HomeGuard: Community-Driven Hierarchical Federated Learning for Robust Smart-Home Intrusion Detection

Philipp Eichhammer[1], Christian Berger[2] and Hans P. Reiser[3]
[1]*University of Passau, Passau, Germany*
[2]*Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany*
[3]*Reykjavik University, Reykjavik, Iceland*

Keywords: Intrusion Detection System, Federated Learning, Smart Home, Internet of Things.

Abstract: Federated Learning (FL) has emerged as a promising approach to build collaborative Intrusion Detection Systems (IDSs) in the IoT, e.g., in smart homes. FL allows models to be shared without exposing sensitive training data, thus protecting the privacy of IoT users. However, existing FL-based IDSs rely on assumptions that rarely hold in practice, namely homogeneous devices, synchronous participation, and benign contributors. We argue that, in real-world smart homes, IoT devices are highly heterogeneous, resource-constrained, and attractive targets for adversaries, which makes conventional FL less effective or vulnerable to poisoning attacks. We present HOMEGUARD, a collaborative IDS specifically designed for the constraints and threat model of practical smart home IoT infrastructures. In our approach, we rethink FL deployment by (1) *offloading model training* to gateways to manage computational heterogeneity of IoT devices and (2) organizing anomaly detection models into *device-specific communities* based on privacy-preserving traffic fingerprints which do not expose sensitive data. Within communities and across smart homes, HOMEGUARD implements an asynchronous, hierarchical FL architecture that tolerates device churn, uneven data availability, and Byzantine participants. Further, HOMEGUARD applies *Byzantine-robust aggregation* at two levels: within local communities, and globally in the cloud to limit the impact of compromised devices. Experimental evaluation shows that HOMEGUARD achieves an average true positive rate of 97.86% locally and 97.53% globally with a 0% false positive rate, while maintaining robustness against both targeted and untargeted poisoning attacks.

1 INTRODUCTION

Intrusion detection systems (IDSs) with a focus on IoT infrastructures frequently utilize the federated learning (FL) framework for its collaborative properties that preserve privacy (Hernandez-Ramos et al., 2025; Eichhammer and Reiser, 2023; Nguyen et al., 2019). FL allows multiple clients to collaborate by sharing and learning from gained knowledge without requiring the individual, sensitive data to be shared (Konečný et al., 2017). Instead, clients only transmit privacy-preserving *model updates* to a central entity that aggregates these into a unified *global model*. Subsequently, the global model is distributed to the clients for retraining. IDSs use FL to aggregate models trained on the *benign network behavior* of observed devices, aiming to improve overall anomaly-detection accuracy. The key principle is the aggregation of trained models, which indirectly enables training on a larger dataset that encompasses the individual models' training data.

Treating client-side data confidentially while retaining the improved detection accuracy associated with using a combined, large dataset for model training seems like a natural design choice. However, the benefits of this approach depend on a set of *idealized assumptions* which may be easily violated in practical systems. First, both clients and the central entity are assumed to be benign in their actions. A single dishonest client has the potential to manipulate the global model by modifying its model updates, thus potentially compromising the models of all participating clients (Hernandez-Ramos et al., 2025). Furthermore, a malicious central entity can disrupt the entire FL process by controlling the collection and aggregation of model updates. Second, model updates are assumed to arrive within a reasonable time frame, as FL waits for all updates before aggregation. A single delayed or crashed client can delay or even block the

Eichhammer, P., Berger, C. and Reiser, H. P.
HomeGuard: Community-Driven Hierarchical Federated Learning for Robust Smart-Home Intrusion Detection.
DOI: 10.5220/0015063200004103

In *Proceedings of the 23rd International Conference on Security and Cryptography (SECRYPT 2026) - Volume 1*, pages 108-120
ISBN: 978-989-758-858-7; ISSN: 2184-7711

entire aggregation process (Cox et al., 2024). Third, the models are assumed to be trained on data that is comparable not only in structure but also in underlying content. While the structure of the training data is the same across the participating models in the FL process, different contents, as is the case with different benign network behavior of different IoT devices, can lead to an aggregated model not improving the common goal (Nguyen et al., 2019).

To address these issues, several improvements to FL frameworks have been introduced over time. Those encompass robustness against Byzantine attackers (Nguyen et al., 2022; Sattler et al., 2020; Sáez-de-Cámara et al., 2023), asynchrony of model updates (Cox et al., 2024), and clustering of clients (Eichhammer and Reiser, 2023; Nguyen et al., 2019). However, as we explain in Section 2 in more depth, current approaches typically address only parts of this design space. In particular, they often rely on assumptions that may be difficult to satisfy in practical smart-home deployments, because they either do not account for the potential lack of computational power of IoT devices (Nguyen et al., 2022; Fereidooni et al., 2023; Cox et al., 2024), assume mostly homogeneous training data (Nguyen et al., 2022; Cox et al., 2024), or do not yet fully consider uneven device distributions in the context of smart homes (Eichhammer and Reiser, 2023; Fereidooni et al., 2023; Nguyen et al., 2019). This makes existing FL approaches less suitable for use in practical IDS in smart homes.

We present HOMEGUARD, a collaborative network-based IDS for smart homes. HOMEGUARD trains the anomaly detection models and performs intrusion detection on *gateways* that connect IoT devices to the Internet. Thus, FL processing tasks are offloaded from IoT devices, allowing us to use off-the-shelf hardware. Further, HOMEGUARD clusters all available IoT devices into *communities*. Communities select devices based on their similarity in network traffic behavior. Within each community, an FL process is initiated across the models of the devices it contains. This ensures a certain degree of homogeneity in the training data and allows aggregated models to increase the intrusion detection accuracy of the community models. Moreover, HOMEGUARD also protects the FL process within each community against faulty IoT devices. Faults could be an IoT device exposing malicious training data to perform model poisoning or slowing down aggregation by delaying model updates. Finally, HOMEGUARD enables collaboration across multiple smart homes by allowing them to contribute model updates to a *replicated, cloud-based aggregator*. Through collaboration, we enable a smart home to extend its locally trained models with knowledge from other smart homes. This allows both *extending existing communities* and *creating virtual*, i.e., global-spanning communities for devices that are not included in a local community, so they can also benefit from FL. HOMEGUARD also protects the FL process within each global-spanning community against faulty smart home gateways.

The remainder of this paper is structured as follows: Section 2 outlines the smart home IoT challenges motivating HOMEGUARD; Section 3 describes the system and threat model; Section 4 details our IDS approach; Section 5 presents the evaluation; and Sections 6 and 7 cover related work and conclusions.

2 MOTIVATION

To begin, we briefly motivate the design of HOMEGUARD by reviewing the challenges typically encountered in IoT infrastructures. In IoT infrastructures, and especially in smart homes, a wide range of devices coexist, often in very uneven numbers across device types (Batalla et al., 2017). This heterogeneity creates several challenges for FL:

Ⓒ1 **IoT Device Capabilities.** Existing FL approaches assume that clients train and execute their machine learning models (Konečný et al., 2017; Nguyen et al., 2022; Fereidooni et al., 2023; Cox et al., 2024). If we were to apply this concept to the IoT infrastructure, the IoT devices, in addition to executing their designated tasks, would *act as clients* within the FL infrastructure. Thus, IoT devices are assumed to manage data collection, model training, and model execution. However, the landscape of IoT devices utilized in smart home IoT infrastructures is broad, ranging from single-purpose, low-powered environmental sensors and actuators, e.g., temperature sensors or light switches, to multi-purpose, computation-capable smart devices, such as smart TVs (Batalla et al., 2017). We argue that processing capabilities and vendor-specific, closed-source software strongly limit the extensibility of devices to *make them clients* (which need to perform custom machine-learning processing). As a consequence, installing and integrating standard IoT devices requires greater effort (see (Wu et al., 2022)) and would severely limit the selection of suitable devices in any real-world setting.

Ⓒ2 **Model Diversity.** The heterogeneity of IoT devices further causes the problem of unadjusted aggregated global models (Nguyen et al., 2019). IoT devices exhibit distinct network traffic patterns that require customized anomaly detection models. If the FL infrastructure *aggregates across all* anomaly de-

tection models, then the resulting global model may not precisely capture aspects of the individual models. This would lead to a reduced detection accuracy of individual models if the global model were distributed. Yet, this problem remains widely unaddressed in existing systems (Nguyen et al., 2022; Fereidooni et al., 2023; Cox et al., 2024), which omit the potential of model accuracy improvement in real-world deployments, thus creating a research gap.

(C3) **Disparity in IoT Device Counts.** The numbers of IoT devices within the different *classes* of IoT devices may vary by the order of magnitudes (Batalla et al., 2017). For instance, environmental sensors, such as temperature sensors, may be installed in every room of a smart home, whereas other devices, such as a smart door opener, may be installed only once. This discrepancy, in combination with device-specific FL infrastructures, could result in the underrepresentation of certain device classes, thereby excluding them from model aggregation. For example, there could be only one smart door lock in a smart home, which, given its network traffic behavior, would not have any other models to aggregate with (Eichhammer and Reiser, 2023).

(C4) **Data Availability Imbalance.** The *quantity of network traffic data* that can be collected by observation within a certain time frame also varies. For instance, low-power single-purpose sensors and actuators are likely to expose less traffic than a multi-purpose multi-media device. However, to train accurate anomaly detection models, a certain size of training data is required. Classical FL approaches perform model aggregation when all model updates are received (Konečný et al., 2017). This creates a bottleneck, as aggregation must wait for the device with the fewest data records to reach a specified threshold before training the model. Yet, most existing approaches assume such a synchronous update of model information within an IoT infrastructure (Nguyen et al., 2022; Fereidooni et al., 2023; Nguyen et al., 2019; Eichhammer and Reiser, 2023).

3 SYSTEM AND THREAT MODEL

In this section, we present our system model and introduce the threat model on which HOMEGUARD operates.

3.1 System Model

We consider a three-tier IoT architecture consisting of *devices*, *smart-home gateways*, and a *cloud layer*. Each smart home contains exactly one gateway and a non-empty set of IoT devices. Every device is attached to exactly one gateway, while a gateway may manage multiple devices. Devices may join and leave over time. Gateways observe the traffic metadata of their attached devices and are assumed to have sufficient compute and storage resources to train local models and participate in FL. HOMEGUARD organizes learning hierarchically into *local communities* and *global communities*. A local community groups device-level models within a smart home, while a global community groups smart-home contributions at the cloud layer.

Smart Homes and Devices. Let $H = \{h_1, \ldots, h_n\}$ denote the set of smart homes. We assume each smart home $h \in H$ has exactly one gateway g_h and a set of attached devices D_h.

Local Communities. A local community lc is formed from device-level models of similar device classes within a smart home. Let

$$M(lc) = \{m_1, \ldots, m_{n_{lc}}\}$$

be the set of participating device-level models in the local community lc. We assume that each local community tolerates up to f_{lc} Byzantine participants and satisfies

$$n_{lc} \geq 3f_{lc} + 1.$$

of IoT devices whose corresponding models contribute to the local community lc. Thus, a local community lc is considered *correct* if at most f_{lc} of its participating IoT devices are Byzantine. From a *liveness* perspective, the FL process in a local community lc can always be initiated. This happens as soon as $2f_{lc} + 1$ trained models become available, of which a majority originate from correct devices.

Global Communities. For a global community c let

$$H_c \subseteq H$$

denote the set of smart homes contributing to c. Each smart home contributes at most one input to a given global community, namely either (i) the aggregated output of one of its local communities, or (ii) the model of a single standalone device that does not belong to any local community. For every c, we assume

$$|H_c| \geq 3f_c + 1,$$

where at most f_c contributing smart homes are faulty. Hence, at least $2f_c + 1$ contributions in every global community c originate from non-faulty smart homes.

Fault Model. We define faults with respect to a specific global community c. A smart home $h \in H_c$ is considered faulty with respect to c if at least one of the following conditions holds:

1. *Faulty gateway.* the gateway g_h is Byzantine.

2. *Faulty local-community contribution.* if the contribution of h to c is derived from a local community lc, then at least $f_{lc}+1$ of the $2f_{lc}+1$ participating model inputs in lc are poisoned (i.e., contributed by Byzantine devices).
3. *Faulty standalone-device contribution.* if the contribution of h to c is the model of a single device not belonging to a local community, and this device is Byzantine.

Let $\mathsf{Faulty}_c(h)$ denote the predicate that smart home h is faulty with respect to the global community c:

$$\mathsf{Faulty}_c(h) \equiv \mathsf{GatewayFaulty}(h) \\ \vee \big(\exists lc \text{ contributed by } h : |\mathsf{Byz}(lc)| \geq f_{lc}+1\big) \\ \vee \big(\exists d \text{ contributed by } h : \mathsf{Standalone}(d) \wedge \mathsf{Byz}(d)\big).$$

Accordingly, the global fault assumption is

$$\forall c : \quad |\{h \in H_c \mid \mathsf{Faulty}_c(h)\}| \leq f_c.$$

This model captures the idea that the cloud layer reasons about smart-home contributions rather than individual devices directly. A smart home is considered faulty for a global community as soon as the contribution it sends to that community can no longer be trusted: if the gateway is Byzantine, if the local aggregation producing the contribution is already corrupted by a Byzantine majority, or if the contribution is based on a single Byzantine standalone device.

3.2 Threat Model

In FL, poisoning attacks (Xia et al., 2023) aim to corrupt the aggregated model, causing incorrect predictions. In the context of an anomaly-based IDS, these attacks are either untargeted — disrupting normal system operation by generating erroneous alerts — or targeted, where malicious traffic is injected into training data to prevent the model from detecting a subsequent attack (Xia et al., 2023). Poisoning can further be carried out by manipulating training data (data poisoning) or model weights directly (model poisoning). HomeGuard accounts for all of these attack variants *(targeted and untargeted, data and model poisoning)*.

4 HomeGuard

In this section, we first provide an overview of our system architecture, followed by an in-depth description of our community strategy and its integration into an asynchronous hierarchical FL infrastructure.

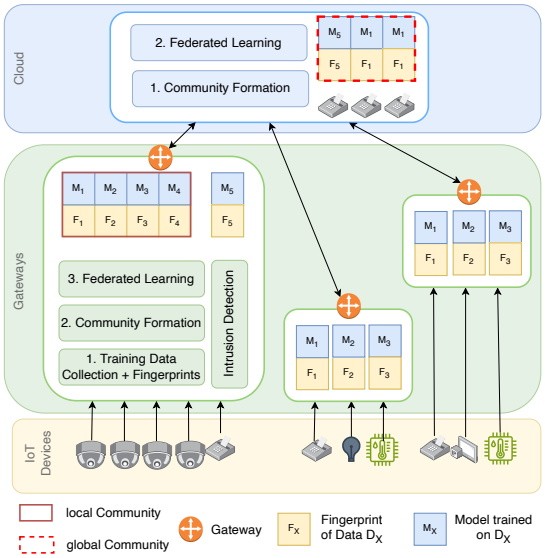

Figure 1: The architecture of HomeGuard. The main idea is to offload model training and intrusion detection from IoT devices to gateways.

4.1 Architecture Overview

HomeGuard is an intrusion detection system that is optimized for use in smart home IoT infrastructures, as shown in Figure 1. It is capable of detecting anomalies in the network traffic behavior of IoT devices while tolerating malicious or crashed devices and gateways. Further, it enables a privacy-preserving exchange of trained knowledge across smart home infrastructures. Thereby, HomeGuard aims to solve challenges and issues with smart home IoT infrastructures (see Section 2), by achieving the following goals:

- (G1) **High Intrusion Detection Accuracy.** The intrusion detection system achieves high detection accuracy. This means the *false positive rate (FPR)* has to be as low as possible, while maintaining a high *true positive rate (TPR)*.
- (G2) **Real-World Applicability.** The system is aware of the challenges and limitations of the infrastructure on which it is deployed. As a result, it provides features to address the challenges mentioned in Sect. 2, i.e., IoT device heterogeneity and capabilities, data availability, and model diversity.
- (G3) **Byzantine Robustness.** Our IDS has to be robust against Byzantine faulty actors within the system. Here, compromised IoT devices pose a potential threat, and compromised gateways can expose harmful behavior, hindering the system from achieving its goals.

Overview. The operating principle of HOMEGUARD focuses on the gateways and the cloud layer. Starting with IoT devices, they communicate via gateways to other devices and services on the Internet. For each IoT device, the corresponding gateway records the network traffic, and a fingerprint of this traffic is created for later community formation (see Sect. 4.2). This offloads the training and execution of anomaly-detection machine-learning models from IoT devices to gateways, enabling the use of off-the-shelf hardware and addressing the challenge of IoT device capabilities (C1).

Once a sufficient amount of data has been collected (see Section 4.2), a machine learning model is trained on that data to learn the normal network traffic behavior of the IoT device. Using this model, behavioral anomalies can be detected. This is done for every IoT device in the system.

Furthermore, HOMEGUARD is built on three fundamental concepts, each addressing a different aspect of enhancing FL for a secure and effective deployment in an IDS for smart home IoT infrastructures. Those concepts are: (1) asynchronous community formation; (2) Byzantine-robust federated learning; (3) hierarchical infrastructure. At a high level, the interconnections among the concepts are as follows: community formation enables our FL framework to build a separate FL process for each community. A community comprises models trained on similar data, thereby optimally enhancing the aggregated model properties of the contributing models while coping with model diversity, as described by challenge (C2). Within the communities, we perform asynchronous Byzantine-robust model aggregation, thereby strengthening our IDS against poisoning attacks originating from compromised IoT devices. Furthermore, this method handles delayed model updates, thereby enabling robustness to blocking aggregation, addressing challenge (C4). Finally, the hierarchical structure of our IDS allows for an inter-gateway FL infrastructure within global communities. This enables smart home infrastructures to enhance their local models by exchanging privacy-preserving model updates with other smart homes. Underrepresented IoT devices within a single gateway that cannot form a local community could still be part of a global community and benefit from the findings of others through FL, addressing challenge (C3).

Intrusion Detection Mechanism. The intrusion detection mechanism in HOMEGUARD uses machine learning models trained on normal network traffic behavior of IoT devices. For each device, a separate model is trained. Those models are then used to detect anomalies in network behavior and raise alerts.

The machine learning model we use is a Gated Recurrent Unit (GRU). Their application for anomaly detection in network intrusion works as follows. A packet is classified as anomalous if its predicted probability p_i is below a threshold $\delta = 0.01$. To minimize false positives, an alert is triggered only if the proportion of anomalous packets in a consecutive set W exceeds a threshold $\gamma = 0.5$ and $w = |W|$:

$$\frac{|\{pkt_i \in W | p_i < \delta\}|}{w} > \gamma$$

The principle behind the utilization of GRU is to model and learn probabilistic dependencies among sequential symbols, thereby capturing the likelihood of one symbol following another. If, after the model is trained, the arriving symbols differ, an anomaly is detected and, potentially, an alarm is raised.

4.2 Core Features

The following features represent the main contributions of HOMEGUARD, which can be divided into the hierarchical structure, the asynchronous community formation, and asynchronous Byzantine-bobust FL.

4.2.1 Hierarchical Structure

The hierarchical structure of HOMEGUARD enables the use of FL both within and across gateways. It embeds the Byzantine-robust FL processes executed within the communities into two separate layers. The local smart home layer, comprising one gateway and several IoT devices that form local communities, and the global inter-smart home cloud layer, which connects different smart home infrastructures to form global communities and exchange model information.

Extending the model exchange across several smart homes can increase intrusion-detection accuracy, especially for models of IoT devices that are often underrepresented in single smart homes. As an example, a smart home might have only a single entrance door equipped with a smart door lock. A global community across multiple door locks across several smart home infrastructures can potentially increase detection accuracy through model information exchange. Algorithms 1 & 2 depict the main process lifecycles of the gateway and cloud layers.

To protect inter-smart-home communication from cloud-layer crash faults, HOMEGUARD employs $f_{ci}+1$ instances operating in a hot-standby configuration, tolerating up to f_{ci} crashed instances. One computing instance receives and processes all requests from the gateways, while the other instances are activated only when the main instance crashes. We assume that a

sufficiently secure cloud computing instance is hard to attack, hence, we focus on crash faults at this layer.

Algorithm 1: HomeGuard – Gateway.

Input: Gateway g_h managing smart home h; set of IoT devices D_h

foreach *device* $d \in D_h$ **do**
  $traffic \leftarrow$ ObserveTraffic(d);
  if $recordingTime(traffic) \geq 3$ *hours* **then**
    $fingerprint[d] \leftarrow$ ExtractFingerprint($traffic$);
    $model[d] \leftarrow$ TrainGRU($traffic$);
    CommunityFormation($model[d], fingerprint[d]$);

CommunityFormation($\{models\}$, $\{fingerprints\}$) :
if $|models| \geq 2f_d + 1$ **then**
  $communities \leftarrow$ HDBSCAN$(\{fingerprints\})$;
  foreach *community* $C \in communities$ **do**
    if $|C| < 2f_{lc} + 1$ **then**
      $\{stdModels\} \leftarrow C$;
    else
      $\{aggModels\} \leftarrow$ ByzRobustFL$(C, \{models\})$
  SendToCloud($aggModels, stdModels, fingerprints$);

Algorithm 2: HomeGuard – Cloud.

Input: Set of Gateways $G = \{g_1, ..., g_n\}$

// aggModels: aggregated models; stdModels: standalone models

$\{aggModels\}, \{stdModels\} \leftarrow$ ReceiveModels(G);
$\{F_{agg}\}, \{F_{std}\} \leftarrow$ ReceiveFingerprints(G);
if $|\{aggModels\}| + |\{stdModels\}| \geq 2f_c + 1$ **then**
  $\{communities\} \leftarrow$ HDBSCAN$(\{F_{agg}\}, \{F_{std}\})$;
  $\{communities\} \leftarrow$ AggregateStandalone$(\{communities\}, \{stdModels\})$
  foreach *community* $C \in \{communities\}$ **do**
    if $|C| < 2f_c + 1$ **then**
      label C as *standalone*;
    else
      $\{aggModelsCloud\} \leftarrow$ ByzRobustFL$(C, \{aggModels\}, \{stdModels\})$
    SendBackToGateways($\{aggModelsCloud\}$);

4.2.2 Asynchronous Community Formation

Community formation groups device-level models so that each community contains only models trained on similar network traffic. Restricting aggregation to closely related models improves mutual information sharing and detection accuracy (Eichhammer and Reiser, 2023; Nguyen et al., 2019), which is particularly important in smart home IoT environments where device traffic patterns vary widely.

Fingerprinting. To measure similarity without exposing raw traffic data, HomeGuard extracts a privacy-preserving fingerprint from each device's training dataset. The fingerprint captures the statistical structure of traffic via five features: packet-size distributions, inter-arrival time distributions, direction statistics, protocol ratios, and TCP flag ratios (Table 1). These features distinguish broad device classes by capturing network behavior rather than concrete communication. They were carefully selected and are aligned with related work, enabling a precise representation of IoT device network traffic (Nguyen et al., 2019).

Formation Trigger. Community formation is initiated asynchronously, i.e., it does not wait for all devices to finish training before proceeding. Let n_d be the number of IoT devices managed by a gateway, and let f_d be the maximum number of faulty devices tolerated, with $n_d \geq 3f_d + 1$. Formation is triggered as soon as $2f_d + 1$ trained models become available at the gateway, regardless of whether the remaining f_d devices have finished. This threshold is chosen for two reasons. First, waiting for all n_d models would allow a single crashed or slow device to indefinitely delay formation, recreating the synchronous bottleneck of classical FL and endangering liveness. Second, the Byzantine-robust aggregation step (Section 4.2.3) requires at least $2f + 1$ participants to guarantee that a majority of inputs come from honest devices, ensuring safety. Triggering at exactly this threshold satisfies both liveness and safety requirements simultaneously.

Models that complete training after formation has already been triggered are not discarded. They are held in a pending queue and included in the next formation round, which is triggered again once $2f_d + 1$ new or pending models have accumulated. This means community structure is not permanently fixed: each round may refine communities as more devices become available or as devices that previously crashed recover and submit updated models.

Formation Mechanism. As shown in Algo. 1, formation clusters the fingerprint vectors $\{fingerprints\}$ of completed models using HDBSCAN, which handles arbitrary cluster shapes and marks isolated points as noise, which is well-suited to the uneven device distribution in real smart homes (McInnes and Healy, 2017). Each resulting cluster becomes a candidate community, promoted to a full community only if it contains at least $2f_{lc} + 1$ members. Candidates below this threshold are labeled standalone and forwarded to the cloud to join a global community with counterpart devices from other homes.

Fingerprint Aggregation. After local FL, the gateway produces both an aggregated model and an ag-

Table 1: Features included in the fingerprint.

Feature	Explanation	#Features
Packet-size histogram	Distribution of packet sizes	10
IAT histogram	Distribution of inter-arrival times (IAT)	8
Direction statistics	Relationship between in- and outgoing packets	2
Protocol ratios	Share of TCP, UDP, ICMP, and other protocols	4
TCP flag ratios	Share of TCP flags	6

gregated fingerprint, the feature-wise mean of the non-discarded participants. This fingerprint is sent to the cloud alongside the model, enabling the same HDBSCAN-based formation at the inter-gateway level without exposing any raw traffic data.

Global Formation. At the cloud layer, the same logic applies. Let n_g be the number of gateways and f_g the tolerated faulty gateways, with $n_g \geq 3f_g + 1$. Global formation triggers once $2f_g + 1$ gateway contributions arrive, each consisting of either a local community's aggregated model and fingerprint, or a standalone device's individual model and fingerprint. Where a single gateway has multiple standalone models of the same device type, these are pre-aggregated at the gateway to prevent dominance. The cloud then clusters all fingerprints using HDBSCAN, applies Byzantine-robust FL within each global community, and returns the resulting models to the contributing gateways.

4.2.3 Asynchronous Byzantine-Robust FL

To make FL robust against Byzantine attackers, we employ Catalyst (Cox et al., 2024) within our communities, a technique ensuring Byzantine robustness under the assumption that models are delayed or crash. To the best of our knowledge, this is the first integration of Catalyst into a community-based multi-FL infrastructure. For detailed information, we refer to Cox et al. (Cox et al., 2024).

Catalyst requires communities to consist of at least $2f + 1$ models, tolerating up to f faulty or malicious IoT devices. This threshold guarantees that at least $f + 1$ updates originate from honest devices, enabling malicious updates to be statistically identified and excluded before aggregation (Cox et al., 2024). We ensure this by forming communities to that specification, while also resolving Catalyst's liveness issue in the edge case of exactly $2f + 1$ participants. In this case, Catalyst cannot tolerate crashed clients that do not send model updates, thus it acts like a synchronous FL process. By assuming $3f + 1$ IoT devices, we can tolerate up to f crashed devices and still guarantee $2f + 1$ participants for Catalyst.

Robustness is achieved through a multi-stage filtering and aggregation procedure. First, gradient clipping is applied: the gateway computes the distance between each model update and the current global model, derives a clipping bound from the median of these distances, and scales updates accordingly, preventing adversarially large gradients from dominating aggregation (Nguyen et al., 2022; Cox et al., 2024). Subsequently, pairwise cosine distances between updates are computed, and updates are clustered. Since honest models are expected to produce similar gradients, the largest cluster is assumed benign; outlying updates are discarded. The remaining updates are averaged to produce a robust global model update. This combination tolerates arbitrary malicious gradients, provided Byzantine devices do not exceed f.

To support asynchronous training, HOMEGUARD maintains multiple versioned global models via Catalyst. Once $2f + 1$ updates for the latest version are collected, a new global model version is computed. Delayed updates within a predefined version window are incorporated by clustering them with previously accepted updates from the same version. To prevent outdated updates from destabilizing training, a staleness weighting function reduces their influence proportional to their age, while also scaling their contribution relative to the total community size. Together, the clustering-based robust aggregation, threshold-triggered update mechanism, and staleness-aware integration of delayed updates enable a FL strategy that is both Byzantine-resilient and fully asynchronous.

5 EVALUATION

In this section, we evaluate HOMEGUARD's usefulness for smart home IoT infrastructures using a real-world dataset containing IoT malware. Our evaluation is guided by three questions: (1) How effectively does our system detect intrusions in real-world smart home IoT environments? (2) How well does our community formation strategy align with existing methods? (3) Can our approach successfully tolerate poisoning attacks? We conduct our experiments using the Python programming language in combination with the PyTorch framework for machine learning tasks.

5.1 Evaluation Setup

This section introduces the dataset used for evaluation, as well as our strategy to create data poisoning attacks. Further, the metrics for the evaluation are explained.

5.1.1 Dataset

To compare our approach with related work, we use the DIoT dataset, a de facto standard in community-based FL research for IoT environments. Initially proposed by Nguyen et al. (Nguyen et al., 2019) and subsequently used in their model-poisoning research (Nguyen et al., 2022), it reflects realistic IoT network traffic and attacks targeting specific devices. Its explicit association of traffic with individual IoT devices enables precise control over attacks and insights into our detection mechanism. The dataset comprises both benign and malicious traffic, with malicious data generated via the Mirai attack. It includes 33 IoT devices, of which only five have associated malicious traffic. Of the 15 devices deployed in a realistic smart-home environment, attack data is available for four. This smart-home subset (15 devices) forms the primary basis for our evaluation.

5.1.2 Poisoning Attacks

To inject targeted poisoned samples into the training data, we use the label-free contamination strategy proposed by Nguyen et al. (Nguyen et al., 2020). It introduces adversarial samples into the training set at random positions according to a predetermined Poisoned Data Rate (*PDR*), defined as:

$$PDR = \frac{|D_A|}{|D_M|}$$

where D_A denotes the set of adversarial samples and D_M denotes the benign training set of a given device M.

Consistent with prior work (Nguyen et al., 2020), a *PDR* of 35% has been shown to represent an effective trade-off on average: it is low enough to evade regular anomaly detection without poisoning detection, yet high enough to degrade model integrity.

The generation of untargeted poisoning data is similar. Randomly generated data samples are injected into the benign training data. However, the injection ratio is considerably higher than in the targeted case, since the goal of an untargeted attack is to degrade the classification performance of the aggregated model. Aligned with related work (Nguyen et al., 2020), we set the *PDR* to 70%, since the affected device will also receive benign network traffic.

5.1.3 Evaluation Metrics

The primary interest in evaluating an IDS lies in its ability to correctly detect intrusions without mislabeling benign traffic. Hence, we utilize the *false positive rate (FPR)* and the *true positive rate (TPR)*. The *FPR* is the ratio of traffic falsely classified as intrusions, resulting in false alarms. The *TPR* provides the ratio of correctly classified anomalies. The goal is to minimize the *FPR*, not to overwhelm the operator with false alarms, while maximizing the *TPR* for as many intrusions as possible to be detected.

5.2 Experimental Results

The experimental results are introduced in the following subsections with a focus on intrusion detection accuracy, community formation comparison and Byzantine robustness evaluation.

5.2.1 Intrusion Detection Accuracy

To evaluate intrusion-detection accuracy, we divided the evaluation into two scenarios. First, we focus on the fundamental capabilities of our intrusion detection approach, including local community formation and detection accuracy, in the context of a single smart home IoT infrastructure. Second, we evaluate the effectiveness of global communities across smart homes in terms of intrusion-detection accuracy.

Setup. The intrusion detection accuracy is evaluated using the subset of devices (4 out of 15) for which the DIoT dataset includes attack data (DLinkCam 930, DLinkCam 932, EdimaxPlug 1101, EdimaxPlug 2101). The results are then also compared to prior work (Eichhammer and Reiser, 2023; Nguyen et al., 2019). Local models are trained for 17 epochs across three community formation rounds (for a total of 51 epochs), following the configuration used in related work to ensure comparability. We evaluate detection accuracy across four Mirai-based attack phases: *DoS*, *Scan*, *Pre-Infection*, and *Infection*.

Local Detection Accuracy. The results for our intrusion detection accuracy within one smart home, shown in Table 2, indicate strong detection capabilities. Both *DoS* and *Scan* phases achieve near-perfect true positive rates (TPR) and no false positives, with TPRs of over 99.3%. Our approach attains a higher DoS detection rate than the comparable baseline of 88.96%. *Infection* is the most difficult phase to detect (94.28% local, 93.14% global), while *Pre-Infection* remains above 98%.

These results demonstrate that our system performs competitively with state-of-the-art intrusion detection systems in federated IoT environments. Moreover, as already evaluated by (Nguyen et al., 2019), the high TPR and 0% FPR demonstrate the advantage of an IoT device-type FL approach over a single FL process across heterogeneous IoT devices in smart home IoT infrastructures.

Table 2: Comparison of our approach with related work.

	HOMEGUARD		Related Work		
Attack	local	global	(1)	(2)	All
	TPR	TPR	TPR	TPR	FPR
DoS	99.33%	99.25%	-	88.96%	0%
Infection	94.28%	93.14%	-	93.45%	0%
Pre-Infection	98.11%	98.20%	-	100%	0%
Scan	99.72%	99.53%	-	100%	0%
Average	97.86%	97.53%	97.50%	95.60%	0%

(1) = (Eichhammer and Reiser, 2023), (2) = (Nguyen et al., 2019).
(1) provides only average TPR. FPR is 0 for all approaches (combined column).

Global Detection Accuracy. The results of intrusion detection accuracy across several smart homes utilizing global communities, shown in Table 2, are closely related to the local scenario and indicate strong detection capabilities across smart homes. For that, we distributed the individual IoT devices that would form a global community across several gateways and recorded the detection accuracy after the global model aggregation instead of local aggregation. We observe comparable high TPR values across all four attack types, with *DoS*, *Infection*, and *Scan* slightly lower than in the local scenario, whereas *Pre-infection* yields slightly higher results. However, all of this can be explained by slight variations in model training and by its non-deterministic nature. This also demonstrates that the individual models distributed across several gateways benefit from global community formation in terms of intrusion detection accuracy.

5.2.2 Community Formation

We evaluate local and global community formation using data from 15 IoT devices, comparing the clusters formed by our approach with those identified in prior work (Eichhammer and Reiser, 2023; Nguyen et al., 2019).

Formation Quality. Our community formation approach yields comparable results with those reported in related work (Nguyen et al., 2019; Eichhammer and Reiser, 2023), as shown in Table 3. These results are equivalent and stable over longer periods of time for both local and global community formation. The only notable difference between HOMEGUARD and prior work is observed in the grouping of smart plugs, a light switch, and a light gateway. Our method does not group the light switch and the light bridge, whereas (Eichhammer and Reiser, 2023) clusters both devices into a single community. Although our formation strategy differs from (Nguyen et al., 2019), which uses an external clustering approach that returns a fixed community structure, we share a common community structure across all IoT devices. The clustering remains stable throughout the experiment

Table 3: Communities formed by HOMEGUARD (1) compared to the communities formed by related approaches.

Identifier	**Description**	(1)	(2)	(3)
Amazon Echo	Smart Speaker	NC	NC	NC
DLink Cam 930	IP Camera (traffic-heavy)	#1	#1	#1
DLink Cam 932	IP Camera (traffic-heavy)	#1	#1	#1
DLink Switch	Smart Plug (single-purpose)	#2	#2	#2
DLink Sensor	Motion Sensor (single-purpose)	#2	#2	#2
EdimaxPlug 1101	Smart Plug (single-purpose)	#3	#3	#3
EdimaxPlug 2101	Smart Plug (single-purpose)	#3	#3	#3
Ednet Gateway	IoT Gateway (multi-purpose)	NC	NC	NC
Google Home	Smart Speaker (multi-purpose)	NC	NC	NC
Hue Switch	Light Switch (single-purpose)	NC	NC	#4
Lightify	Light Bridge (multi-purpose)	NC	NC	#4
Netatmo	Weather Station (single-purpose)	NC	NC	NC
iKettle	Smart Kettle (single-purpose)	NC	NC	NC
Wemo Insight Switch	Light Switch (single-purpose)	#6	#6	#6
Wemo Switch	Light Switch (single-purpose)	#6	#6	#6

(1) = HOMEGUARD, (2) = (Nguyen et al., 2019),
(3) = (Eichhammer and Reiser, 2023).

and supports the formation of semantically coherent device communities.

Since our local and global community formation is equivalent, we conclude that the fingerprint aggregation, performed in conjunction with model aggregation and explained in Section 4.2, correctly represents the aggregated devices. The formation as presented in the Table 3 was created after the first community formation execution. To do so, all IoT devices were started simultaneously to establish a baseline for community formation and to compare with prior work.

Asynchronous Formation. To evaluate formation in asynchronous settings, we took $2f$ IoT devices (EdimaxPlugs) that form a community and specifically started one device after the initial system startup to simulate joining devices. For all our tests, regardless of the entry point of the straggling devices, the community formations eventually resulted in the same communities as in our synchronous evaluation. Table 4 shows the results. We observe that once the missing device for formation joins the system, the community eventually forms. For one test, we chose to join the missing device in the middle of the training data recording session. This leads to forming the community in the consecutive round when the IoT device completes three hours of training data. The same experiments conducted with gateways and the cloud layer resulted in the same results. It demonstrates that our asynchronous community formation can handle the asynchronous nature of IoT devices, which contributes to our goal (G2) of increasing the real-world applicability of HOMEGUARD for smart homes.

Formation Robustness. To evaluate the influence of targeted and untargeted data poisoning attacks and

Table 4: Asynchronous community formation under varying join times (left) and poisoning scenarios (right).

Community	0h	3h	4.5h
EdimaxPlug	#1	#2	#4
DLink Cam	#1	#2	#4

(a) Community formation round by join time of the missing device after HOMEGUARD started.

Community	Benign	Targeted	Untargeted
Edimax Plug	#1	#1	#1
DLink Cam	#1	#1	#1

(b) Community formation round under benign and poisoned conditions.

Table 5: Byzantine Robustness Evaluation Results.

Attack Target	Scenario	TPR	FPR
IoT Device	Crash	97.50%	0%
	Poison (targeted)	97.40%	0%
	Poison (untargeted)	97.30%	0%
	$> f$ (targ. & untarg.)	0%	0%
Gateway	Crash	97.30%	0%
	Poison (targeted)	97.00%	0%
	Poison (untargeted)	97.20%	0%
	$> f$ (targ. & untarg.)	0%	93.40%

the fingerprints they exhibit, we again used IoT devices forming communities and simulated poisoning attacks. We used $3f+1$ IoT devices, as required by our approach, and simulated f of them as poisoned devices. Table 4 shows the results of our tests, providing evidence for the effectiveness of HOMEGUARD in ensuring liveness for community formation even in the presence of up to f Byzantine IoT devices.

5.2.3 Byzantine Robustness

To evaluate the robustness of HOMEGUARD against Byzantine actors, we create two different scenarios: (1) Simulating Byzantine IoT devices and (2) simulating Byzantine gateways. For a Byzantine IoT device, we adopt the adversary's perspective and poison the training data in targeted or untargeted ways. Additionally, we test crashed IoT devices and investigate their influence on community formation. Byzantine gateways are simulated from a model-poisoning perspective in either targeted or untargeted settings. Further, crashed gateways and manipulated fingerprints are simulated. We rely on Cox et al. (Cox et al., 2024) for the correctness of the aggregation primitive, and evaluate how HOMEGUARD behaves end-to-end under Byzantine conditions at the community and gateway level.

Byzantine IoT Devices. The evaluation setup consists of four IoT devices forming a single community, as determined by prior experiments. We use EdimaxPlug devices and simulate $f = 1$ faulty device, covering both crash and data poisoning failure modes. We measure TPR and FPR across all four attack types.

For crashed IoT devices, there are two scenarios. If the crash occurs before the device participates in traffic recording, no training data are collected, and community formation proceeds unaffected under the standard assumption that at most f devices are faulty. If the crash occurs during the recording phase, the partial training data are used. Depending on the data volume collected before the crash, the device may be clustered into the correct community or excluded. In either case, the device is correctly reintegrated at the latest during the subsequent community formation cycle, which runs every three hours.

For IoT devices executing untargeted poisoning attacks, the difference between the poisoned and benign fingerprints consistently excludes the malicious device from community formation. Crucially, this exclusion does not degrade detection performance: as shown in Table 5, TPR and FPR remain virtually identical to the crash scenario. For targeted poisoning attacks, any device that evades community-level exclusion is subsequently rejected by Catalyst, which prevents the attack from influencing the FL aggregation. Intrusion detection accuracy is therefore unaffected.

As a stress test, we also evaluate the scenario in which the number of faulty devices exceeds the fault-tolerance bound f. Once more than f malicious devices are admitted into a community, TPR drops to 0% for both attack types, as the poisoned updates dominate the FL aggregation. An interesting finding is that because Byzantine IoT devices cannot forge their fingerprints, untargeted attacks cannot be forced into a specific community and therefore do not increase FPR, even when the fault bound is exceeded.

Byzantine Gateways. The gateway evaluation mirrors the IoT setup, with $3f+1=4$ gateways, of which $f = 1$ is Byzantine. Each gateway contributes one model and its corresponding fingerprint, backed by a single honest IoT device providing local training data. The datasets used are identical to those in the IoT experiments.

Unlike crashed IoT devices, a crashed gateway is immediately detectable because it sends no model updates. Under our $3f+1$ gateway model, FL aggre-

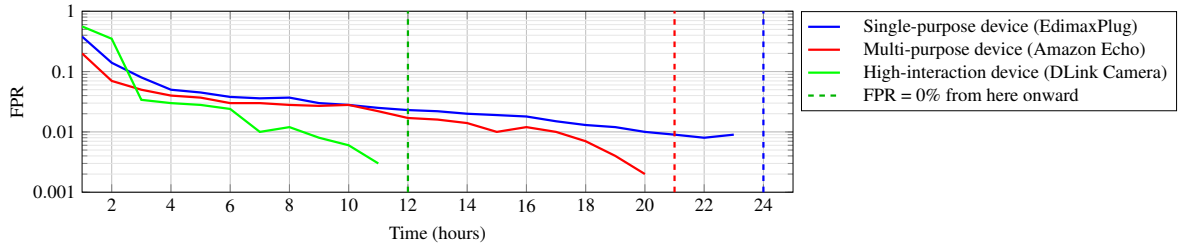

Figure 2: The false positive rate (FPR) in relation to the recording time of the training data.

gation and community formation proceed as soon as $2f+1$ gateways submit their updates. Crashed gateways therefore have no impact on community formation or detection accuracy, as confirmed in Table 5.

Byzantine gateways can mount both targeted and untargeted poisoning attacks. In contrast to IoT-level adversaries, a malicious gateway can also manipulate its fingerprint, enabling it to target a community of its choice. In this evaluation, we assume the adversary does so. Table 5 shows that, provided no more than f gateways are faulty, neither attack type degrades TPR or FPR. When the fault bound is exceeded, however, detection performance collapses entirely, consistent with the results from IoT devices.

5.2.4 Training Data Size

To determine the appropriate community formation interval and training data size, we analyzed the mean hourly communication volume per device and identified three categories: single-purpose (0–500 packets per hour (p/h)), multi-purpose (501–2000 p/h), and high-interaction (above 2000 p/h). For each category, we trained models on data collected in one-hour increments. As shown in Figure 2, the FPR drops sharply within the first few hours across all devices and reaches zero by hour 24 at the latest. Single-purpose devices exhibit higher initial FPR than multi-purpose ones, likely due to their lower traffic volume early on. High-interaction devices (mainly cameras) start with comparatively high FPR but decline rapidly after three hours, likely because cameras only generate high interaction traffic upon detecting activity rather than streaming continuously. Based on these findings, we set the training data duration to 3 hours, as this yields consistently low FPR across all device categories of this dataset. Combined with our FL strategy, which aggregates models from multiple devices within the same community, each individual device only needs to contribute a small amount of training data, e.g., 3 hours if a community contains 7 devices.

Note that the 3 hour threshold is empirically derived from the DIoT dataset and should not be understood as a universal constant. Deployments dominated by continuously streaming multimedia devices or, conversely, by ultra-low-power sensors with sparse traffic would likely shift the point at which the FPR stabilizes. A more sophisticated alternative, which we leave for future work, is to replace the static duration with an adaptive per-device stopping criterion.

6 RELATED WORK

Related work can be broadly categorized into two areas: (1) approaches that focus on community formation in benign and malicious environments, and (2) approaches that address asynchronous, Byzantine-robust, and hierarchical FL. While the former typically focuses solely on detection accuracy by grouping participants based on similarity, the latter are primarily concerned with detecting and tolerating poisoning attacks in FL environments.

Community-Based IDSs. (Cordero et al., 2015) first introduces community-based distributed IDSs, partitioning IDS entities into subsets within a centralized architecture and applying ensemble learning to enhance detection. HOMEGUARD builds on this by incorporating FL within communities, but replaces random group assignments with a dynamic, similarity-based formation strategy suited to FL.

(Nguyen et al., 2019) proposes DIoT, which groups IoT devices into communities using an external categorization tool and runs per-community FL to improve detection accuracy. HOMEGUARD improves on DIoT in two ways: community formation is dynamic and runtime-based rather than reliant on an external tool, and Byzantine tolerance is provided at both device and gateway levels, which DIoT does not focus on.

(Eichhammer and Reiser, 2023) extends DIoT with dynamic runtime community formation, but assumes only uncompromised nodes join the system, which may limit applicability in practical deployments. HOMEGUARD relaxes this assumption while maintaining Byzantine robustness at both levels.

Several prior approaches (Empl et al., 2024; Alcaraz and Lopez, 2024; Murthy et al., 2024) also use the IoT use case in smart home or smart building infrastructures. However, they focus on other aspects, such as specific protocols or lightweight IDS methods. Others (Li et al., 2020; Perdigão et al., 2024) address issues of data imbalance and provide data-centric solution approaches in the field of industrial IoT. HOMEGUARD is focusing on robustness aspects.

Asynchronous, Byzantine-Robust and Hierarchical FL (Cox et al., 2024) propose Catalyst, an asynchronous Byzantine-robust FL approach that HOMEGUARD builds upon. However, Catalyst is designed not to tolerate crashed clients when exactly $2f + 1$ clients participate, and it falls back to synchronous FL. HOMEGUARD resolves this by assuming sufficient participation, guaranteeing aggregation even under f Byzantine clients. Furthermore, HOMEGUARD introduced communities to cope with heterogeneous IoT devices.

(Yu et al., 2023) propose Async-HFL, an asynchronous hierarchical FL framework for three-tier IoT networks with strong convergence properties, but it assumes honest devices, provides no protection against poisoning, and ignores model diversity across heterogeneous device types. HOMEGUARD addresses both gaps via Byzantine-robust aggregation at two levels and privacy-preserving community formation restricted to semantically similar devices.

(Sattler et al., 2020) show that grouping clients by gradient cosine similarity in Clustered FL naturally isolates adversarial participants. However, gradient-based clustering implicitly leaks local training data. HOMEGUARD instead uses privacy-preserving traffic fingerprints. CFL also does not focus on formal fault tolerance guarantees, operates synchronously in a flat architecture, and is less directly applicable to the hierarchical, asynchronous, resource-constrained IoT setting that HOMEGUARD targets.

(Qiu et al., 2020) demonstrate a black-box adversarial attack on deep learning–based IoT network IDS using model extraction and saliency maps, achieving high attack success rates with minimal packet modifications. We cite this work because robustness against adaptive adversaries is a core requirement for IoT intrusion detection and thus represents a key motivation for robust aggregation and attack resilience in collaborative IDS.

To the best of our knowledge, we are the first to approach incorporating a community structure across asynchronous Byzantine-robust FL processes within an IDS IoT environment. We provide a realistic evaluation environment within the IDS space, encompassing various attack types and scenarios.

7 CONCLUSION

HOMEGUARD is a community-driven hierarchical FL framework for intrusion detection in smart home IoT infrastructures. HOMEGUARD addresses key challenges of existing FL-based IDS approaches, which are IoT device heterogeneity, model diversity, device count disparity, and data availability imbalance, by offloading model training to gateways and grouping devices into communities via privacy-preserving traffic fingerprinting and HDBSCAN-based clustering. The asynchronous, Byzantine-robust aggregation via Catalyst tolerates both delayed and malicious updates, while the hierarchical cloud layer enables collaboration across independent smart homes. Our evaluation on the DIoT dataset shows that HOMEGUARD achieves an average TPR of 97.86% locally and 97.53% globally, with a false positive rate of 0% across all attack phases.

Community formation results are stable, semantically coherent, and remain correct in asynchronous settings. To the best of our knowledge, HOMEGUARD is the first approach to combine asynchronous community formation with Byzantine-robust FL in a hierarchical IDS tailored to real-world smart home constraints. Future work includes evaluating HOMEGUARD under a broader range of attack types, exploring adaptive fingerprinting for drifting device behavior, and investigating the scalability of global community formation across larger deployments.

ACKNOWLEDGEMENTS

This work was funded by the Bavarian Ministry of Science and Arts under the project “ForDaySec”.

REFERENCES

Alcaraz, C. and Lopez, J. (2024). Digital twin-assisted anomaly detection for industrial scenarios. *International Journal of Critical Infrastructure Protection*, 47:100721.

Batalla, J. M., Vasilakos, A., and Gajewski, M. (2017). Secure Smart Homes: Opportunities and Challenges. *ACM Comput. Surv.*, 50(5):75:1–75:32.

Cordero, C. G., Vasilomanolakis, E., Mühlhäuser, M., and Fischer, M. (2015). Community-based collaborative intrusion detection. In *International Conference on Security and Privacy in Communication Systems*, pages 665–681. Springer.

Cox, B., Mălan, A., Chen, L. Y., and Decouchant, J. (2024). Asynchronous byzantine federated learning. *arXiv preprint arXiv:2406.01438*.

Eichhammer, P. and Reiser, H. P. (2023). A self-forming community approach for intrusion detection in heterogeneous networks. In *Nordic Conference on Secure IT Systems*, pages 263–280. Springer.

Empl, P., Böhm, F., and Pernul, G. (2024). Process-aware intrusion detection in mqtt networks. In *Conference on data and application security and privacy*, pages 91–102.

Fereidooni, H., Pegoraro, A., Rieger, P., Dmitrienko, A., and Sadeghi, A.-R. (2023). Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning. *arXiv preprint arXiv:2312.04432*.

Hernandez-Ramos, J. L., Karopoulos, G., Chatzoglou, E., Kouliaridis, V., Marmol, E., Gonzalez-Vidal, A., and Kambourakis, G. (2025). Intrusion Detection Based on Federated Learning: A Systematic Review. *ACM Comput. Surv.*, 57(12):309:1–309:65.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2017). Federated Learning: Strategies for Improving Communication Efficiency. (arXiv:1610.05492).

Li, W., Meng, W., and Au, M. H. (2020). Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in iot environments. *Journal of Network and Computer Applications*, 161:102631.

McInnes, L. and Healy, J. (2017). Accelerated hierarchical density based clustering. In *International Conference on Data Mining Workshops (ICDMW)*, pages 33–42.

Murthy, A., Asghar, M. R., and Tu, W. (2024). A lightweight intrusion detection for internet of things-based smart buildings. *Security and privacy*, 7(4):e386.

Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., and Sadeghi, A.-R. (2019). DÏoT: A Federated Self-learning Anomaly Detection System for IoT. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767.

Nguyen, T. D., Rieger, P., Chen, H., Yalame, H., Möllering, H., Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Zeitouni, S., et al. (2022). FLAME: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1415–1432.

Nguyen, T. D., Rieger, P., Miettinen, M., and Sadeghi, A.-R. (2020). Poisoning Attacks on Federated Learning-based IoT Intrusion Detection System. In *Workshop on Decentralized IoT Systems and Security*, San Diego, CA. Internet Society.

Perdigão, D., Cruz, T., Simões, P., and Abreu, P. H. (2024). Data-centric federated learning for anomaly detection in smart grids and other industrial control systems. In *Network Operations and Management Symposium (NOMS)*, pages 1–5. IEEE.

Qiu, H., Dong, T., Zhang, T., Lu, J., Memmi, G., and Qiu, M. (2020). Adversarial attacks against network intrusion detection in iot systems. *IEEE Internet of Things Journal*, 8(13):10327–10335.

Sáez-de-Cámara, X., Flores, J. L., Arellano, C., Urbieta, A., and Zurutuza, U. (2023). Clustered Federated Learning Architecture for Network Anomaly Detection in Large Scale Heterogeneous IoT Networks. *Computers & Security*, 131:103299.

Sattler, F., Muller, K.-R., Wiegand, T., and Samek, W. (2020). On the Byzantine Robustness of Clustered Federated Learning. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8861–8865, Barcelona, Spain. IEEE.

Wu, D., Ullah, R., Harvey, P., Kilpatrick, P., Spence, I., and Varghese, B. (2022). Fedadapt: Adaptive offloading for iot devices in federated learning. *IEEE Internet of Things Journal*, 9(21):20889–20901.

Xia, G., Chen, J., Yu, C., and Ma, J. (2023). Poisoning attacks in federated learning: A survey. *IEEE Access*, 11:10708–10722.

Yu, X., Cherkasova, L., Vardhan, H., Zhao, Q., Ekaireb, E., Zhang, X., Mazumdar, A., and Rosing, T. (2023). Async-hfl: Efficient and robust asynchronous federated learning in hierarchical iot networks. In *Conference on Internet of Things Design and Implementation*, pages 236–248.