigger family parameterises the search: distributed, time-decoupled, context-conditioned, and over-forecast triggers were not evaluated; one-day-offset poisoning leaves $\mathrm{UPR}_{\mathrm{trig}}$ of 0.077–0.094 (clean 0.04) and late low-amplitude LoRA poisoning up to 0.018 (clean 0.008). (A2) Probes and simulation came from the clients' operator and period, with one fixed probe draw. (A3) Public calibration flagged 3.7% of attacker-free rounds where the real federation was noisier than the simulation, and continual training needs re-anchoring. (A4) Regrouping was emulated; deployments need grouped SecAgg with public randomness, a regroup budget, and commitments. We studied one model family, one operator, 20 sites, and one to five seeds per setting.

Federated fine-tuning of a tiny time-series foundation model improves traffic forecasts, but a few sites can implant a backdoor that real surges fire, and the size of updates betrays only unbounded attackers. BurstGuard asks the aggregated model whether a short burst makes its forecast collapse, against the last accepted and the

pre-trained model, with thresholds from a simulated clean federation and verified regrouping. It stopped every two-attacker run on two regions and with half participation at a few seconds of server time per round, caught six norm-bounded attackers through its anchor score, and left detector-aware attackers only a small residual. Future work will run bound grouped SecAgg+, widen the trigger family, and evaluate other foundation forecasters and larger federations.

Supplementary Information. Online Resource 1 (ESM_1.pdf): threshold rules, public calibration, and configurations; proofs and worked examples; full notation; detailed setup and implementation; experiment grids with seeds and releases. Online Resource 2 (ESM_2.pdf): detailed results of RQ1–RQ5 (all tables, traces, replays, and cost).

Acknowledgements. The original concept and initial implementation of BurstGuard were developed at HCMC University of Technology and Engineering (HCMUTE). We sincerely thank HCMUTE for its support and for providing a conducive research environment throughout this work.

Statements and Declarations

Funding. No funding was received for conducting this study.
Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.
Ethics approval and consent. Not applicable: public, anonymised, aggregated data; attacks ran only in simulation.
Data availability. The telecommunications data of Milan and Trentino [3] are available on Harvard Dataverse under the Open Database License (https://doi.org/10.7910/DVN/EGZHFV, https://doi.org/10.7910/DVN/QLCABU). The logs of all 990 runs will be archived with a DOI together with the code.
Code availability. https://github.com/lamnguyentt92/burstguard-paper.
Use of large language models. Use of large language models. The authors used ChatGPT solely to check grammar and correct spelling errors during the preparation of the manuscript.

References

[1] Andreina S, Marson GA, Möllering H, et al (2021) BaFFLe: Backdoor detection via feedback-based federated learning. In: Proc. IEEE ICDCS, pp 852–863, https://doi.org/10.1109/ICDCS51616.2021.00086

[2] Bagdasaryan E, Veit A, Hua Y, et al (2020) How to backdoor federated learning. In: Proc. AISTATS, pp 2938–2948

[3] Barlacchi G, De Nadai M, Larcher R, et al (2015) A multi-source dataset of urban life in the city of Milan and the Province of Trentino. Scientific Data 2(1):150055. https://doi.org/10.1038/sdata.2015.55

[4] Bell J, Gascón A, Lepoint T, et al (2023) ACORN: Input validation for secure aggregation. In: Proc. USENIX Security, pp 4805–4822

[5] Bell JH, Bonawitz KA, Gascón A, et al (2020) Secure single-server aggregation with (poly)logarithmic overhead. In: Proc. ACM CCS, pp 1253–1269, https://doi.org/10.1145/3372297.3417885

[6] Beutel DJ, Topal T, Mathur A, et al (2020) Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390, https://doi.org/10.48550/arXiv.2007.14390

[7] Blanchard P, El Mhamdi EM, Guerraoui R, et al (2017) Machine learning with adversaries: Byzantine tolerant gradient descent. In: Advances in Neural Information Processing Systems (NeurIPS), pp 119–129

[8] Bonawitz K, Ivanov V, Kreuter B, et al (2017) Practical secure aggregation for privacy-preserving machine learning. In: Proc. ACM CCS, pp 1175–1191, https://doi.org/10.1145/3133956.3133982

[9] Cao X, Fang M, Liu J, et al (2021) FLTrust: Byzantine-robust federated learning via trust bootstrapping. In: Proc. NDSS, https://doi.org/10.14722/ndss.2021.24434

[10] Ekambaram V, Jati A, Dayama P, et al (2024) Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In: Advances in Neural Information Processing Systems (NeurIPS), pp 74147–74181, https://doi.org/10.52202/079017-2359

[11] Gehlhar T, Marx F, Schneider T, et al (2023) SafeFL: MPC-friendly framework for private and robust federated learning. In: Proc. IEEE Security and Privacy Workshops (SPW), Deep Learning Security and Privacy Workshop, URL https://eprint.iacr.org/2023/555

[12] Hu EJ, Shen Y, Wallis P, et al (2022) LoRA: Low-rank adaptation of large language models. In: Proc. ICLR

[13] Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. International Journal of Forecasting 22(4):679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

[14] Kotowski K, Shendy R, Nalepa J, et al (2025) Trojan horse hunt in time series forecasting for space operations. arXiv preprint arXiv:2506.01849, https://doi.org/10.48550/arXiv.2506.01849

[15] Le VT, Nguyen TTL (2026) BurstGuard: source code, experiment grids, and analysis scripts. GitHub repository, URL https://github.com/lamnguyentt92/burstguard-paper

[16] Lycklama H, Burkhalter L, Viand A, et al (2023) RoFL: Robustness of secure federated learning. In: Proc. IEEE S&P, pp 453–476, https://doi.org/10.1109/SP46215.2023.10179400

[17] McMahan B, Moore E, Ramage D, et al (2017) Communication-efficient learning of deep networks from decentralized data. In: Proc. AISTATS, pp 1273–1282

[18] Nguyen TD, Rieger P, Chen H, et al (2022) FLAME: Taming backdoors in federated learning. In: Proc. USENIX Security, Boston, MA, pp 1415–1432

[19] Pillutla K, Kakade SM, Harchaoui Z (2022) Robust aggregation for federated learning. IEEE Transactions on Signal Processing 70:1142–1154. https://doi.org/10.1109/TSP.2022.3153135

[20] Sun Z, Kairouz P, Suresh AT, et al (2019) Can you really backdoor federated learning? arXiv preprint arXiv:1911.07963, https://doi.org/10.48550/arXiv.1911.07963

[21] Wang B, Yao Y, Shan S, et al (2019) Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: Proc. IEEE S&P, pp 707–723, https://doi.org/10.1109/SP.2019.00031

[22] Xie C, Chen M, Chen PY, et al (2021) CRFL: Certifiably robust federated learning against backdoor attacks. In: Proc. ICML, pp 11372–11382

[23] Yin D, Chen Y, Ramchandran K, et al (2018) Byzantine-robust distributed learning: Towards optimal statistical rates. In: Proc. ICML, pp 5650–5659

[24] Zhang C, Dang S, Shihada B, et al (2021) Dual attention-based federated learning for wireless traffic prediction. In: Proc. IEEE INFOCOM, pp 1–10, https://doi.org/10.1109/INFOCOM42981.2021.9488883

[25] Zhang Z, Panda A, Song L, et al (2022) Neurotoxin: Durable backdoors in federated learning. In: Proc. ICML, pp 26429–26446

[26] Zhang Z, Li J, Yu S, et al (2023) SAFELearning: Secure aggregation in federated learning with backdoor detectability. IEEE Transactions on Information Forensics and Security 18:3289–3304. https://doi.org/10.1109/TIFS.2023.3280032

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- BurstGuardpaper2.pdf
- BurstGuardpaper1.pdf