CHALMERS
UNIVERSITY OF TECHNOLOGY

FedGT: Identification of Malicious Clients in Federated Learning with Secure Aggregation

Citation for the original published paper (version of record):
Xhemrishi, M., Östman, J., Wachter-Zeh, A. et al (2025). FedGT: Identification of Malicious Clients in Federated Learning with Secure Aggregation. IEEE Transactions on Information Forensics and Security, 20: 2577-2592. http://dx.doi.org/10.1109/TIFS.2025.3539964

(article starts on next page)

FedGT: Identification of Malicious Clients in Federated Learning With Secure Aggregation

Marvin Xhemrishi, *Graduate Student Member, IEEE*, Johan Östman, Antonia Wachter-Zeh, *Senior Member, IEEE*, and Alexandre Graell i Amat, *Senior Member, IEEE*

***Abstract*—Federated learning (FL) has emerged as a promising approach for collaboratively training machine learning models while preserving data privacy. Due to its decentralized nature, FL is vulnerable to poisoning attacks, where malicious clients compromise the global model through altered data or updates. Identifying such malicious clients is crucial for ensuring the integrity of FL systems. This task becomes particularly challenging under privacy-enhancing protocols such as secure aggregation, creating a fundamental trade-off between privacy and security. In this work, we propose FedGT, a novel framework designed to identify malicious clients in FL with secure aggregation while preserving privacy. Drawing inspiration from group testing, FedGT leverages overlapping groups of clients to identify the presence of malicious clients via a decoding operation. The clients identified as malicious are then removed from the model training, which is performed over the remaining clients. By choosing the size, number, and overlap between groups, FedGT strikes a balance between privacy and security. Specifically, the server learns the aggregated model of the clients in each group—vanilla federated learning and secure aggregation correspond to the extreme cases of FedGT with group size equal to one and the total number of clients, respectively. The effectiveness of FedGT is demonstrated through extensive experiments on three datasets in a cross-silo setting under different data-poisoning attacks. These experiments showcase FedGT's ability to identify malicious clients, resulting in high model utility. We further show that FedGT significantly outperforms the private robust aggregation approach based on the geometric median recently proposed by Pillutla et al. and the robust aggregation technique Multi-Krum in multiple settings.**

***Index Terms*—AI security, federated learning, group testing, malicious clients, poisoning attacks, privacy, secure aggregation, security.**

Received 8 July 2024; revised 30 November 2024 and 20 January 2025; accepted 31 January 2025. Date of publication 10 February 2025; date of current version 5 March 2025. This work was supported in part by German Research Foundation (DFG) under Grant WA 3907/7-1; in part by the Swedish Innovation Agency (VINNOVA) under Grant 2021-04783; in part by the Swedish Research Council (VR) under Grant 2020-03687 and Grant 2023-05065; and in part by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The associate editor coordinating the review of this article and approving it for publication was Prof. Haibo Hu. *(Corresponding author: Marvin Xhemrishi.)*

Marvin Xhemrishi and Antonia Wachter-Zeh are with the TUM School of Computation, Information and Technology, Technical University of Munich, 80333 Munich, Germany (e-mail: marvin.xhemrishi@tum.de; antonia.wachter-zeh@tum.de).

Johan Östman is with AI Sweden, 417 56 Gothenburg, Sweden (e-mail: johan.ostman@ai.se).

Alexandre Graell i Amat is with the Department of Electrical Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden (e-mail: alexandre.graell@chalmers.se).

Digital Object Identifier 10.1109/TIFS.2025.3539964

I. Introduction

FEDERATED learning (FL) [1] is a distributed machine learning paradigm that enables multiple devices (clients) to collaboratively train a model while preserving data privacy by sharing only local model updates instead of raw data. This approach has gained attention due to its potential to safeguard sensitive client data.

In its original form, FL is susceptible to model-inversion attacks [2], [3], which allow the central server to infer clients' data from their local model updates. As demonstrated in [4], such attacks can be mitigated by employing secure aggregation protocols [5], [6]. These protocols guarantee that the server only observes the aggregate of the client models instead of individual models. However, they also limit the server's ability to detect malicious or faulty updates, potentially enabling adversarial behavior to go undetected [7].

A salient problem in FL is poisoning attacks [8], where malicious and/or faulty clients corrupt the jointly-trained global model by introducing mislabeled training data (*data poisoning*) [9], [10], or by modifying local model updates (*model poisoning*) [11]. Poisoning attacks pose a serious security risk for critical applications. Defensive measures against these threats generally fall into two categories: robust aggregation and anomaly detection. Robust aggregation techniques [12], [13], [14] are reactive approaches designed to mitigate the effect of poisoned models, whereas anomaly detection techniques are inherently proactive and aim to identify and eliminate corrupted models [15], [16], [17]. Robust aggregation techniques can introduce bias, especially when clients have heterogeneous data [15], and their effectiveness tends to diminish with increasing number of malicious clients [18]. Moreover, a recurring issue with defense mechanisms is their reliance on accessing individual client models, leaving clients vulnerable to model-inversion attacks. Thus, a fundamental trade-off emerges between privacy and security, where privacy-preserving techniques aim to obscure individual client updates, while security-focused schemes seek to identify anomalies by inspecting these updates [19].

In this work, we tackle this privacy-security trade-off by addressing the critical question: How can poisoning detection schemes be designed to effectively identify malicious clients in a privacy-preserving manner? Our main contributions are as follows:

- We propose FedGT, a novel framework for identifying malicious clients in FL with secure aggregation. Our

framework is inspired by group testing [20], a paradigm to identify defective items in a large population that significantly reduces the required number of tests compared to the naive approach of testing each item individually. FedGT's key idea is to group clients into overlapping groups. For each group, the central server observes the aggregated model of the clients and runs a suitable test to identify the presence of malicious clients in the group. The malicious clients are then identified through a decoding operation at the server, allowing for their removal from the training of the global model.

- We introduce two distinct variants of FedGT, named FedGT-$\hat{n}_{\mathsf{m}}$ and FedGT-Δ, that differ in the decoding rule used in the inference process. FedGT-$\hat{n}_{\mathsf{m}}$ relies solely on the estimated number of malicious clients, whereas FedGT-Δ employs a Neyman-Pearson-based inference method. While FedGT-$\hat{n}_{\mathsf{m}}$ minimizes false alarms at the cost of higher misdetections, FedGT-Δ prioritizes reducing misdetections with a higher tolerance for false alarms. Notably, both versions are hyperparameter-free and make no assumptions on the nature of the adversaries.
- FedGT trades-off client's data privacy, provided by secure aggregation, with *security*, understood here as the ability to identify malicious clients. It encompasses both non-private vanilla FL and privacy-oriented methods, e.g., secure aggregation, by selecting group sizes of one and the total number of clients, respectively. However, by allowing group sizes between these two extremes, FedGT strikes a balance between privacy and security, i.e., improved identification capability comes at the cost of secure aggregation involving fewer clients. We quantify the privacy implications of a given assignment matrix and devise an algorithm to group clients in a way that ensures privacy is not violated.
- We showcase FedGT's effectiveness in identifying malicious clients without a significant impact on model utility through experiments on the MNIST, CIFAR-10, and ISIC2019 datasets under both targeted and untargeted offline data-poisoning attacks. Our focus is specifically on the cross-silo scenario, wherein the number of clients is moderate (up to 50 [21]) and offline data-poisoning is the predominant attack vector [22]. Real-world examples of this critical setting include collaborations among banks developing anti-money laundering models or hospitals co-training models for medical diagnosis [23], [24].

Organization: The paper is organized as follows. In Section II, we discuss the related work and Section III sets the notation and discusses the preliminaries. We introduce our framework and its ingredients in Section IV. Section V describes two different strategies of inferring the identity of malicious clients. In Section VI we describe the experimental setup and present the empirical results. Section VII concludes the paper and discusses the limitations of this work.

II. Related Work

To the best of our knowledge, only the works [25], [26], [27] address resiliency against poisoning attacks in conjunction with secure aggregation. The work [25] is the first single-server solution to account for both privacy and security in FL. The protocol is based on dropout-resilient secure aggregation, where the server utilizes secret sharing to first obtain the pairwise Euclidean distance between client updates and then selects which clients to aggregate by means of multi-Krum [12]. However, it is not clear whether pairwise differences can leak additional information. In [26], a robust aggregation protocol, dubbed RFA, is proposed. This protocol is based on an approximate geometric median, computed by means of secure aggregation. However, RFA lacks the capability to identify malicious clients and is known to be inferior to other robust aggregation techniques, especially when dealing with heterogeneous client data [28]. The work [27] presents a privacy-preserving tree-based robust aggregation method. In particular, each leaf in the tree consists of a subgroup of clients who securely aggregate their local models. To achieve privacy between subgroups, masking is done on all but the last parameters in the aggregated models. By using the Euclidean distance between the unmasked parameters and the corresponding parameters in the global model, an outlier removal scheme, based on variance thresholding, is used iteratively to determine what groups should contribute to the global model. The approach in [27] is the method closest to ours as it relies on dividing clients into subgroups and testing the group aggregates. However, contrary to FedGT, it is unable to identify malicious clients and leverage the information of overlapping groups.

III. Preliminaries

A. Notation

We use lowercase bold letters and uppercase bold letters to denote row vectors and matrices, respectively, e.g., $\boldsymbol{x}$ and $\boldsymbol{X}$. The i-th element of vector $\boldsymbol{x}$ is denoted as x_i. We use calligraphic letters to denote sets, e.g., $\mathcal{X}$. For an integer x, we use the notation $[x]$ to denote the set of all positive integers less than equal to x, i.e., $[x] = \{1, 2, \ldots, x\}$. The empty set is denoted by $\varnothing$. $\mathbb{N}$ denotes the set of natural numbers. The logical disjunction operator is represented by $\vee$ and the logical conjunction operator by $\wedge$. For a matrix $\boldsymbol{X}$ and a vector $\boldsymbol{x}$, we use the notation $\boldsymbol{x} \vee \boldsymbol{X}^{\mathsf{T}}$ to denote a matrix-vector operator, similar to the multiplication, where the dot product is performed using the logical conjunction and the addition is computed using the logical disjunction. Finally, we denote by $w_{\mathsf{H}}(\boldsymbol{x})$ the Hamming weight of vector $\boldsymbol{x}$, i.e., the number of nonzero entries of $\boldsymbol{x}$. For convenience, the key variables used throughout the paper are summarized in Table I.

B. Group Testing

Group testing [20], [29] encompasses a family of test schemes aiming at identifying items affected by some particular condition, usually called *defective* items (e.g., individuals infected by a virus), among a large population of n items (e.g., all individuals). The overarching goal of group testing is to design a testing scheme such that the number of tests needed to identify the defective items is minimized. The principle behind group testing is that, if the number of defective items is significantly smaller than n, then negative tests on groups (or

TABLE I
NOTATIONS

Notation	Representation
n	Total number of clients
n_{m}	Total number of malicious clients
m	Total number of groups
$\mathcal{P}_i$	Set of the members of the i-th group
$\boldsymbol{d}$	The defective vector realization
$\hat{\boldsymbol{d}}$	Estimated defective vector
$\mathcal{M}$	The set of malicious clients
d_i	The status (malicious "1" or benign "0") of client i
$\boldsymbol{A}$	The assignment matrix for group testing
$\boldsymbol{s}$	The syndrome vector (computed as in (1))
s_i	The status of group i ("1" contaminated group, "0" otherwise)
$\boldsymbol{t}$	The test vector
t_i	The test value for group $i, i \in [m]$
δ	The prevalence
$\hat{\delta}$	The estimated value for prevalence
$n_{\mathsf{m}}^{\max}$	The maximum number of malicious clients for FedGT as in (4)
$\hat{n}_{\mathsf{m}}$	The estimated number of malicious clients by FedGT as in (17)
κ	Probability constraint of having only malicious groups
P_{MD}	The probability of misdetection (see (2))
P_{FA}	The probability of false alarms (see (3))
β	Weighting parameter between P_{MD} and P_{FA}
Λ	Neyman-Pearson thresholding of the a posteriori LLR
Δ	Neyman-Pearson thresholding of the LLR (see (7))
$\Delta\,(n_{\mathsf{m}})$	The values of Δ used for different $\hat{n}_{\mathsf{m}}$ by FedGT-Δ (see (16))
$Q(\boldsymbol{t}\|\boldsymbol{s})$	Binary model for the test noise
p	Crossover probability of a binary symmetric channel
$k_{\max}$	Maximum number of clusters computable as in (5)
$\mathsf{s}^{(k)}$	Silhouette score for a clustering outcome with k clusters
d_k	Dunn index for a clustering outcome with k clusters
$\mathsf{s}^{\text{thres}}$	Threshold on the silhoutte score
$\hat{k}$	Estimated number of clusters as in (6)
$\boldsymbol{v}$	Vector of utility metric (e.g. accuracy, recall) per group
$\boldsymbol{p}$	Vector of the first principal component representation per group
L	Label of a data point
n_c	Total number of classes
α	Parameter for the Dirichlet distribution that controls the heterogenity

pools) of items can spare many individual tests. Following this principle, items are grouped into overlapping groups, and tests are performed on each group. Based on the test results on the groups, the defective items can then be identified—in general with some probability of error—via a decoding operation.

C. Threat Model

We consider a cross-silo scenario with an honest-but-curious server and n clients out of which n_{m} are compromised (referred to as *malicious* clients). The number of malicious clients and their identities are unknown to the server.

A client may be compromised due to hardware malfunction or adversarial corruption. In the latter case, we assume that the malicious clients can collude and perform coordinated attacks against the global model. In this paper, we focus on offline[1] data-poisoning attacks, which represent the most realistic type of attack in cross-silo FL [22]. In cross-silo FL, clients are typically large established entities, making offline attacks—where erroneous data is used unintentionally or due to malicious tampering—the primary concern. Online attacks, like model poisoning or online data poisoning, are considered less feasible in this context since adversaries are unlikely to gain control over the client behavior during training [22].

[1]We use the terminology "online" and "offline" attacks to describe the behavior of malicious clients. Offline attacks occur before training begins, where malicious clients manipulate the static dataset used for training. In contrast, online attacks occur progressively during the training process, where malicious clients continuously introduce poisoned data, dynamically refining their attack strategy.

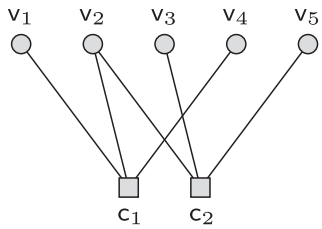

Fig. 1. The bipartite graph of the matrix $\boldsymbol{A}$ in Example 1. The circles represent variable nodes and the squares represent check nodes.

IV. FedGT: Group Testing for FL With Secure Aggregation

We consider a population of n clients, n_{m} of which are malicious. We define the *defective vector* $\boldsymbol{d} = (d_1, d_2, \ldots, d_n)$ with entries representing whether a client j is malicious ($d_j = 1$) or not ($d_j = 0$). It follows that $\sum_{j=1}^{n} d_j = n_{\mathsf{m}}$. Note that $\boldsymbol{d}$ is unknown, i.e., we do not know a priori which clients are the malicious ones.

Borrowing ideas from group testing [20], the n clients are grouped into m overlapping *test groups*. We denote by $\mathcal{P}_i$ the set of client indices belonging to test group $i \in [m]$, i.e., if client j is a member of test group i, then $j \in \mathcal{P}_i$.

Definition 1 (Assignment Matrix): The assignment of clients to test groups can be described by an assignment matrix $\boldsymbol{A} = (a_{i,j})$, $i \in [m]$, $j \in [n]$, where $a_{i,j} = 1$ if client j participates in test group i and $a_{i,j} = 0$ otherwise.

The assignment of clients to test groups, i.e., matrix $\boldsymbol{A}$, can also be conveniently represented by a bipartite graph consisting of n *variable nodes* (VNs) $\mathsf{v}_1, \ldots, \mathsf{v}_n$ corresponding to the n clients, and m *constraint nodes* (CNs) $\mathsf{c}_1, \ldots, \mathsf{c}_m$, corresponding to the m test groups. An edge between VN v_j and CN c_i is then drawn if client j participates in test group i, i.e., if $a_{i,j} = 1$. Matrix $\boldsymbol{A}$—and hence the corresponding bipartite graph—is a design choice that may be decided offline, analogous to the model architecture, and shared with the clients for transparency.

Example 1: The bipartite graph corresponding to a scenario with 5 clients and 2 test groups with assignment matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

is depicted in Fig. 1.

In FedGT, for each test group, a secure aggregation mechanism is employed to reveal only the aggregate of the client models in the test group to the server. Let $\boldsymbol{u}_i$, $i \in [m]$, be the aggregate model of test group i. For each test group i, the central server applies a binary test on the corresponding aggregate model, $\mathsf{t} : \boldsymbol{u}_i \to \{0, 1\}$. Let $t_i = \mathsf{t}(\boldsymbol{u}_i) \in \{0, 1\}$ be the result of the test for test group i, where $t_i = 1$ if the test is positive, i.e., there is at least a malicious client in the group, and $t_i = 0$ if the test is negative, i.e., no malicious clients are in the group. We collect the result of the m tests in the binary vector $\boldsymbol{t} = (t_1, t_2, \ldots, t_m)$.

We propose a suitable test in Section IV-C. However, we remark that the proposed framework is general and can be applied to any test on the test group aggregates.

We define the syndrome vector $\boldsymbol{s} = (s_1, \ldots, s_m)$, where $s_i = 1$ if at least one client participating in test group i is

malicious and $s_i = 0$ if no client participating in test group i is malicious, i.e.,

$$s_i = \bigvee_{j \in \mathcal{P}_i} d_j \quad \text{and} \quad \boldsymbol{s} = \boldsymbol{d} \vee \boldsymbol{A}^{\mathsf{T}}. \tag{1}$$

For perfect (non-noisy) test results, it follows that $\boldsymbol{t} = \boldsymbol{s}$. However, note that the result of a test may be erroneous, i.e., the result of the test may be $t_i = 1$ even if no malicious clients are present (i.e., $s_i = 0$) or $t_i = 0$ even if malicious clients are present (i.e., $s_i = 1$). In general, the (noisy) test vector $\boldsymbol{t}$ is statistically dependent on the syndrome vector $\boldsymbol{s}$ according to an (unknown) probability distribution $Q(\boldsymbol{t}|\boldsymbol{s})$.

Given the test results $\boldsymbol{t}$ and the assignment matrix $\boldsymbol{A}$, the goal of FedGT is to identify the malicious clients, i.e., infer the defective vector $\boldsymbol{d}$. The design of the assignment matrix $\boldsymbol{A}$ and the corresponding inference problem is akin to an error-correction coding problem, where the assignment matrix $\boldsymbol{A}$ can seen as the parity-check matrix of a code, and the inference problem corresponds to a decoding operation based on $\boldsymbol{A}$ and $\boldsymbol{t}$. Thus, a suitable choice for $\boldsymbol{A}$ is the parity-check matrix of a powerful error-correcting code, i.e., with good distance properties. Furthermore, $\boldsymbol{d}$ can be inferred by applying conventional decoding techniques. We denote by $\hat{\boldsymbol{d}} = (\hat{d}_1, \ldots, \hat{d}_n)$ the estimated defective vector provided by the decoding operation, and define $\hat{\mathcal{M}} = \{i : \hat{d}_i = 1\}$. Once $\hat{\boldsymbol{d}}$ has been obtained, clients $i \in \hat{\mathcal{M}}$ are excluded from the training and the server aggregates the models of the remaining—flagged non-malicious—clients by means of secure aggregation.

The performance of FedGT, measured in terms of the utility of the model, is affected by two quantities: the misdetection probability, i.e., the probability that a malicious client is flagged as non-malicious, and the false-alarm probability, i.e., the probability that a non-malicious client is flagged as malicious, defined as[2]

$$P_{\mathsf{MD}} \triangleq \frac{1}{n} \sum_{i=1}^{n} \Pr(\hat{d}_i = 0 | d_i = 1), \tag{2}$$

$$P_{\mathsf{FA}} \triangleq \frac{1}{n} \sum_{i=1}^{n} \Pr(\hat{d}_i = 1 | d_i = 0). \tag{3}$$

A high misdetection probability will result in many malicious clients poisoning the global model, hence yielding poor utility, while a high false-alarm probability will result in excluding many non-malicious clients from the training, thereby also impairing the utility. The misdetection and false-alarm probabilities depend in turn on the assignment matrix $\boldsymbol{A}$, the decoding strategy, and the nature of the test performed. We discuss the decoding strategy to estimate $\boldsymbol{d}$ in Section V.

A. Privacy-Security Trade-off

Vanilla FL [1] and FL with *full* secure aggregation [5] can be seen as the two extreme cases of FedGT, corresponding to n (non-overlapping) groups and a single group with n clients, respectively. In vanilla FL, tests on individual models can be conducted, facilitating the identification of malicious clients. However, this comes at the expense of clients' privacy. In contrast, full secure aggregation provides privacy by enabling the server to observe only the aggregate of the n models, but it does not permit the identification of malicious clients.

Under FedGT, the server observes m aggregated models $\boldsymbol{u}_1 \ldots, \boldsymbol{u}_m$, with $\boldsymbol{u}_i = \sum_{j=1}^{n} a_{i,j} \boldsymbol{c}_j$ and $\boldsymbol{c}_j$ being the local model of client j. The privacy of the clients increases with the number of models aggregated [30]. Hence, FedGT trades privacy for providing security, i.e., identification of malicious clients. Furthermore, there might be additional privacy loss due to the aggregates being from overlapping groups. This loss depends on the assignment matrix $\boldsymbol{A}$ and is agnostic to the number of malicious clients participating in the training. The privacy of FedGT is given in Proposition 1. We write the proposition assuming the malicious clients follow the threat model as in Section III-C, i.e., the malicious clients perform offline attacks and cannot control the client's behavior during the training, which is a realistic assumption for cross-silo FL.

Proposition 1: Let the assignment of clients to test groups be defined by assignment matrix $\boldsymbol{A}$ and let r be the smallest non-zero Hamming weight of the vectors in the row span of $\boldsymbol{A}$ (in the coding theory jargon, the minimum Hamming distance of the code generated by $\boldsymbol{A}$ as its generator matrix). Then FedGT achieves the same privacy as a secure aggregation scheme with r clients.

Proof: Due to the overlapping groups arising from matrix $\boldsymbol{A}$, there might exist a vector $\boldsymbol{b} \in \mathbb{R}^m$ such that $\sum_{i=1}^{m} b_i \boldsymbol{u}_i = \boldsymbol{c}_j$ for some $j \in [n]$, or equivalently $\boldsymbol{b}\boldsymbol{A} = \boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is the j-th unit vector. This will occur if $\boldsymbol{e}_j \in \mathsf{Sp}_{\mathrm{r}}(\boldsymbol{A})$, where $\mathsf{Sp}_{\mathrm{r}}(\boldsymbol{A})$ is the row span of $\boldsymbol{A}$. Generally speaking, for a subset $\mathcal{R} \subset [n]$ of cardinality r, if $\sum_{\iota \in \mathcal{R}} f_\iota \boldsymbol{e}_\iota \in \mathsf{Sp}_{\mathrm{r}}(\boldsymbol{A})$ where $f_\iota \neq 0$, there exists a vector $\boldsymbol{b}'$ such that $\sum_{i=1}^{m} b'_i \boldsymbol{u}_i = \sum_{\iota \in \mathcal{R}} f_\iota \boldsymbol{c}_\iota$. Thus, we conclude that FedGT achieves the same privacy as a secure aggregation scheme with $r < n$ clients, where r is the smallest non-zero cardinality of the subset $\mathcal{R}$. In other words, r is the smallest non-zero Hamming weight of the vectors in the row span of $\boldsymbol{A}$. ■

Note that no aggregation of less than r client models can be revealed to the server. Aggregation of $r > 1$ client models has been shown to effectively mitigate model-inversion attacks from the server [4, Table 10].

B. The Choice of Assignment Matrix $\boldsymbol{A}$

The assignment matrix $\boldsymbol{A}$ should be carefully chosen to balance the trade-off between privacy and security: To improve the identification of malicious clients, one should choose $\boldsymbol{A}$ as the parity-check matrix of an error-correcting code with good distance properties, while to achieve a high privacy level, $\boldsymbol{A}$ should correspond to the generator matrix of a code of large minimum Hamming distance.

On the other hand, for FedGT to effectively detect malicious clients with a low probability of false alarm, it is essential that some group tests yield negative results: If all tests are positive, i.e., $\boldsymbol{t} = \mathbf{1}$, FedGT will flag all clients as malicious, resulting in the highest probability of false alarm of $P_{\mathsf{FA}} = \frac{n - n_{\mathrm{m}}}{n}$.

As the number of malicious clients grows, the likelihood of observing only positive test outcomes, i.e., $\boldsymbol{t} = \mathbf{1}$, increases.

[2] As in other works, we use the normalization factor $1/n$. Hence, P_{MD} and P_{FA} are not strictly probabilities, as their values do not lie between 0 and 1.

Furthermore, the choice of $\boldsymbol{A}$ highly impacts the probability of having all groups contaminated. More precisely, the probability of having all groups contaminated is fully determined by $\boldsymbol{A}$ and n_{m}. For small matrices $\boldsymbol{A}$, this probability can be computed exactly, while for larger ones, it can be approximated using a Monte Carlo approach.

The assignment matrix $\boldsymbol{A}$ should be chosen such that the probability of having all groups contaminated is small (note that n_{m} is out the designer's control). Alternatively, one can impose a constraint on the probability of all groups being contaminated and find the assignment matrix $\boldsymbol{A}$ that supports the maximum number of malicious clients, $n_{\mathsf{m}}^{\max}$, such that this constraint is satisfied. The value $n_{\mathsf{m}}^{\max}$ is intrinsic to $\boldsymbol{A}$ and can be obtained offline, as outlined next.

Consider the best-case scenario of noiseless tests, i.e., $\boldsymbol{t} = \boldsymbol{s}$ and let the n_{m} malicious clients be assigned uniformly at random. Let S_i be the random variable corresponding to the syndrome of the i-th group (corresponding also to the test outcome of the i-th group for noiseless group testing) and $\boldsymbol{S} = (S_1, \ldots, S_m)$ (the corresponding realizations are defined in (1)). Then, for a fixed assignment matrix $\boldsymbol{A}$, we can solve

$$n_{\mathsf{m}}^{\max} = \arg\max_{n_{\mathsf{m}}} \left\{ \Pr(\boldsymbol{S} = \boldsymbol{1} | N_{\mathsf{m}} = n_{\mathsf{m}}) \leq \kappa \right\}, \tag{4}$$

where N_{m} is the random variable that represents the number of malicious clients and κ denotes the probability constraint of all groups being contaminated. This approach provides a systematic procedure of identifying assignment matrices that are suitable for a given scenario.

Algorithm 1 The Search for the Assignment Matrix $\boldsymbol{A}$

Input: $n \in \mathbb{N}$, desired privacy level $r \in \mathbb{N}, r \leq n$, $\kappa \in [0, 1]$, $\mathcal{A} = \{\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots\}$: set of parity-check matrices of codes of length n.
Output: $\boldsymbol{A}$ or None
$\mathcal{A}' \leftarrow \varnothing$;
foreach $\boldsymbol{A}_i \in \mathcal{A}$ **do**
  $\rho_i \leftarrow \min_{\boldsymbol{x} \in \mathsf{Sp}_{\mathsf{r}}(\boldsymbol{A}_i), \boldsymbol{x} \neq \boldsymbol{0}} w_{\mathsf{H}}(\boldsymbol{x})$;
  if $\rho_i < r$ **then**
    continue ; /* Lower privacy than required */
  else
    $\mathcal{A}' \leftarrow \mathcal{A}' \cup \{\boldsymbol{A}_i\}$;
  end
end
if $|\mathcal{A}'| = 0$ **then**
  $\boldsymbol{A} \leftarrow$ None ; /* No possible $\boldsymbol{A}$*/
else
  $\boldsymbol{A} \leftarrow \arg\max_{\boldsymbol{A}_i \in \mathcal{A}'} \left\{ n_{\mathsf{m}}^{\max}(\boldsymbol{A}_i) \right\}$; /* Compute $n_{\mathsf{m}}^{\max}(\boldsymbol{A}_i)$ as in (4)*/
end
return $\boldsymbol{A}$;

In Algorithm 1, we provide a simple way to find the assignment matrix $\boldsymbol{A}$ for the grouping of clients. The algorithm takes as input the number of clients n, the desired privacy level (equivalent to the privacy provided by secure aggregation with r clients), the probability constraint κ, and a set of assignment matrices corresponding to well-established error-correcting codes (set $\mathcal{A}$). This set can be taken from online databases [31], [32], [33]. The search in Algorithm 1 is tailored toward a fixed privacy constraint and searches for the assignment matrix in $\mathcal{A}$ that maximizes $n_{\mathsf{m}}^{\max}$. We note that the search could be reversed, i.e., a minimum requirement for $n_{\mathsf{m}}^{\max}$ can be imposed and the search would find the parity-check matrix yielding the highest privacy. Moreover, also a hybrid search could be used, which is definitely application specific. However, it is important to note that the parity-check matrices of the codes are easily available and the search could be tailored to any specific application.

C. Test Design

FedGT can be applied to any test on the test group aggregates. However, it is essential to design an accurate test, as the performance of FedGT is impaired by the noisiness of the test. In this section, we propose a test that, as shown in the numerical results section, yields low error rate.

To begin, we make the observation that the utility of an aggregated model tends to decrease as it gets contaminated by a larger number of poisoned models. Moreover, the work in [9] shows that when using dimensionality reduction tools like principal component analysis (PCA), the local models of the malicious clients tend to cluster around similar values, even in the first component.[3] Empirically, we observed that the findings from [9] apply also to our group testing scenario: the aggregated models of test groups with the same number of malicious clients tend to cluster around the same value in the first component.

Motivated by this observation, we propose a testing strategy in which we first cluster the test groups (i.e., the corresponding aggregated models) into clusters based on the number of malicious clients involved in the test group. Then, for each cluster, we compute the average utility of the aggregated models using a small validation dataset at the server (a minor assumption as motivated in Section VI), and finally, we declare the result of the test for the test groups within the cluster with highest average utility as negative ($t = 0$) and the result of the test for all other test groups as positive ($t = 1$). The details of the proposed testing strategy are outlined below.

Let v_i denote a measured utility metric of the aggregated model of test group i evaluated on the validation dataset, and let $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$. Also, let p_i be the first principal component representation of the aggregated model of test group i evaluated on the fully-connected layer, and let $\boldsymbol{p} = (p_1, \ldots, p_m)$. We form m points $\boldsymbol{c}_i = (v_i, p_i)$ and cluster them using the k-means algorithm [34]. Since k-means requires the number of clusters k, we compute the maximum possible number of clusters,

$$k_{\max} = \min \left\{ m, \max_{i \in [m]} |\mathcal{P}_i| + 1 \right\}, \tag{5}$$

and perform k-means clustering for all $k \in [k_{\max}]$.

The next step is to determine the optimal number of clusters. Two popular metrics for this purpose are the Silhouette score

[3]The first component captures the highest variance of the original observations (models).

[35] and the Dunn index [36]. In scenarios like ours, where the number of points is relatively small compared to the number of clusters, the Dunn index tends to perform better than the Silhouette score. However, the Dunn index is not effective at determining if the data should be clustered into a single cluster. To address these limitations, we propose a combined approach: first, we use the Silhouette score to assess whether the data should be clustered into one or several clusters. If multiple clusters are indicated, we then use the Dunn index to determine the precise number of clusters.

For a data point $\boldsymbol{c}_i$ in cluster $\mathcal{C}_u$, the Silhouette score is defined as

$$\mathsf{s}_i = \begin{cases} \dfrac{\mathsf{b}_i - \mathsf{a}_i}{\max\{\mathsf{a}_i, \mathsf{b}_i\}}, & |\mathcal{C}_u| > 1, \\ 0, & |\mathcal{C}_u| = 1, \end{cases}$$

where b_i is the smallest mean distance of $\boldsymbol{c}_i$ to all points in any other cluster,

$$\mathsf{b}_i = \min_{u' \neq u} \frac{1}{|\mathcal{C}_{u'}|} \sum_{\boldsymbol{c}_j \in \mathcal{C}_{u'}} \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_2^2 ,$$

and a_i is the mean distance of $\boldsymbol{c}_i$ to all other points in the same cluster,

$$\mathsf{a}_i = \frac{1}{|\mathcal{C}_u| - 1} \sum_{\boldsymbol{c}_j \in \mathcal{C}_u, j \neq i} \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_2^2 .$$

For each k, the Silhouette score of the corresponding clustering result, denoted as $\mathsf{s}^{(k)}$, is then the average of the individual Silhouette scores, i.e.,

$$\mathsf{s}^{(k)} = \frac{1}{m} \sum_{i \in [m]} \mathsf{s}_i .$$

The Dunn index [36] is defined as

$$\mathsf{d}_k = \frac{\min\limits_{i \in [k], j \in [k]: i \neq j} \|\bar{\boldsymbol{c}}_i - \bar{\boldsymbol{c}}_j\|_2^2}{\max\limits_{u \in [k]} \max\limits_{\boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C}_u} \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_2^2},$$

where $\bar{\boldsymbol{c}}_i$ denotes the center point of cluster $\mathcal{C}_i$.

Based on the Silhouette score and the Dunn index, we determine the number of clusters as

$$\hat{k} = \arg\max_{i \in [k]} \{\mathsf{d}_i\} \mathbb{1}\{\mathsf{s}_{\max} \geq \mathsf{s}^{\text{thres}}\} + \mathbb{1}\{\mathsf{s}_{\max} < \mathsf{s}^{\text{thres}}\}, \quad (6)$$

where $\mathsf{s}_{\max} = \max_{k \in [k_{\max}]} \mathsf{s}^{(k)}$. That is, we first threshold the Silhouette score to decide whether the datapoints $\{\boldsymbol{c}_i\}$ should be clustered into one or several clusters and, if multiple clusters are suggested, we identify the precise number of clusters using the Dunn index (selecting the number of clusters that maximizes the Dunn index).

We use the outcome of the clustering to determine the test result for each test group. Note that each cluster corresponds to a different number of malicious clients. In this paper, however, we consider binary test results, i.e., for a given test group the test is positive ($t = 1$) if the test determines that there is at least one malicious client in the group and the test is negative ($t = 0$) if there is none. Hence, we only need to distinguish between clusters corresponding to test groups with no malicious clients and clusters corresponding to test groups with malicious clients. To this aim, for every cluster $\mathcal{C}_i$, $i \in [\hat{k}]$, we compute the average utility as $\bar{v}_i = \frac{1}{|\mathcal{C}_i|} \sum\limits_{j: \boldsymbol{c}_j \in \mathcal{C}_i} v_j$ and apply the decision rule

$$t_j = \begin{cases} 0 & j : \boldsymbol{c}_j \in \mathcal{C}_\iota \\ 1 & j : \boldsymbol{c}_j \notin \mathcal{C}_\iota \end{cases}, \qquad \text{where } \iota = \arg\max_{i \in [\hat{k}]} \bar{v}_i ,$$

i.e., our test strategy flags all test groups within the cluster with highest utility as benign ($t = 0$) and all other test groups as containing malicious clients ($t = 1$).

D. Communication Cost

FedGT introduces a communication overhead only in the round(s) where group testing is performed. Considering a single-round of FedGT, which is the setting of our experiments, the overall communication cost of the FL consists of:

- **Before group testing**: Number of communication rounds $\times$ communication complexity of secure aggregation with n clients.
- **Group testing round**: $m\times$ communication complexity of secure aggregation with $\max_{j \in [m]} |\mathcal{P}_j|$ clients (maximum size of groups).
- **After group testing**: Number of rounds $\times$ communication complexity of secure aggregation with $n - |\hat{\mathcal{M}}|$ clients (number of clients classified as benign from the group testing).

The overall communication cost heavily depends on the scheme used for secure aggregation. In [37, Table I], the communication cost for different secure aggregation schemes is tabulated. For example, LightSecAgg [38] has a total communication complexity per round of $\mathcal{O}(n(C+1))$, where C is the model size. In this case, over K training rounds, FedGT yields a total communication complexity of $\mathcal{O}((C+1)(nK + m \max_{j \in [m]} |\mathcal{P}_j|)$ where we have assumed $\hat{\mathcal{M}} = \varnothing$, the worst-case scenario from a communication perspective.

V. Decoding: Inferring the Defective Vector $\boldsymbol{d}$

Given the test results $\boldsymbol{t}$ and the assignment matrix $\boldsymbol{A}$, FedGT estimates the defective vector $\boldsymbol{d}$. In this section, we present two decoding strategies based on probabilistic decision metrics to estimate the defective vector $\boldsymbol{d}$.

A. Strategy 1: Neyman-Pearson Based Inference

In our first strategy, we consider optimal inference in a Neyman-Pearson sense, which prescribes for some $\Delta' > 1$

$$\hat{d}_i = \begin{cases} 0 & \text{if } \Pr(\boldsymbol{t}|d_i = 0) > \Pr(\boldsymbol{t}|d_i = 1)\Delta' \\ 1 & \text{if } \Pr(\boldsymbol{t}|d_i = 0) < \Pr(\boldsymbol{t}|d_i = 1)\Delta'. \end{cases}$$

The Neyman-Pearson criterion can be rewritten in terms of the log-likelihood ratio (LLR) $L_i = \log(\Pr(\boldsymbol{t}|d_i = 0)/\Pr(\boldsymbol{t}|d_i = 1))$ as

$$\hat{d}_i = \begin{cases} 0 & \text{if } L_i > \Delta \\ 1 & \text{if } L_i < \Delta, \end{cases} \quad (7)$$

where $\Delta = \log(\Delta')$. Further, we can write the LLR L_i as

$$L_i = \log\left(\frac{\Pr(d_i = 0|\boldsymbol{t})}{\Pr(d_i = 1|\boldsymbol{t})}\right) - \log\left(\frac{\Pr(d_i = 0)}{\Pr(d_i = 1)}\right) \quad (8)$$

$$= L_i^{\mathsf{APP}} - \log\left(\frac{1-\delta}{\delta}\right), \tag{9}$$

where δ is the *prevalence* of malicious clients in the population of n clients, i.e., the probability of a client being malicious, $\delta = \Pr(d_i = 1)$. In a frequentist approach to probability, $\delta = n_{\mathsf{m}}/n$. Using (8), the Neyman-Pearson criterion in (7) can be rewritten in terms of the a posteriori LLR L_i^{APP} as

$$\hat{d}_i = \begin{cases} 0 & \text{if } L_i^{\mathsf{APP}} > \Lambda \\ 1 & \text{if } L_i^{\mathsf{APP}} < \Lambda, \end{cases} \tag{10}$$

where

$$\Lambda = \Delta + \log\left(\frac{1-\delta}{\delta}\right). \tag{11}$$

In general, if Λ increases, then P_{FA} increases and P_{MD} decreases. Note that Λ depends on the prevalence, i.e., the number of malicious clients n_{m}, which is in general not known. In the following, we provide the means to estimate the number of malicious clients n_{m}.

1) Estimation of the Number of Malicious Clients: For a given $n_{\mathsf{m}} \in [n_{\mathsf{m}}^{\max}] \cup \{0\}$, we consider all patterns of n_{m} malicious clients and define Z as the random variable representing the number of zero syndromes, i.e.,

$$Z = \sum_{i=1}^{m} \mathbb{1}\{S_i = 0\}, \tag{12}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function. Also, define

$$z = \sum_{i=1}^{m} \mathbb{1}\{s_i = 0\} \tag{13}$$

and

$$\hat{z} = \sum_{i=1}^{m} \mathbb{1}\{t_i = 0\}. \tag{14}$$

Note that, for a noiseless test, $\hat{z} = z$.

The decoder has the vector of test results $\boldsymbol{t}$ at its disposal. This information can be used to estimate n_{m} via the maximum likelihood criterion as

$$\hat{n}_{\mathsf{m}} = \arg\max_{n_{\mathsf{m}}} \Pr(Z = \hat{z} | N_{\mathsf{m}} = n_{\mathsf{m}}). \tag{15}$$

The likelihood $\Pr(Z = \hat{z}|N_{\mathsf{m}} = n_{\mathsf{m}})$ can be computed exactly for small enough assignment matrices $\boldsymbol{A}$. Note that the accuracy of the estimate is expected to deteriorate with increasing test noise.

We use the estimate $\hat{n}_{\mathsf{m}}$ to estimate the prevalence as $\hat{\delta} = \hat{n}_{\mathsf{m}}/n$. The estimated prevalence $\hat{\delta}$ can then be used in (11) to obtain Λ. However, one must still choose Δ. To this end, we consider an ideal setting, i.e., $\boldsymbol{t} = \boldsymbol{s}$ and $\hat{n}_{\mathsf{m}} = n_{\mathsf{m}}$, and find

$$\hat{\Delta}(n_{\mathsf{m}}) = \arg\min_{\Delta} \{\mathbb{E}[\beta P_{\mathsf{MD}} + (1-\beta) P_{\mathsf{FA}}]\}, n_{\mathsf{m}} \in [n_{\mathsf{m}}^{\max}], \tag{16}$$

where $\beta \in [0,1]$ weights between false alarm and misdetection and their dependency on Δ is implicit. The expectation is with respect to the n_{m} malicious clients being sampled uniformly at random and can be computed via Monte-Carlo estimation. Notably, (16) can be solved offline. During decoding, we set $\Lambda = \hat{\Delta}(\hat{n}_{\mathsf{m}}) + \log\left(\frac{1-\hat{\delta}}{\hat{\delta}}\right)$.

Hereafter, we refer to FedGT with decoding strategy 1 as FedGT-Δ. We remark that the number of clients flagged as malicious by FedGT-Δ may differ from the estimated value $\hat{n}_{\mathsf{m}}$, i.e., $w_{\mathsf{H}}(\hat{\boldsymbol{d}})$ is not necessarily equal to $\hat{n}_{\mathsf{m}}$.

B. Strategy 2: Flagging $\hat{n}_m$ Clients

We propose an alternative strategy to infer the defective vector $\hat{\boldsymbol{d}}$ by relying entirely on the estimated number of malicious clients $\hat{n}_{\mathsf{m}}$. This strategy is based on the observation that the a posteriori LLRs L_i^{APP} indicate the likelihood of a client being benign (see (8)), with higher values indicating to a more confident guess. Accordingly, Strategy 2 declares the $\hat{n}_{\mathsf{m}}$ clients with smallest L_i^{APP} as malicious and the remaining clients as benign.

Let $\boldsymbol{L}^{\mathsf{APP}} = \left(L_1^{\mathsf{APP}}, L_2^{\mathsf{APP}}, \ldots, L_n^{\mathsf{APP}}\right)$ be the vector containing the a posteriori LLRs for all clients and $\widetilde{\boldsymbol{L}}^{\mathsf{APP}} = \left(L_{i_1}^{\mathsf{APP}}, L_{i_2}^{\mathsf{APP}}, \ldots, L_{i_n}^{\mathsf{APP}}\right)$ be a sorted version of $\boldsymbol{L}^{\mathsf{APP}}$ with LLRs ordered in ascending order, i.e., $L_{i_j}^{\mathsf{APP}} \geq L_{i_k}^{\mathsf{APP}}$ for $j > k$. For an estimated number of malicious clients $\hat{n}_{\mathsf{m}}$, we define the decision rule as

$$\hat{d}_i = \begin{cases} 1 & \text{if } i \in \{i_1, i_2, \ldots, i_{\hat{n}_{\mathsf{m}}}\} \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

where $\{i_1, i_2, \ldots, i_{\hat{n}_{\mathsf{m}}}\}$ is the set of the indices of the $\hat{n}_{\mathsf{m}}$ smallest elements in $\boldsymbol{L}^{\mathsf{APP}}$. Note that using this decision strategy, contrary to FedGT-Δ, the number of nonzero entries in $\hat{\boldsymbol{d}}$ is always $\hat{n}_{\mathsf{m}}$, i.e., $w_{\mathsf{H}}(\hat{\boldsymbol{d}}) = \hat{n}_{\mathsf{m}}$. Henceforth, we refer to FedGT with decoding strategy 2 as FedGT-$\hat{n}_{\mathsf{m}}$.

Both FedGT-Δ and FedGT-$\hat{n}_{\mathsf{m}}$, require the a posteriori LLRs L_i^{APP}. For not-too-large matrices $\boldsymbol{A}$, they can be computed efficiently via the forward-backward algorithm [39], which exploits the trellis representation of the assignment matrix $\boldsymbol{A}$. For large matrices $\boldsymbol{A}$, the computation of the a posteriori LLRs is not feasible, and one needs to resort to suboptimal decoding strategies, e.g. belief propagation [40].

Given our focus on the cross-silo setting, where the number of clients is limited, we next present how to obtain the a posteriori LLRs for this setting using the trellis representation of the assignment matrix $\boldsymbol{A}$ and the forward-backward algorithm. In Section V-C, we describe how to obtain the trellis diagram for a given assignment matrix $\boldsymbol{A}$, and in Section V-D, we discuss the forward-backward algorithm to compute the a posteriori LLRs to infer $\boldsymbol{d}$.

C. Trellis Representation of Assignment Matrix $\boldsymbol{A}$

In this section, we describe the trellis representation corresponding to assignment matrix $\boldsymbol{A}$, which can be used to compute the a posteriori LLRs as described in Section V-D. The trellis representation was originally introduced for linear block codes in [41] and applied to group testing in [42].

For a given defective vector $\tilde{\boldsymbol{d}}$ (not necessarily the true one), define the *syndrome vector* $\tilde{\boldsymbol{s}} = (\tilde{s}_1, \ldots, \tilde{s}_n)$, where $\tilde{s}_i$ is given by $\tilde{s}_i = \bigvee_{j \in \mathcal{P}_i} \tilde{d}_j$. The syndrome vector can be written as a function of the defective vector $\tilde{\boldsymbol{d}}$ and the assignment matrix as $\tilde{\boldsymbol{s}} = \tilde{\boldsymbol{d}} \vee \boldsymbol{A}^{\mathsf{T}}$. Note that several defective vectors are compatible with a given syndrome $\tilde{\boldsymbol{s}}$. Let $\mathcal{D}$ be the set of all possible

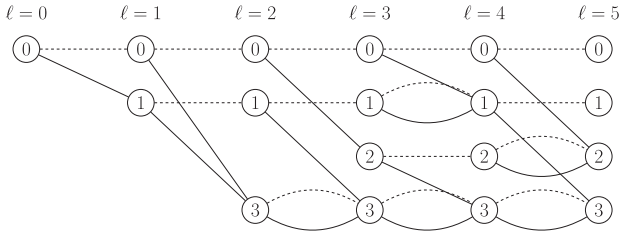

Fig. 2. Trellis representation of matrix $\boldsymbol{A}$ in Example 1. The dashed edges correspond to the symbol "0", while the solid edges correspond to the symbol "1".

defective vectors, i.e., all binary tuples of length n. We denote by $\mathcal{D}_{\boldsymbol{s}}$ the set of defective vectors compatible with syndrome vector $\boldsymbol{s}$, i.e., $\mathcal{D}_{\tilde{\boldsymbol{s}}} = \{\tilde{\boldsymbol{d}} \in \mathcal{D} : \tilde{\boldsymbol{d}} \vee \boldsymbol{A}^{\mathsf{T}} = \tilde{\boldsymbol{s}}\}$.

Let $\boldsymbol{a}_j$ be the j-th column of matrix $\boldsymbol{A}$. The syndrome corresponding to defective vector $\tilde{\boldsymbol{d}}$ can then be rewritten as $\tilde{\boldsymbol{s}} = \bigvee_{i=1}^{n}(\tilde{d}_i \wedge \boldsymbol{a}_i^{\mathsf{T}})$. This equation naturally leads to a trellis representation of the assignment matrix $\boldsymbol{A}$ as explained next. A trellis is a graphical way to represent matrix $\boldsymbol{A}$, consisting of a collection of nodes connected by edges. The trellis corresponding to matrix $\boldsymbol{A}$ in Example 1 is depicted in Fig. 2. Horizontally, the nodes, called trellis states, are grouped into sets indexed by parameter $\ell \in \{0, \ldots, n\}$, referred to as the trellis depth.

Let $\tilde{\boldsymbol{s}}_\ell$ be the *partial* syndrome vector at trellis depth $\ell \in [n]$ corresponding to $\tilde{\boldsymbol{d}}$, given as $\tilde{\boldsymbol{s}}_\ell = \bigvee_{i=1}^{\ell}(\tilde{d}_i \wedge \boldsymbol{a}_i^{\mathsf{T}})$. It is easy to see that $\tilde{\boldsymbol{s}}_\ell$ can be obtained from $\tilde{\boldsymbol{s}}_{\ell-1}$ as $\tilde{\boldsymbol{s}}_\ell = \tilde{\boldsymbol{s}}_{\ell-1} \vee (\tilde{d}_\ell \wedge \boldsymbol{a}_\ell^{\mathsf{T}})$, with $\tilde{\boldsymbol{s}}_0$ being the all-zero vector. The trellis representation is such that each state in the trellis represents a particular partial syndrome. The trellis is then constructed as follows: At trellis depth $\ell = 0$ there is a single trellis state corresponding to $\tilde{\boldsymbol{s}}_0$. At trellis depth $\ell \in [n]$, the trellis states correspond to all possible partial syndrome vectors $\tilde{\boldsymbol{s}}_\ell$ for all possible partial syndrome vectors $(\tilde{d}_1, \ldots, \tilde{d}_\ell)$, with $\tilde{d}_i \in \{0, 1\}$. For example, at trellis depth $\ell = 1$ there are only two trellis states, corresponding to partial syndromes $0 \wedge \boldsymbol{a}_1^{\mathsf{T}} = (0, \ldots, 0)$ and $1 \wedge \boldsymbol{a}_1^{\mathsf{T}} = (a_{1,1}, \ldots, a_{1,m})$, i.e., for $\tilde{d}_1 = 0$ and $\tilde{d}_1 = 1$, respectively. Note that at trellis depth $\ell = n$, there are 2^m trellis states, corresponding to all possible syndromes $\tilde{\boldsymbol{s}}$. For simplicity, we label the trellis state corresponding to partial syndrome vector $\tilde{\boldsymbol{s}}_\ell = (s_{\ell,1}, \ldots, s_{\ell,m})$ by its decimal representation $\sum_{i=1}^{m} \tilde{s}_{\ell,i} 2^{i-1}$. Finally, an edge from the node at trellis depth ℓ corresponding to partial syndrome $\tilde{\boldsymbol{s}}_\ell$ to the node at trellis depth $\ell + 1$ corresponding to partial syndrome $\tilde{\boldsymbol{s}}_{\ell+1}$ is drawn if $\tilde{\boldsymbol{s}}_{\ell+1} = \tilde{\boldsymbol{s}}_\ell \vee (\tilde{d}_{\ell+1} \wedge \boldsymbol{a}_{\ell+1}^{\mathsf{T}})$, with $\tilde{d}_{\ell+1} \in \{0, 1\}$. The edge is labeled by the value of $\tilde{d}_{\ell+1}$ enabling the transition between $\tilde{\boldsymbol{s}}_\ell$ and $\tilde{\boldsymbol{s}}_{\ell+1}$.

Example 2: For the trellis of Fig. 2, corresponding to the assignment matrix $\boldsymbol{A}$ in Example 1 with $n = 5$ nodes and $m = 2$ tests, the number of trellis states at trellis depth $\ell = 5$ is $2^2 = 4$, i.e., all length-2 binary vectors (in decimal notation $\{0, 1, 2, 3\}$). At trellis depth $\ell = 2$, there are three states, corresponding to all possible partial syndromes $\tilde{\boldsymbol{s}} = \bigvee_{i=1}^{2}(\tilde{d}_i \wedge \boldsymbol{a}_i^{\mathsf{T}})$, i.e., all possible (binary) linear combinations of the two first columns of matrix $\boldsymbol{A}$, resulting in states $(0, 0) \vee (0, 0) = (0, 0) = 0$, $(0, 0) \vee (1, 1) = (1, 1) = 3$, $(1, 0) \vee (0, 0) = (1, 0) = 1$, and $(1, 0) \vee (1, 1) = (1, 1) = 3$.

The trellis graphically represents all possible defective vectors $\tilde{\boldsymbol{d}}$ and their connection to the syndromes $\tilde{\boldsymbol{s}}$ via the assignment matrix $\boldsymbol{A}$. In particular, the paths along the trellis originating in the all-zero state at trellis depth $\ell = 0$ and ending in trellis state $\tilde{\boldsymbol{s}}$ at trellis depth $\ell = n$ correspond to all defective vectors $\tilde{\boldsymbol{d}}$ compatible with syndrome $\tilde{\boldsymbol{s}}$.

D. The Forward-Backward Algorithm

The a posteriori LLRs can be computed efficiently using the trellis representation of matrix $\boldsymbol{A}$ introduced in the previous subsection via the forward-backward algorithm [39]. Let $\mathcal{E}_\ell^{(0)}$ and $\mathcal{E}_\ell^{(1)}$ be the set of edges connecting trellis states at trellis depth $\ell - 1$ with states at trellis depth ℓ labeled by $\tilde{d}_\ell = 0$ and $\tilde{d}_\ell = 1$, respectively. The a posteriori LLRs L_ℓ^{APP} can be computed as

$$L_\ell^{\mathsf{APP}} = \log \sum_{(\sigma', \sigma) \in \mathcal{E}_\ell^{(0)}} \alpha_{\ell-1}(\sigma') \gamma(\sigma', \sigma) \beta_\ell(\sigma) - \log \sum_{(\sigma', \sigma) \in \mathcal{E}_\ell^{(1)}} \alpha_{\ell-1}(\sigma') \gamma(\sigma', \sigma) \beta_\ell(\sigma)\,, \quad (18)$$

where (σ', σ) denotes an edge connecting state σ' at trellis depth $\ell - 1$ with state σ at trellis depth ℓ.

The quantities $\alpha_{\ell-1}(\sigma')$ and $\beta_\ell(\sigma)$ are called the forward and backward metrics, respectively, and can be computed using the recursions

$$\alpha_\ell(\sigma) = \sum_{\sigma'} \alpha_{\ell-1}(\sigma') \gamma_\ell(\sigma', \sigma)\,,$$
$$\beta_{\ell-1}(\sigma') = \sum_{\sigma} \beta_\ell(\sigma) \gamma_\ell(\sigma', \sigma)\,,$$

with initialization of the forward recursion $\alpha_0(0) = 1$ and of the backward recursion $\beta_n(\sigma) = Q(\boldsymbol{t}|\boldsymbol{s}(\sigma))$, where $\boldsymbol{s}(\sigma)$ is the syndrome corresponding to trellis state σ. The quantity $\gamma_\ell(\sigma', \sigma)$ is called the branch metric and is given by

$$\gamma_\ell(\sigma', \sigma) = \begin{cases} 1 - \delta & \text{if } (\sigma', \sigma) \in \mathcal{E}_\ell^{(0)} \\ \delta & \text{if } (\sigma', \sigma) \in \mathcal{E}_\ell^{(1)}. \end{cases}$$

The a posteriori LLRs computed via (18) are then used to make decisions on $\{d_i\}$ according to (10).

E. FedGT Hyperparameters

As discussed in the previous section, the decoder requires the distribution $Q(\boldsymbol{t}|\boldsymbol{s})$ and the prevalence δ, which are in general unknown. For the prevalence, we use the estimate $\hat{\delta} = \hat{n}_{\mathsf{m}}/n$ as outlined in Section V-E. (For the case where the estimated number of malicious clients is zero, $\hat{n}_{\mathsf{m}} = 0$, we do not run the decoder and flag all clients as benign). On the other hand, the distribution $Q(\boldsymbol{t}|\boldsymbol{s})$, i.e., the noisiness of the test, is test-dependent and difficult to estimate. Here, we assume a simple model for $Q(\boldsymbol{t}|\boldsymbol{s})$ which, as shown in the experiments section (Section VI), yields excellent results. In particular, we assume that $Q(\boldsymbol{t}|\boldsymbol{s})$ factorizes as $Q(\boldsymbol{t}|\boldsymbol{s}) = \prod_{i=1}^{m} Q(t_i|s_i)$ and model $Q(t_i|s_i)$ as a binary symmetric channel (BSC), i.e., $Q(t_i|s_i) = 1 - p$ if $t_i = s_i$ and $Q(t_i|s_i) = p$ if $t_i \neq s_i$. In words, we assume that, for each group, the result of the test is erroneous with probability p.

Our model for $Q(\boldsymbol{t}|\boldsymbol{s})$ requires a single parameter, p. Using the correct value of p improves the decoder performance in

terms of misdetection and false-alarm probabilities. However, even with the proposed simple BSC model, accurately estimating p is challenging. Therefore, we arbitrarily select a value for p and demonstrate that our decoder remains robust to this choice (see Section VI-B). Specifically, we choose a small value for the crossover probability p (as a relatively accurate test is preferred), namely $p = 0.05$.

FedGT-Δ also requires choosing parameter β (see in (16)), which balances the misdetection and false-alarm probabilities. The impact of a higher false alarm or misdetection probability depends on the scenario. For instance, when facing a powerful attack, a near-zero misdetection probability is preferable. Conversely, in heterogeneous settings, a low false alarm is crucial to avoid penalizing correct and unique data points. If prior knowledge of the scenario is available, one can set $\beta < 0.5$ to empashize lowering the false-alarm probability or $\beta > 0.5$ to prioritize reducing the misdetection probability. Here, we assume no prior knowledge and set $\beta = 0.5$, meaning we weight misdetections and false alarms equally.

Overall, since we fix p and (for FedGT-Δ) β independently of the dataset and the nature of the attack, FedGT requires no hyperparameter tuning.

VI. Experiments

A. Setup

We consider a cross-silo scenario with $n = 15$ clients (all participating in each training round) out of which n_{m} are malicious. In Section VI-E, we also provide results for $n = 30$ clients. We remark that these numbers are aligned with current cross-silo applications [21], [22], [43]. The goal of the server is to prevent an attack by identifying the malicious clients and exclude their models from the global aggregation. The experiments are conducted for image classification problems on the MNIST [44], CIFAR-10 [45], and ISIC2019 [46] datasets for which we rely on a single-layer neural network, a ResNet-18 [47], and an Efficientnet-B0 [48] pretrained on Imagenet dataset, respectively.

Similar to previous works [16], [17], [49], [50], [51], we assume that the server has a small validation dataset at its disposal to perform the group tests (the validation dataset is not used for training). Such dataset is not required by FedGT, but is used here due to our choice for the tests in the experiments. The validation dataset should contain data that are sampled from a distribution close to the underlying distribution of the (benign) clients' datasets, i.e., it should be a *quasi-dataset* [16], [49]. For the experiments, we create the validation dataset by randomly sampling 100 data-points from the available data. As a result, the label distribution may not be uniform. For MNIST and CIFAR10, the remaining data points (of size 59900 and 49900) are split evenly at random among the 15 clients, resulting in homogeneous data among the clients, and used for training. We evaluate the performance of FedGT under targeted attacks for scenarios with heterogeneous client data distributions, modeled using a Dirichlet distribution with varying values of α, and show the results in Appendix C. For ISIC2019, we follow [21] and randomly partition the dataset into a training and a test set consisting of 19859 and 3388 samples, respectively. We then partition the training dataset into six parts according to the image acquisition system used to collect the images. Finally, we iteratively split the largest partition in half until we have 15 partitions. This procedure results in a heterogeneous setting where both label distributions and number of samples differ among clients (for the details of the ISIC2019 experiments we refer the reader to Appendix A).

For MNIST, we use the cross-entropy loss and stochastic gradient descent with a learning rate of 0.01, batch size of 64, and number of local epochs equal to 1. For CIFAR-10, we use the cross-entropy loss and stochastic gradient descent with momentum and parameters taken from [1]: the learning rate is 0.05, momentum is 0.9, and the weight decay is 0.001. Furthermore, the batch size is set to 128 and the number of local epochs is set to 5. For ISIC2019, we use the focal loss in [52] and stochastic gradient descent with a learning rate of 0.0005, momentum of 0.9, and weight decay equal to 0.0001. The batch size equals 64 and the number of local epochs is set to 1. Furthermore, we use the same set of augmentations as in [21] to encourage generalization during the training. The results presented are averaged over 10, 5, and 3 runs for MNIST, CIFAR-10, and ISIC2019, respectively.

For the experiments over ISIC2019, due to the heterogeneous client data (see Appendix), the identities of the malicious clients, i.e., the realizations of vector $\boldsymbol{d}$, significantly impact the results. Therefore, for $n_{\mathsf{m}} > 0$, we run the experiments 3 times with different realizations of $\boldsymbol{d}$ but the same client data distribution. In particular, we evaluate three different scenarios: i) the very heterogeneous clients (clients 4 and 10) are not malicious; ii) only one of them is malicious, and iii) both of them are malicious.

In our experiments, we use the test strategy outlined in Section IV-C within FedGT. In particular, for the experiments over MNIST and CIFAR-10, we use $\mathsf{s}^{\text{thres}} = 0.6$ (see (6)) and due to the heterogeneity of ISIC2019, we use $\mathsf{s}^{\text{thres}} = 0$, i.e., the clustering solution is decided solely from the Dunn index.

We show the performance of FedGT using both FedGT-$\hat{n}_{\mathsf{m}}$ and FedGT-Δ. We compare their performance to four benchmarks: "no defense", "oracle", RFA [26] and Multi-Krum [12] (MKrum). The no defense benchmark corresponds to plain FL including all clients, i.e., disregarding some clients may be malicious, while the oracle is an ideal setting where the server knows the malicious clients and discards them. Note that RFA belongs to a short list of defense mechanisms that also provide privacy. Multi-Krum is a defense mechanism that assumes a large Euclidean distance between malicious and benign models. Though initially non-private, it can leverage tools from secret sharing for privacy [25]. However, it also requires prior knowledge of the number of malicious clients, n_{m}, which is not feasible in practice. In this work, Multi-Krum is provided the true value of n_{m} and therefore we do not perform experiments for $n_{\mathsf{m}} = 0$.

To demonstrate the effectiveness of FedGT, we perform the group testing step only once during the training. This constitutes the weakest version of our framework as the group testing may be performed in each round at the expense of increased communication cost (see Section IV-D). In particular, for MNIST, we perform the group testing in the first round

and for CIFAR-10 and ISIC2019, in the fifth round. We pick as the assignment matrix a parity-check matrix of a BCH code [53] of length 15 and redundancy 8, meaning that we create a group testing scheme where the 15 clients are pooled into 8 groups, each containing 4 clients. This choice of $\boldsymbol{A}$ allows for $n_{\mathsf{m}}^{\max} = 5$, where κ in (15) is set to 20%. Also, as discussed in Section V-E we set $\beta = 0.5$ in (16) for FedGT-Δ. Tuning β may yield better performance, especially for the experiments over ISIC2019, but would require prior knowledge. Hence, we only restrict our focus to the general solution, where $\beta = 0.5$.

For the considered setup, FedGT yields a communication overhead and privacy guarantee as follows.

- **Communication cost.** Considering a secure aggregation scheme with linear communication complexity such as LightSecAgg [38], the communication cost of the group testing round is approximately $2\times$ the complexity of secure aggregation with 15 clients. Compared to RFA, which requires $3\times$ communication cost of secure aggregation with 15 clients *in each round*, FedGT yields a significantly reduced communication cost. The communication cost of the private version of Multi-Krum [25] from the server side is only slightly higher than that of secure aggregation, but from the client-side is much higher, namely scales linearly with the number of clients. Moreover, Multi-Krum has to be performed at every round, which makes it computationally more expensive throughout the learning.
- **Privacy.** With our choice of assignment matrix, FedGT guarantees the same level of privacy of full secure aggregation with 4 clients. This stems from the property that any linear combination of the server's group aggregates leads to an aggregation involving no fewer than 4 client models, as elucidated in Proposition 1.

B. Robustness Toward the Crossover Probability p

The decoding strategy employed in FedGT-Δ requires selecting the optimal parameter Δ, which, due to the trellis-based decoding approach, depends on the unknown crossover probability p. In our experiments, we set $p = 0.05$. Next, we empirically demonstrate that FedGT-Δ is robust to a mismatch in the assumed p.

In Table II (upper half), we present the value of the objective $0.5\ P_{\mathsf{MD}} + 0.5\ P_{\mathsf{FA}}$ for different values of p and n_{m} when Δ is obtained via (16) and $p = 0.05$ is believed to be the true value, i.e., we assess the impact of a mismatch in p. It can be seen that the objective function is robust to a mismatch in p for all $n_{\mathsf{m}} \in [n_{\mathsf{m}}^{\max}]$. Hence, p may be chosen to hedge for the anticipated noise in the testing strategy and one does not have to be concerned about the impact of a mismatch on the choice of Δ. Similarly, we conducted a robustness analysis of FedGT-$\hat{n}_{\mathsf{m}}$ with respect to the crossover probability p in terms of misdetection and false-alarm probabilities, and tabulated the results in the bottom half of Table II. The analysis shows that FedGT-$\hat{n}_{\mathsf{m}}$ is robust to variations in p: for $n_{\mathsf{m}} \in [3]$ for $p \leq 20\%$, the values of the objective $0.5 \cdot (P_{\mathsf{MD}} + P_{\mathsf{FA}})$ are constant up two 2 decimal digits. For $n_{\mathsf{m}} = 4$ and 5, the values of the objective differ by at most 0.01. It is important to note that the optimization of β is relevant only for FedGT-Δ (see (16)), and we use the same $\beta = 0.5$ for the sake of comparison and generality.

TABLE II

ROBUSTNESS OF THE OBJECTIVE, $0.5\ P_{\mathsf{MD}} + 0.5\ P_{\mathsf{FA}}$, WITH VARYING p FOR A Δ OBTAINED FROM (16) WITH $p = 5\%$ FOR FEDGT-Δ STRATEGY (UPPER HALF). SIMILARLY, THE SAME ANALYSIS IS PERFORMED FOR THE OTHER STRATEGY, FEDGT-$\hat{n}_{\mathsf{m}}$ (BOTTOM HALF)

n_{m} \ p	1%	2.5%	5%	7.5%	10%	12.5%	15%	17.5%	20%
FedGT-Δ strategy									
1	0	0	0	0	0	0	0	0	0
2	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.02
3	0.07	0.07	0.07	0.07	0.07	0.07	0.06	0.06	0.06
4	0.14	0.14	0.14	0.14	0.14	0.14	0.15	0.15	0.16
5	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.17
FedGT-$\hat{n}_{\mathsf{m}}$ strategy									
1	0	0	0	0	0	0	0	0	0
2	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
4	0.09	0.09	0.09	0.09	0.09	0.09	0.10	0.09	0.09
5	0.13	0.14	0.13	0.13	0.14	0.14	0.14	0.13	0.13

C. Experimental Results for Targeted Attacks

For targeted data-poisoning, we consider offline label-flipping attacks. We refer to the attacked label as the source label and the resulting label after the flip as the target label. For MNIST, we consider malicious clients to flip source label 1 into target label 7. As such, the objective of the malicious clients is to cause the global model to misclassify 1's into 7's. Similarly, for CIFAR-10, malicious clients change source label 7, i.e., horse, into target label 4, i.e., deer. For the ISIC2019 dataset, malicious clients mislabel source label 0, i.e., melanoma, into target label 1, i.e., mole. Note that this attack has a significant medical impact, as the goal of the attacker is to force the model to classify cancer into non-cancer. Since the adversary's goal is not to deteriorate the global model but to make it misinterpret the source label as the target label, we adopt the *attack accuracy* as the primary metric of interest. The attack accuracy is defined as the fraction of source labels classified as the target label in the test dataset. Moreover, as a successful defense mechanism should not compromise the overall utility of the model, we employ the accuracy on the test dataset as a secondary performance metric. While an online targeted data poisoning attack is not realistic in cross-silo FL settings, we conduct experiments to evaluate its impact. Details of the attack strategy and the corresponding results are provided in Appendix B.

For the utility metric adopted in the testing strategy (see Section IV-C), we consider the source label recall, i.e., the fraction of source labels classified into the correct label, to flag test groups containing malicious clients and perform PCA on the weights of the fully connected layer incoming to the source label. We remark that to identify a label under attack, one may simply monitor, e.g., the recall of each label in the different test groups, using the validation dataset. For a targeted attack, the test noisiness obtained from our experiments is 5.41%, 9.16%, and 19.53% for MNIST, CIFAR-10, and ISIC2019, respectively (we recall that we used $p = 0.05$ in all our experiments). At first glance, the test results for ISIC2019 appear to be very noisy. However, we note that, due to the

TABLE III

ATTACK ACCURACY (ATT) AND BALANCED ACCURACY (ACC) MEASURED AFTER SPECIFIED COMMUNICATION ROUNDS FOR MNIST, CIFAR10, AND ISIC2019 DATASETS. ALL ENTRIES ARE PROVIDED AS MEAN AND STANDARD DEVIATION WITH VALUES IN %

	Oracle		RFA [26]		MKrum [12]		FedGT-$\hat{n}_{\mathsf{m}}$		FedGT-Δ		No defense	
n_{m}	ATT ↓	ACC ↑	ATT ↓	ACC ↑	ATT ↓	ACC ↑	ATT ↓	ACC ↑	ATT ↓	ACC ↑	ATT ↓	ACC ↑
MNIST (10 communication rounds)												
0	0.07 ± 0.07	90.32 ± 0.09	0.07 ± 0.07	90.32 ± 0.10	–	–	0.08 ± 0.06	90.18 ± 0.10	0.08 ± 0.06	90.18 ± 0.11	0.07 ± 0.07	90.32 ± 0.09
1	0.04 ± 0.04	90.32 ± 0.17	0.10 ± 0.07	90.32 ± 0.10	0.06 ± 0.06	90.19 ± 0.11	0.05 ± 0.06	90.21 ± 0.09	0.05 ± 0.06	90.19 ± 0.10	0.15 ± 0.04	90.30 ± 0.17
2	0.04 ± 0.04	90.33 ± 0.19	0.13 ± 0.06	90.31 ± 0.12	0.06 ± 0.06	90.19 ± 0.12	0.06 ± 0.07	90.17 ± 0.13	0.07 ± 0.07	90.16 ± 0.12	0.18 ± 0.04	90.23 ± 0.15
3	0.04 ± 0.04	90.31 ± 0.17	0.15 ± 0.04	90.29 ± 0.10	0.05 ± 0.06	90.17 ± 0.11	0.06 ± 0.07	90.12 ± 0.09	0.07 ± 0.07	90.13 ± 0.08	0.36 ± 0.13	90.10 ± 0.16
4	0.04 ± 0.04	90.32 ± 0.16	0.16 ± 0.04	90.29 ± 0.09	0.08 ± 0.06	90.16 ± 0.12	0.19 ± 0.13	90.05 ± 0.13	0.07 ± 0.05	90.11 ± 0.11	1.03 ± 0.23	89.89 ± 0.19
5	0.04 ± 0.04	90.34 ± 0.16	0.17 ± 0.03	90.26 ± 0.10	0.04 ± 0.06	90.16 ± 0.11	1.53 ± 2.75	89.77 ± 0.45	0.07 ± 0.05	90.07 ± 0.10	3.15 ± 0.40	89.49 ± 0.18
CIFAR10 (30 communication rounds)												
0	4.10 ± 0.27	81.66 ± 0.16	3.86 ± 0.27	81.94 ± 0.28	–	–	4.12 ± 0.28	81.49 ± 0.40	4.28 ± 0.49	81.23 ± 0.91	4.10 ± 0.27	81.66 ± 0.16
1	3.36 ± 0.56	81.69 ± 0.22	5.44 ± 0.67	81.65 ± 0.20	7.62 ± 1.87	81.35 ± 0.18	4.84 ± 1.87	81.07 ± 0.73	4.40 ± 1.35	80.41 ± 2.22	5.72 ± 0.68	81.45 ± 0.06
2	4.10 ± 0.83	81.44 ± 0.22	7.74 ± 1.84	81.49 ± 0.39	11.1 ± 4.67	80.60 ± 0.45	4.82 ± 2.55	80.83 ± 0.41	4.54 ± 1.75	78.46 ± 1.97	9.62 ± 1.72	81.11 ± 0.30
3	3.56 ± 0.32	81.13 ± 0.32	11.06 ± 0.62	81.03 ± 0.21	14.9 ± 8.03	80.23 ± 0.87	4.32 ± 2.31	80.82 ± 0.43	4.92 ± 1.23	79.01 ± 2.22	17.62 ± 2.23	80.12 ± 0.55
4	3.94 ± 1.07	81.07 ± 0.18	16.92 ± 3.07	80.52 ± 0.56	40.24 ± 21.8	77.21 ± 2.09	10.7 ± 5.53	80.28 ± 0.70	4.90 ± 1.09	78.68 ± 1.61	26.42 ± 2.18	79.25 ± 0.31
5	3.74 ± 0.43	80.54 ± 0.11	25.16 ± 3.94	79.67 ± 0.37	37.50 ± 15.5	77.55 ± 1.50	18.62 ± 7.87	79.54 ± 0.76	5.32 ± 1.19	76.53 ± 0.47	38.40 ± 6.48	78.12 ± 0.64
ISIC2019 (40 communication rounds)												
0	25.04	63.79	21.72	64.96	–	–	16.09	62.91	15.92	60.70	25.87	63.29
1	21.72 ± 1.76	63.14 ± 0.24	23.27 ± 0.83	63.24 ± 1.61	21.17 ± 2.68	62.27 ± 0.25	16.97 ± 0.16	63.28 ± 0.73	17.69 ± 0.75	62.17 ± 1.45	25.43 ± 2.78	62.00 ± 0.61
2	21.23 ± 2.51	63.19 ± 0.81	24.71 ± 1.86	62.63 ± 1.37	20.29 ± 1.34	61.97 ± 0.28	18.79 ± 1.96	63.12 ± 0.38	19.07 ± 1.64	62.47 ± 0.97	26.70 ± 3.16	62.52 ± 0.51
3	21.28 ± 2.54	62.16 ± 1.48	29.30 ± 3.05	62.63 ± 0.21	23.33 ± 3.14	61.04 ± 0.29	18.74 ± 2.84	62.89 ± 0.75	19.13 ± 2.52	58.47 ± 4.05	30.51 ± 4.62	61.92 ± 0.68
4	20.18 ± 2.13	61.65 ± 0.53	31.29 ± 2.25	62.10 ± 1.10	25.21 ± 0.23	57.80 ± 0.18	18.57 ± 1.88	62.58 ± 0.18	19.90 ± 3.18	56.15 ± 4.32	34.11 ± 2.55	61.50 ± 1.02
5	20.01 ± 1.85	61.01 ± 0.40	38.47 ± 3.52	61.73 ± 1.15	33.89 ± 0.87	58.07 ± 0.31	22.66 ± 0.55	61.41 ± 0.19	17.69 ± 0.86	54.90 ± 2.85	38.70 ± 3.57	60.30 ± 1.40

high heterogeneity, some benign clients may actually harm the model due to their data distribution, even without containing poisoned data. FedGT identifies some of these clients as malicious—thus yielding higher utility—, which explains the higher noisiness of the test results. To allow for a comparison between our two schemes on a given dataset with respect to P_{MD} and P_{FA}, we empirically evaluate $\beta P_{\mathsf{MD}} + (1-\beta) P_{\mathsf{FA}}$ for $\beta = 0.5$ (we optimized (16) for $\beta = 0.5$) by averaging our experimental results over both independent runs and over n_{m}. For FedGT-$\hat{n}_{\mathsf{m}}$, we obtained 8.13%, 8.78% and 8.83%, and for FedGT-Δ, we obtained 10.54%, 13.80% and 19.83% for the MNIST, CIFAR-10 and ISIC2019 datasets, respectively. Hence, based on this metric, Fed-$\hat{n}_{\mathsf{m}}$ is the preferred version. Recall that FedGT-Δ does not limit the number of clients flagged malicious, i.e., the $w_{\mathsf{H}}(\hat{\boldsymbol{d}})$ and risks trading higher false-alarm for lower misdetection.

In Table III, we give the attack accuracy and top-1 (or balanced) accuracy of FedGT-$\hat{n}_{\mathsf{m}}$ and FedGT-Δ. For comparison, we also provide results for no defense, oracle, RFA [26] and Multi-Krum [12]. The results are shown as the mean and standard deviation in % obtained from a Monte-Carlo-based simulation approach. However, please note that the experiment over ISIC2019 for $n_{\mathsf{m}} = 0$ is performed only once, as we do not investigate client data distribution other than the one depicted in Appendix A.

For MNIST, we observe a modest impact of the label flip, even for $n_{\mathsf{m}} = 5$. Nevertheless, FedGT-Δ effectively mitigates the attack accuracy compared to no defense. Notably, it significantly outperforms RFA (which lacks the capability of identifying malicious clients and entails a much larger communication complexity) and performs close to the oracle. FedGT-$\hat{n}_{\mathsf{m}}$ outperforms RFA for $1 \leq n_{\mathsf{m}} \leq 3$, but falls short for other values of n_{m}. Multi-Krum [12] outperforms both versions of FedGT for $n_{\mathsf{m}} \geq 3$ (FedGT-Δ only slightly). However, we stress that the results for Multi-Krum are obtained by allowing the aggregator to know the number of malicious clients n_{m}, which is infeasible in practice.

For CIFAR10, the label flip attack has a significant impact, as can be seen from the no-defense attack accuracy, nearing 40% for $n_{\mathsf{m}} = 5$. We observe that Multi-Krum performs very poorly in terms of attack accuracy for all $n_{\mathsf{m}} \geq 1$. Both versions of FedGT significantly outperform RFA in terms of attack accuracy for all $n_{\mathsf{m}} \geq 1$, especially for larger values of n_{m}, with FedGT-Δ performing very close to the oracle. For example, for $n_{\mathsf{m}} = 5$, FedGT-n_{m} and FedGT-Δ reduce the attack accuracy to 18.32% and 5.32%, respectively, compared to 25.16% RFA. (We note that the pronounced reduction in attack accuracy by FedGT-Δ is achieved at the expense of a slight penalty in accuracy for larger values of n_{m}).

For ISIC2019, RFA performs poorly, achieving only a small improvement in attack accuracy with respect to no defense (RFA is known to underperform for heterogeneous data across clients [28]). Similary, Multi-Krum performs slightly better than RFA but still poorly, due to the heterogeneous data distribution. Both versions of FedGT significantly diminish the attack accuracy, even outperform the oracle. This can be explained from the data heterogeneity across clients where some clients, although not malicious, will be biased to output a given label, e.g., client 4 and client 10 (see Appendix A). Hence, due to the testing strategy, FedGT may identify benign clients exhibiting extreme heterogeneity as malicious to be removed from the training, ultimately reducing the attack accuracy and benefiting the overall utility of the global model. Compared to the experiments on MNIST and CIFAR10, FedGT-$\hat{n}_{\mathsf{m}}$ yields the strongest performance over the two metrics. This is again attributed to heterogeneity, as FedGT-Δ removes too many clients, resulting in a high false-alarm probability.

In Fig. 3, we plot the attack accuracy of the label-flip attack over communication rounds for different values of n_{m}. From the CIFAR10 and ISIC2019 experiments, the impact of the group testing is clearly seen with the attack accuracy rapidly dropping in round 5.

D. Experimental Results for Untargeted Attacks

Next, we consider a label permutation attack where malicious clients offset their data labels by 1, i.e., $L_{\text{new}} = (L_{\text{old}} +$

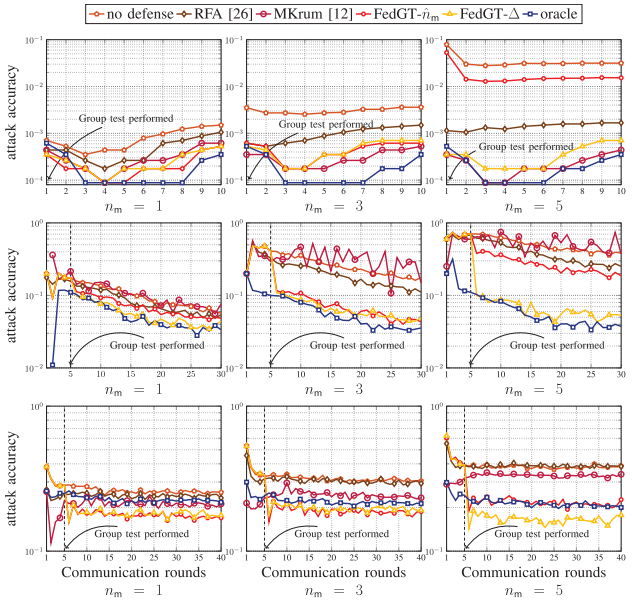

Fig. 3. Average attack accuracy on the MNIST (row 1), CIFAR10 (row 2) and ISIC2019 (row 3) datasets for varying number of malicious clients. These results are obtained from FL experiments where n_m clients out of $n = 15$ total clients act as malicious by deploying a label-flip attack.

TABLE IV

TOP-1 OR BALANCED ACCURACY (ACC) MEASURED AFTER SPECIFIED COMMUNICATION ROUNDS FOR MNIST, CIFAR10, AND ISIC2019 DATASETS, FOR EXPERIMENTS WITH UNTARGETED ATTACKS. ALL ENTRIES ARE PROVIDED AS MEAN AND STANDARD DEVIATION WITH VALUES IN %

	Oracle	RFA [26]	MKrum [12]	FedGT-$\hat{n}_m$	FedGT-Δ	No defense
n_m	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ACC ↑
MNIST (10 communication rounds)						
0	90.18 ± 0.10	90.18 ± 0.09	–	90.18 ± 0.10	90.18 ± 0.10	90.18 ± 0.10
1	90.19 ± 0.10	90.22 ± 0.11	90.19 ± 0.11	90.21 ± 0.09	90.19 ± 0.10	89.96 ± 0.08
2	90.18 ± 0.12	90.20 ± 0.11	90.19 ± 0.12	90.14 ± 0.19	90.10 ± 0.18	89.01 ± 0.11
3	90.18 ± 0.09	90.17 ± 0.09	90.17 ± 0.11	89.94 ± 0.28	90.08 ± 0.16	87.52 ± 0.13
4	90.16 ± 0.10	90.17 ± 0.09	90.16 ± 0.12	88.72 ± 1.20	89.75 ± 1.06	85.06 ± 0.21
5	90.17 ± 0.12	90.16 ± 0.09	90.16 ± 0.11	84.96 ± 5.20	89.61 ± 1.34	80.36 ± 0.26
CIFAR10 (30 communication rounds)						
0	81.66 ± 0.16	81.94 ± 0.28	–	81.40 ± 0.48	80.47 ± 2.32	81.66 ± 0.16
1	81.94 ± 0.27	81.64 ± 0.19	81.51 ± 0.21	81.64 ± 0.34	81.67 ± 0.36	81.49 ± 0.24
2	81.60 ± 0.15	81.40 ± 0.21	81.26 ± 0.27	81.12 ± 0.45	80.77 ± 0.71	80.73 ± 0.08
3	81.28 ± 0.17	81.13 ± 0.31	81.15 ± 0.26	80.90 ± 0.40	79.78 ± 1.86	77.73 ± 2.91
4	80.99 ± 0.30	79.78 ± 0.47	80.65 ± 0.22	80.40 ± 0.46	78.92 ± 1.75	56.49 ± 1.82
5	80.94 ± 0.39	78.52 ± 2.43	37.37 ± 35.2	71.71 ± 8.70	76.79 ± 0.76	49.07 ± 19.31
ISIC2019 (40 communication rounds)						
0	61.88	61.26	–	62.70	62.74	61.88
1	61.63 ± 1.03	62.07 ± 0.60	61.89 ± 0.65	61.80 ± 0.45	63.13 ± 0.78	61.03 ± 1.17
2	62.53 ± 1.40	62.86 ± 0.66	60.73 ± 2.04	63.58 ± 0.19	62.48 ± 1.66	59.13 ± 1.06
3	61.84 ± 1.80	60.15 ± 1.55	56.35 ± 1.72	61.48 ± 1.46	57.01 ± 4.03	54.02 ± 1.51
4	61.34 ± 0.80	58.87 ± 1.17	51.37 ± 0.55	58.88 ± 3.27	53.27 ± 2.85	49.75 ± 0.80
5	58.87 ± 0.19	52.34 ± 0.64	40.79 ± 1.49	55.47 ± 0.71	50.12 ± 1.89	42.64 ± 1.96

1) mod n_c, where n_c is the number of classes. The attack aims at deteriorating the classification accuracy over all labels, i.e., an attacker wants to lower the top-1 accuracy (MNIST and CIFAR-10) or the balanced accuracy (ISIC2019). For this reason, we use the top-1 accuracy (MNIST and CIFAR10) and the balanced accuracy (ISIC2019) on the test groups' aggregates as the qualitative metric in the testing strategy (see Section VI-A) and perform PCA on the flattened weights of the entire fully connected layer. The balanced accuracy is the average recall per class, used to take into account class imbalances, as in the case of ISIC2019 [21]. The test error probability is 2.29%, 4.58%, and 3.91% for experiments over MNIST, CIFAR-10 and ISIC2019, respectively. As in Section VI-C, we evaluate $\beta P_{\mathsf{MD}} + (1-\beta) P_{\mathsf{FA}}$ for $\beta = 0.5$ from the untargeted attack experiments. For FedGT-$\hat{n}_m$, we obtain $6.93\%, 7.05\%$ and 6.12% and for FedGT-Δ we obtain 8.55%, 10.33% and 11.50% for MNIST, CIFAR-10 and ISIC2019, respectively. Similarly to Section VI-C, FedGT-$\hat{n}_m$ achieves lower scores, suggesting it to be the preferred version.

In Table IV, we show the top-1 accuracy versus n_m for MNIST and CIFAR-10, and the balanced accuracy for ISIC2019. The results are tabulated as the mean and standard deviation in %. For all cases, with no defense, a significant drop in accuracy is observed as the number of malicious clients grows. For MNIST, FedGT-Δ achieves similar performance to RFA, Multi-Krum, and oracle for all considered n_m. On the other hand, FedGT-$\hat{n}_m$ performs comparably to the other defenses for $n_m \leq 3$, but its performance declines for $n_m = 4$ and $n_m = 5$. For CIFAR-10, both versions of FedGT perform similar to RFA for $n_m \leq 4$, but worse for $n_m = 5$. The robust performance of RFA is anticipated due to the untargeted attack rendering malicious client models significantly different from benign ones given the i.i.d. data distribution across clients. Consequently, the geometric median—essentially performing a majority vote—assigns the malicious models a very low weight. Multi-Krum performs very good for $n_m \leq 4$, but its performance degrades drastically at $n_m = 5$.

For ISIC2019, FedGT-$\hat{n}_m$ performs better than RFA and Multi-Krum for all values of n_m, except $n_m = 1$. Moreover, for $n_m = 0$ and $n_m = 2$, FedGT-$\hat{n}_m$ performs even better than oracle due to the heterogeneity of the data distribution among clients. This means that some clients can be flagged as malicious just because they deteriorate the utility of the global model due to their data samples. FedGT-Δ performs better or similar to RFA for $n_m \leq 2$ but worse than RFA for $n_m \geq 3$, while it outperforms Multi-Krum for all $n_m \in [5]$. We note that this result is due to some realizations of defective vectors triggering the decoder to falsely flag as malicious clients with more homogeneous data distribution and resort to learning with clients with heterogeneous data.

In Fig. 4, we plot the top-1 accuracy for MNIST and CIFAR-10 and the balanced accuracy for ISIC2019 over different communication rounds. For $n_m = 1$, the attack is not very powerful (regardless of the dataset), and the no defense and oracle benchmarks have similar performance. For $n_m \in \{3, 5\}$, the impact on the top-1 accuracy of the attack for MNIST and CIFAR-10 is significant, as shown by the significant gap between the no defense and oracle curves. FedGT-Δ closes this gap, but RFA outperforms our strategy, due to its majority decision-based aggregation technique. For $n_m = 5$, FedGT-$\hat{n}_m$ suffers compared to the other techniques for $n_m = 5$. This is due to its reliance on an accurate estimate $\hat{n}_m$, something that becomes harder with a larger n_m as more test groups are contaminated. For the ISIC2019 dataset, FedGT-$\hat{n}_m$ outperforms RFA for $n_m = 3, 5$, while FedGT-Δ performs poorly for $n_m \geq 3$. This occurs due to the heterogeneity of ISIC2019 where a false-alarm incurs a significant penalty on the global model. Nevertheless, FedGT-Δ outperforms Multi-Krum for all n_m.

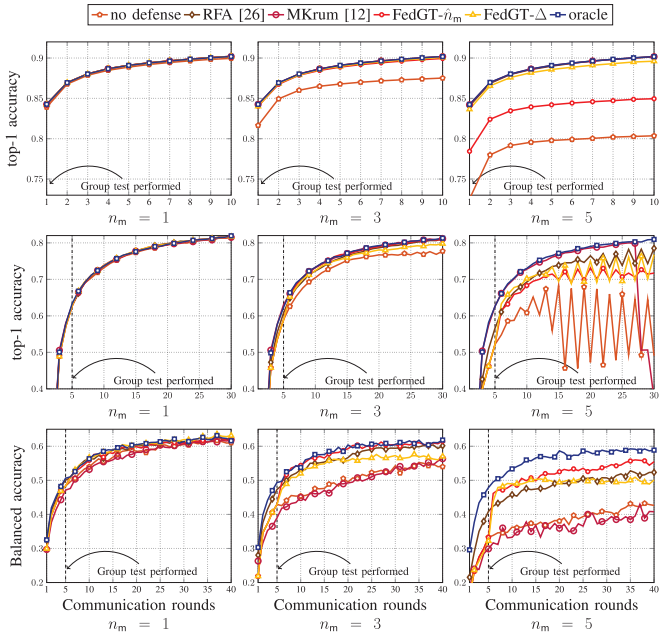

Fig. 4. Average top-1 accuracy on the MNIST (row 1), CIFAR10 (row 2) and ISIC2019 (row 3) datasets for varying n_m.

Finally, we observe an interesting phenomenon for the experiments over the CIFAR-10 dataset. For $n_m = 5$, the no defense curve exhibits significant fluctuations throughout the rounds. Although the performance of FedGT-$\hat{n}_m$ also fluctuates, it does so to a significantly lesser extent, while the fluctuations are more pronounced in FedGT-Δ and RFA.

E. Federated Learning With More Clients

Hitherto, the experiments have focused on a cross-silo FL scenario with 15 clients. Next, we investigate the performance of FedGT for a cross-silo FL scenario with a larger number of clients, specifically $n = 30$ clients. For this scenario, we choose as the assignment matrix the parity-check matrix of a $(30, 18)$ cyclic code of length 30 and dimension 18, resulting in 12 groups, each containing 6 clients. The dual of this cyclic code has minimum Hamming distance 6, thus FedGT preserves the same clients' privacy of secure aggregation with 6 clients. This choice of $\boldsymbol{A}$ allows for $n_m^{\max} = 8$, where the probability in (4) is constrained to 20%, i.e., $\kappa = 0.2$.

We investigate a scenario with $n_m = 6$ malicious clients and both a targeted attack and an untargeted attack. We conduct experiments on the MNIST and CIFAR-10 datasets, with the same hyperparameters as specified in Section VI-A. Due to the relatively high number of clients, we do not run the experiments over ISIC2019, as this dataset is tailored to FL scenarios with a smaller number of clients.

In Fig. 5, we plot the attack accuracy of the targeted attack (row 1) and the top-1 accuracy of the untargeted attack (row 2), respectively. For a targeted attack, FedGT-Δ performs very close to the oracle and Multi-Krum (for MNIST) and outperforms RFA for both datasets and Multi-Krum for CIFAR-10, with the improvement in performance being significant for the CIFAR-10 dataset compared to Multi-Krum and RFA. For an untargeted attack, FedGT-Δ performs similar to RFA, Multi-Krum and oracle.[4]

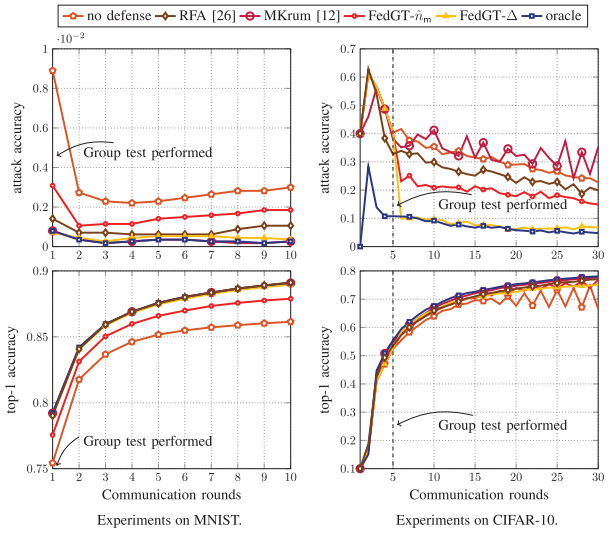

Fig. 5. Experimental results for a federated learning with $n = 30$ clients out of which $n_m = 6$ of them are malicious. The attack accuracy of a targeted attack is shown in row 1 and the top-1 accuracy of an untargeted attack strategy is shown in row 2. We show the performance of FedGT along with no defense, oracle, RFA [26] and Multi-Krum (MKrum) [12].

VII. Conclusion

We proposed FedGT, a novel and flexible framework for identifying malicious clients in FL that is compatible with secure aggregation and does not require hyperparameter tuning. By grouping clients into overlapping groups, FedGT enables the identification of malicious clients at the expense of secure aggregation involving fewer clients. Experiments conducted in a cross-silo scenario for different data-poisoning attacks demonstrate the effectiveness of FedGT in identifying malicious clients, resulting in high model utility and low attack accuracy. Remarkably, FedGT significantly outperforms the recently-proposed robust federated aggregation (RFA) protocol based on the geometric median (which is unable to identify malicious clients and entails a much higher communication cost) and the well-known robust aggregation technique Multi-Krum (even though Multi-Krum assumes the unrealistic prior knowledge of the number of malicious clients) across multiple scenarios. To the best of our knowledge, this is the first work that provides a solution for identifying malicious clients in FL with secure aggregation.

In this paper, we focused on the cross-silo federated learning scenario, where (offline) data poisoning attacks are particularly prevalent, and the number of clients is relatively small (fewer than 50). Extending FedGT to a cross-device setting, involving thousands or more clients, presents significant challenges. Specifically, the optimal decoder employed in our approach has exponential complexity with the number of tests, rendering it impractical for large-scale cross-device federated learning. Future work could explore scalable adaptations of FedGT to address these challenges, such as leveraging suboptimal decoding strategies like belief propagation.

[4]The source code is available at https://github.com/johanos1/FedGT

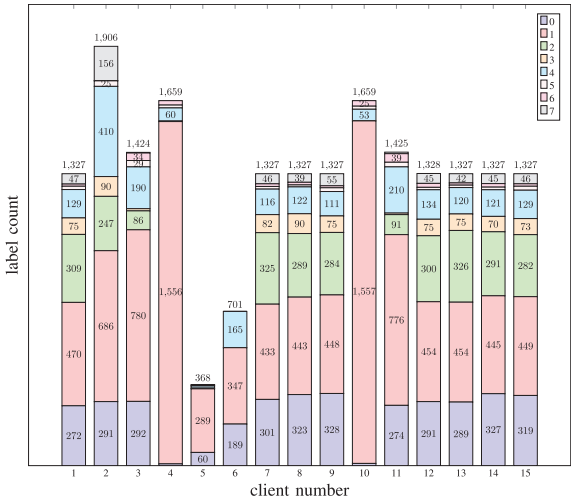

Fig. 6. Client data partitions for experiments over the ISIC2019 dataset.

APPENDIX A
DETAILS OF THE ISIC2019 EXPERIMENT

The ISIC2019 dataset [46] is a public dataset consisting of images of various skin lesion types, including malignant melanomas and benign moles, used for research in dermatology and skin cancer detection. We use the ISIC2019 dataset and follow [21] by first restricting our usage to 23247 data samples (out of 25331 entries) from the public dataset due to metadata availability and then preprocessing by resizing the shorter side to 224 pixels and normalizing the contrast and brightness of the images. Next, we randomly divide the data into a test and a training dataset of size 3388 and 19859, respectively. The server validation set is created by randomly sampling 100 data entries from the dataset. Next, as in [21], the remaining 19759 samples are partitioned into 6 partitions with respect to the image acquisition system used. The 6 partitions are then split into 15 partitions by iteratively splitting the largest partition in half. This procedure results in partitions with heterogeneity in the number of data samples and label distribution (see Fig. 6), and in the feature distribution due to different acquisition systems (see [21, Fig. 1.f]).

Due to the large imbalance in the dataset (label 1 corresponds to 48.7% whereas label 5 and 6 are represented by about 1% of the entries), the focal loss is used in the training [52] and we use the balanced accuracy to assess the performance of the trained network. Furthermore, to encourage generalization during training, we follow [21, App. H] and apply random augmentations to the training data.

Finally, the heterogeneous data partitioning causes the choice of malicious clients to significantly impact the outcome of the experiment. For this reason, we let the set of malicious clients $\mathcal{M}_i \subset \mathcal{M}_j$ for $j > i$ to ensure that results across different values of n_{m} are comparable.

APPENDIX B
ONLINE TARGETED DATA POISONING ATTACKS

Online targeted data poisoning attacks are a significant threat, capable of substantially degrading the utility of global models. While such attacks are not realistic in cross-silo FL, in this appendix we demonstrate the resiliency of FedGT against this type of attack. Inspired by the online data poisoning strategy described in [22], we implement the following online targeted data poisoning attack: A malicious client receives the global model from the server and generates the logit values on its (unpoisoned) dataset. The attacker then targets a specific class label (source) and constructs a poisoned dataset as follows: *i)* if a data sample is classified correctly by the global model, the poisoned label is set to the class with the second-largest logit value; *ii)* if a data sample is misclassified, the malicious client alters the original label to the most likely label predicted by the global model.

The goal of this attack is to undermine the global model's ability to correctly classify the source label. The first poisoning rule targets the model when it classifies correctly, aiming to confuse it, while the second rule reinforces the model's confidence in its misclassifications. This attack is applied in every communication round, with the malicious clients dynamically updating their poisoned labels in each round.

We conduct experiments to evaluate the performance of FedGT against this online targeted data poisoning attack, with the results summarized in Table V. We perform experiments on the MNIST, CIFAR-10 and ISIC2019 datasets. For the source labels, we select label 1 for MNIST, label 7 (horse) for CIFAR-10, and label 0 (melanoma) for ISIC2019. Since the attacker aims to disrupt the model's classification ability for a specific label, we use the recall of the source label as the primary evaluation metric. Additionally, as with the results in Table III, we report the top-1 accuracy for the balanced datasets (MNIST, CIFAR10) and the balanced accuracy for ISIC2019 to account for its class imbalance. We compare the performance of both FedGT variants—FedGT-$\hat{n}_{\mathsf{m}}$ and FedGT-Δ—against no defense, oracle, and the two robust aggregation techniques RFA [26] and Multi-Krum [12]. For both FedGT versions, we apply the same testing algorithm as explained in Section VI-C.

Experiments on MNIST show that, in general, Multi-Krum performs as the most effective defense mechanism (we recall that Multi-Krum requires the knowledge of n_{m}, which is unrealistic); however, FedGT-Δ remains competitive across all $n_{\mathsf{m}} \in [5]$. In contrast, FedGT-$\hat{n}_{\mathsf{m}}$ demonstrates strong performance for $n_{\mathsf{m}} \in [3]$, and performs very close to the oracle for $n_{\mathsf{m}} = 1$. Notably, FedGT-Δ performs very close to oracle in terms of recall but suffers a noticeable drop in top-1 accuracy for $n_{\mathsf{m}} \geq 3$. On CIFAR-10, both versions of FedGT outperform Multi-Krum and RFA for all values of n_{m}. In fact, Multi-Krum performs even worse than no defense, highlighting its limitations. For the ISIC2019 dataset, both versions of FedGT outperform RFA and Multi-Krum for all values of n_{m}. Moreover, FedGT-Δ performs better than oracle, while FedGT-$\hat{n}_{\mathsf{m}}$ outperforms oracle for $n_{\mathsf{m}} \in [4]$. This can be attributed to the heterogeneous data distribution among clients, as shown in Fig. 6. Some highly heterogeneous clients have a deteriorating effect on the global model despite being benign. FedGT effectively mitigates this issue by flagging these clients as malicious, thus limiting their impact. It is important to note that while FedGT-Δ achieves a good source recall, it comes at the cost of reduced global balanced accuracy. However, in this experiment, the source label is "melanoma", which

TABLE V

RECALL OF THE SOURCE LABEL (REC) AND TOP-1, (MNIST, CIFAR10) OR BALANCED ACCURACY (ISIC2019) (SHOWN AS ACC) MEASURED AFTER SPECIFIED COMMUNICATION ROUNDS FOR MNIST, CIFAR10, AND ISIC2019 DATASETS. ALL ENTRIES ARE PROVIDED AS MEAN AND STANDARD DEVIATION WITH VALUES IN %

	Oracle		RFA [26]		M-Krum [12]		FedGT-$\hat{n}_{\mathsf{m}}$		FedGT-Δ		No defense	
n_{m}	REC ↑	ACC ↑	REC ↑	ACC ↑	REC ↑	ACC ↑	REC ↑	ACC ↑	REC ↑	ACC ↑	REC ↑	ACC ↑
MNIST (10 communication rounds)												
1	96.58 ± 0.10	90.19 ± 0.10	96.41 ± 0.14	90.19 ± 0.10	96.56 ± 0.17	90.19 ± 0.11	96.58 ± 0.11	90.21 ± 0.11	96.57 ± 0.11	90.19 ± 0.10	95.84 ± 0.13	90.24 ± 0.10
2	96.55 ± 0.12	90.18 ± 0.12	96.20 ± 0.13	90.22 ± 0.12	96.57 ± 0.13	90.19 ± 0.12	96.46 ± 0.31	90.17 ± 0.12	96.33 ± 0.31	90.19 ± 0.13	94.41 ± 0.23	90.20 ± 0.11
3	96.57 ± 0.10	90.18 ± 0.09	95.90 ± 0.18	90.24 ± 0.09	96.55 ± 0.16	90.17 ± 0.11	96.17 ± 0.69	90.16 ± 0.10	96.35 ± 0.55	90.15 ± 0.08	92.26 ± 0.30	90.05 ± 0.12
4	96.53 ± 0.13	90.16 ± 0.10	95.30 ± 0.28	90.24 ± 0.13	96.49 ± 0.17	90.16 ± 0.12	94.40 ± 1.79	90.12 ± 0.11	96.06 ± 1.54	90.12 ± 0.11	89.47 ± 0.48	89.78 ± 0.13
5	96.51 ± 0.17	90.17 ± 0.12	94.44 ± 0.26	90.23 ± 0.10	96.49 ± 0.18	90.16 ± 0.11	90.22 ± 5.43	89.76 ± 0.53	94.63 ± 2.80	89.98 ± 0.24	84.78 ± 1.64	89.26 ± 0.22
CIFAR10 (30 communication rounds)												
1	84.48 ± 1.45	81.54 ± 0.12	79.78 ± 2.15	81.44 ± 0.31	78.04 ± 1.92	81.19 ± 0.21	81.12 ± 5.48	81.16 ± 0.67	84.32 ± 2.72	80.77 ± 1.28	77.04 ± 2.83	81.25 ± 0.21
2	83.64 ± 2.30	81.07 ± 0.44	72.76 ± 2.89	81.02 ± 0.17	64.92 ± 5.54	80.06 ± 0.69	83.70 ± 1.44	81.13 ± 0.46	83.90 ± 1.90	80.20 ± 2.11	69.16 ± 3.42	80.64 ± 0.24
3	84.38 ± 1.14	81.17 ± 0.49	64.34 ± 1.14	80.22 ± 0.38	45.08 ± 12.5	77.93 ± 1.24	77.36 ± 7.79	80.36 ± 0.62	79.70 ± 2.70	78.51 ± 2.12	58.82 ± 2.69	79.53 ± 0.26
4	83.02 ± 1.98	80.96 ± 0.24	56.88 ± 3.96	79.38 ± 0.37	37.14 ± 15.2	77.08 ± 1.51	74.84 ± 6.88	80.14 ± 0.38	82.10 ± 1.78	78.14 ± 1.66	48.54 ± 2.93	78.59 ± 0.44
5	82.80 ± 1.95	80.56 ± 0.40	45.56 ± 2.97	78.34 ± 0.40	38.06 ± 23.6	77.06 ± 2.49	57.26 ± 12.1	78.61 ± 1.12	80.20 ± 2.81	75.62 ± 0.60	38.04 ± 3.23	77.44 ± 0.37
ISIC2019 (40 communication rounds)												
1	52.29 ± 1.69	61.90 ± 0.96	55.06 ± 1.64	61.71 ± 0.67	53.95 ± 5.09	62.44 ± 0.75	58.54 ± 1.97	61.74 ± 1.19	59.59 ± 3.05	62.01 ± 0.58	43.06 ± 5.24	61.98 ± 0.75
2	51.35 ± 2.04	61.22 ± 0.29	50.25 ± 5.38	61.46 ± 0.20	52.24 ± 6.33	60.89 ± 0.28	62.63 ± 2.31	59.94 ± 0.58	61.30 ± 1.05	57.13 ± 2.48	37.53 ± 7.79	62.02 ± 0.83
3	50.47 ± 2.28	59.49 ± 1.36	45.55 ± 6.65	60.24 ± 0.27	40.96 ± 7.32	60.49 ± 0.65	60.36 ± 1.17	59.97 ± 0.63	61.03 ± 5.45	51.85 ± 4.99	26.92 ± 8.37	59.88 ± 1.26
4	53.40 ± 3.69	59.20 ± 1.82	41.40 ± 5.51	60.18 ± 0.97	35.43 ± 0.96	56.95 ± 0.82	53.73 ± 7.31	59.87 ± 1.39	59.20 ± 2.74	53.42 ± 6.19	20.34 ± 6.64	58.86 ± 2.08
5	51.02 ± 5.48	58.60 ± 2.07	32.23 ± 6.16	58.60 ± 0.63	24.10 ± 8.72	56.27 ± 0.72	44.78 ± 5.69	58.47 ± 0.36	61.64 ± 0.77	49.08 ± 0.29	11.11 ± 4.75	57.76 ± 0.80

is a serious disease. Ensuring its correct classification is of paramount importance, justifying the prioritization of source label recall.

APPENDIX C
TARGETED ATTACKS ON THE CIFAR10 DATASET WITH HETEROGENOUS DATA DISTRIBUTION

In this section, we demonstrate the performance of FedGT for $n = 15$ clients under heterogeneous data distributions. To model heterogeneity, we distribute each class in the dataset among the clients following a Dirichlet distribution with parameter $\alpha \in \{0.4, 0.6, 0.8, 1.0\}$. The Dirichlet distribution is widely used to model heterogeneous client data, with smaller values of α corresponding to greater heterogeneity [54].

We conduct experiments on the CIFAR10 dataset for a targeted attack strategy, following the approach described in Section VI-C. For FedGT-Δ and FedGT-$\hat{n}_{\mathsf{m}}$, we use the same testing strategy as outlined in Section VI-C and compare their performance against RFA [26], Multi-Krum [12], no defense, and oracle. Note that Multi-Krum requires prior knowledge of n_{m}, making it an impractical option in real-world scenarios. The experiments are performed for $n_{\mathsf{m}} \in \{1, 3, 5\}$, and the results are averaged over 5 independent runs. In Fig. 7, we plot the attack accuracy as a function of α (first row) and the top-1 accuracy as a function of α (second row), measured after 30 communication rounds. Although the adversary's goal is to maximize the attack accuracy, we provide both the attack accuracy and top-1 accuracy to ensure that the defense does compromise the utility of the global model.

From Fig. 7, we observe that for $n_{\mathsf{m}} = 1$ (first column), both FedGT-$\hat{n}_{\mathsf{m}}$ and FedGT-Δ achieve lower attack accuracies compared to RFA and Multi-Krum, with FedGT-Δ performing very close to oracle. However, in terms of top-1 accuracy, FedGT-Δ performs worse than the other strategies, making FedGT-$\hat{n}_{\mathsf{m}}$ the best-performing defense overall for $n_{\mathsf{m}} = 1$. For $n_{\mathsf{m}} = 3$ and 5, FedGT-Δ achieves the lowest attack accuracies (excluding the oracle), and the impact of the defense on the top-1 accuracy is negligible, making FedGT-Δ the best-performing defense across all values of $\alpha = \{0.4, 0.6, 0.8, 1.0\}$.

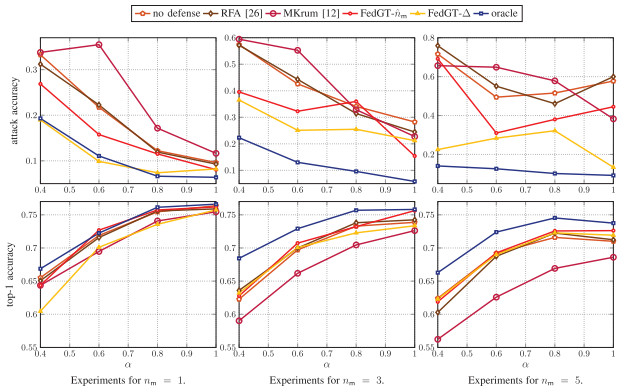

Fig. 7. Experimental results over CIFAR10 dataset for a targeted attack (see Section VI-C). The clients data distribution is non-identically and independently distributed that follows a Dirichlet distribution with parameter $\alpha \in \{0.4, 0.6, 0.8, 1.0\}$. We show the impact of the attack on the attack (first row) and top-1 accuracy (last row) for both versions of FedGT, RFA [26], Multi-Krum [12], no defense and oracle. We plot the results for $n_{\mathsf{m}} \in \{1, 3, 5\}$ measured after 30 communication rounds.

While both versions of FedGT perform well in heterogeneous settings, there remains room for improvement, as their performance still falls short of the oracle. This gap is particularly evident for $n_{\mathsf{m}} = 3$ and $n_{\mathsf{m}} = 5$, whereas for the homogeneous settings, the gap is smaller (see Table III). We believe that this discrepancy arises from our testing strategy, which assumes that benign clients exhibit similar behavior—a premise that does not hold in heterogeneous settings. Therefore, a new testing strategy that accounts for non-iid scenarios is necessary to further enhance the performance of FedGT in such scenarios.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stats. (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[2] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.

[3] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 2512–2520.

[4] D. I. Dimitrov, M. Balunović, N. Konstantinov, and M. Vechev, "Data leakage in federated averaging," *Trans. Mach. Learn. Res.*, vol. 2022, Jan. 2022.

[5] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[6] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (Poly)Logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1253–1269.

[7] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021.

[8] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 8635–8645.

[9] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2020, pp. 480–501.

[10] H. Wang et al., "Attack of the tails: Yes, you really can backdoor federated learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 16070–16084.

[11] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "The limitations of federated learning in Sybil settings," in *Proc. Int. Symp. Res. Attacks, Intrusions Defenses (RAID)*, Jan. 2020, pp. 301–316.

[12] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 119–129.

[13] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5636–5645.

[14] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 233–239.

[15] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," 2020, *arXiv:2002.00211*.

[16] R. A. Mallah, D. López, G. Badu-Marfo, and B. Farooq, "Untargeted poisoning attack detection in federated learning via behavior AttestationAl," *IEEE Access*, vol. 11, pp. 125064–125079, 2023.

[17] T. D. Nguyen et al., "FLAME: Taming backdoors in federated learning," in *Proc. USENIX Secur.*, 2022, pp. 1415–1432.

[18] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 2545–2555.

[19] X. Gong, Y. Chen, Q. Wang, and W. Kong, "Backdoor attacks and defenses in federated learning: State-of-the-art, taxonomy, and future directions," *IEEE Wireless Commun.*, vol. 30, no. 2, pp. 114–121, Apr. 2023.

[20] R. Dorfman, "The detection of defective members of large populations," *Ann. Math. Statist.*, vol. 14, no. 4, pp. 436–440, Dec. 1943.

[21] J. O. D. Terrail et al., "FLamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 5315–5334.

[22] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 1354–1371.

[23] (2023). *Project Aurora: The Power of Data, Technology and Collaboration to Combat Money Laundering Across Institutions and Borders*. Bank Int. Settlements. Accessed: Nov. 29, 2024. [Online]. Available: https://www.bis.org/publ/othp66.pdf

[24] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Informat.*, vol. 112, pp. 59–67, Apr. 2018.

[25] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.

[26] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022.

[27] Z. Zhang, J. Li, S. Yu, and C. Makaya, "SAFELearning: Enable backdoor detectability in federated learning with secure aggregation," 2021, *arXiv:2102.02402*.

[28] S. Li, E. C.-H. Ngai, and T. Voigt, "An experimental study of Byzantine-robust aggregation schemes in federated learning," *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 975–988, Dec. 2024.

[29] M. Aldridge, O. Johnson, and J. Scarlett, "Group testing: An information theory perspective," *Found. Trends Commun. Inf. Theory*, vol. 15, nos.3–4, pp. 196–392, 2019.

[30] A. R. Elkordy, J. Zhang, Y. H. Ezzeldin, K. Psounis, and S. Avestimehr, "How much privacy does federated learning with secure aggregation guarantee?," in *Proc. Privacy Enhancing Technol. (PoPETS)*, 2022, pp. 510–526.

[31] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-correcting Codes*, vol. 16.Amsterdam, The Netherlands: Elsevier, 1977.

[32] M. Grassl. (2007). *Bounds on the Minimum Distance of Linear Codes and Quantum Codes*. [Online]. Available: https://www.codetables.de

[33] M. Helmling et al. (2019). *Database of Channel Codes and ML Simulation Results*. [Online]. Available: http://www.uni-kl.de/channel-codes

[34] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[35] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[36] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *J. Cybern.*, vol. 4, no. 1, pp. 95–104, Jan. 1974.

[37] T. Jahani-Nezhad, M. A. Maddah-Ali, S. Li, and G. Caire, "SwiftAgg: Communication-efficient and dropout-resistant secure aggregation for federated learning with worst-case security guarantees," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 103–108.

[38] J. So et al., "LightSecAgg: A lightweight and versatile design for secure aggregation in federated learning," in *Proc. Mach. Learn. Syst. (MLSys)*, 2022, pp. 694–720.

[39] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.

[40] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[41] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 1, pp. 76–80, Jan. 1978.

[42] G. Liva, E. Paolini, and M. Chiani, "Optimum detection of defective elements in non-adaptive group testing," in *Proc. 55th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2021, pp. 1–6.

[43] W. Heyndrickx et al., "MELLODDY: Cross-pharma federated learning at unprecedented scale unlocks benefits in QSAR without compromising proprietary information," *J. Chem. Inf. Model.*, vol. 64, no. 7, pp. 2331–2344, Apr. 2024.

[44] Y. LeCun and C. Cortes. (2010). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[46] N. C. F. Codella et al., "Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC)," in *Proc. IEEE 15th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2018, pp. 168–172.

[47] A. Krizhevsky, V. Nair, and G. Hinton. (2009). *CIFAR-10 (canadian Institute for Advanced Research)*. [Online]. Available: http://www.cs.toronto.edu/

[48] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6105–6114.

[49] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and M. Yang, "Justinian's GAAvernor: Robust distributed learning with gradient aggregation agent," in *Proc. USENIX Secur. Symp.*, 2020, pp. 1641–1658.

[50] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Jan. 2021.

[51] J. Park, D.-J. Han, M. Choi, and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021, pp. 840–851.

[52] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2999–3007.

[53] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, no. 1, pp. 68–79, 1960.

[54] T.-M. Harry Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019, *arXiv:1909.06335*.