Network-Level Adversaries in Federated Learning

Giorgio Severi[*§], Matthew Jagielski[†§¶], Gökberk Yar[*], Yuxuan Wang[‡¶], Alina Oprea[*], Cristina Nita-Rotaru[*]

* Northeastern University, Khoury College of Computer Sciences

† Google Research

‡ LinkedIn Corporation

Abstract—Federated learning is a popular strategy for training models on distributed, sensitive data, while preserving data privacy. Prior work identified a range of security threats on federated learning protocols that poison the data or the model. However, federated learning is a networked system where the communication between clients and server plays a critical role for the learning task performance. We highlight how communication introduces another vulnerability surface in federated learning and study the impact of network-level adversaries on training federated learning models. We show that attackers dropping the network traffic from carefully selected clients can significantly decrease model accuracy on a target population. Moreover, we show that a coordinated poisoning campaign from a few clients can amplify the dropping attacks. Finally, we develop a server-side defense which mitigates the impact of our attacks by identifying and up-sampling clients likely to positively contribute towards target accuracy. We comprehensively evaluate our attacks and defenses on three datasets, assuming encrypted communication channels and attackers with partial visibility of the network.

I. INTRODUCTION

Federated Learning (FL) or collaborative learning, introduced by McMahan et al. [1], has become a popular method for training machine learning (ML) models in a distributed fashion. In FL, a set of clients perform local training on their private datasets and contribute their model parameters to a centralized server. The server aggregates the local model parameters into the global model, which is then disseminated back to the clients, and the process continues iteratively until convergence. Training ML models using FL allows the server to take advantage of a large amount of training data, and each client to retain the privacy of their data. FL can be deployed in either a cross-silo setting with a small number of participants available in all rounds of the protocol, or in a cross-device setting with a large number of participants, which are sampled by the server and participate infrequently in the protocol.

Recent work studying the security of FL has highlighted the risk of attacks by compromised clients through data poisoning [2] and model poisoning [3], [4] under different objectives such as availability, targeted, and backdoor attacks. While availability objectives aim at compromising the accuracy of the model indiscriminately [5], [6], targeted and backdoor attacks only affect a specific subset of samples [4], [7], [8] and are harder to detect. Among the proposed defenses, availability attacks can be mitigated by gradient filtering [5], [6], [9], while targeted and backdoor attacks could be addressed by clipping

gradient norms at the server, during model updates, to limit the individual contributions to the global model [4], [7].

FL is a networked system where communication between server and clients plays a critical role as the global model is learned in rounds with contributions from multiple clients. Thus, communication represents another point of vulnerability that an attacker can exploit to influence the global model learned by the system. In this case, an attacker leverages network-level information and attack capabilities to prevent the ML algorithm from *receiving* the information needed to learn an accurate model. An attacker can exploit the specific communication channels for FL to perturb messages sent between server and clients via jamming [10], BGP hijacking [11], compromising routers [12], hosts [13], or transport-level protocols [14]. Network-level attacks have been shown to impact other systems such as Bitcoin [15], payment-channel networks [16], and connected cars [17]. These adversaries are especially relevant when political or economic incentives result in a government or company exploiting an AS for which sub-populations are geographically localized. For instance, this attack may be used to censor a word prediction model (especially targeting a country's minority language), or modify words associated with a competitor company or unfavorable political movements.

In this paper, we conduct the first study of network-level adversaries' impact on machine learning models trained with federated learning. We analyze the ways an attacker can exploit information sent over the network to maximally damage the learning algorithm under realistic conditions such as encrypted channels and partial visibility of the network. Specifically, we show that attackers who can carefully orchestrate dropping of the network traffic of selected clients can significantly decrease model accuracy on a target population. Our key insight is that a few clients contribute significantly to the accuracy of the model for target subpopulations, and we create an algorithm that allows the attacker to identify such clients, and, by interfering with only those clients, significantly outperform randomized packet dropping. Moreover, we show that model poisoning attacks from a small set of clients can further amplify the impact of the targeted dropping attacks. As an example, on a text classification problem, selectively dropping updates from only 5 clients from the target population can lead to a decrease in accuracy from 87% to 17%, on a particular target class. Furthermore, by adding a poisoning attack of equal size, the model accuracy decreases to 0% on the same target class. Finally, we consider a resource limited adversary, who can

---

observe only a small subset of the participating clients, and show that our attacks are still effective, for instance inflicting a 43% relative accuracy drop when observing only a third of the clients in a computer vision task.

Federated learning system owners do not usually control the underlying network and might not be able to implement network-level mitigations to make communication more resilient. Complementary to such network-level defenses, we propose a server-level defense that is agnostic to how the dropping attack is performed. Our defense modifies client sampling in each round of FL training to increase the likelihood of selecting clients who sent useful updates in previous rounds of the protocol, while decreasing the probability of selecting less relevant clients. Interestingly, defensive client selection leverages the same procedure for client identification employed by the network-level adversary. The defense is extremely effective against a targeted dropping attack. For instance, in the same text classification task mentioned above, while an unmitigated attack would completely disrupt the model accuracy on the target task (from 87% to 17%), the defense achieves an accuracy of 96%, which exceeds that of the original model before an attack is mounted. Moreover, our defense can be combined with a standard poisoning defense based on update clipping to withstand both targeted dropping and model poisoning attacks. On the same task, the combined dropping and poisoning attack brings the target accuracy to 0, and the combination of our defense with clipping results in 94% accuracy on the target population. For encrypted communication the improvements compared to the original accuracy can be as high as 34%.

To summarize, the contributions of the paper are: (i) the first study of network-level adversaries in FL, specifically packet dropping attacks targeting a subpopulation and amplification by using model poisoning attacks; (ii) an algorithm for identification of highly contributing clients, allowing the attacker to be more effective than just randomly dropping traffic between clients and server; (iii) a defense based on up-sampling highly contributing clients to the learning task which can be combined with clipping to mitigate both targeted dropping and poisoning attacks; (iv) a comprehensive evaluation across multiple model architectures, datasets, and data modalities (image and text). For reproducibility, all code is publicly released[1].

II. BACKGROUND AND THREAT MODEL

A. Federated Learning

FL considers a set of n clients, each with a local dataset D_i, and a server S. Clients train locally on their own datasets and the server requests model updates, rather than data, building an aggregated global model iteratively over time [18]. We consider here the Federated Averaging training algorithm designed for cross-device settings [1]. In each round $1 \leq t \leq T$, the server randomly selects a subset of $m \leq n$ clients to participate in training, and sends them the current global model f_{t-1}. Each selected client i trains locally using dataset D_i, for a fixed

[1]https://github.com/ClonedOne/Network-Level-Adversaries-in-Federated-Learning

number of T_L epochs. The server updates the global model f_t by mean aggregation of local updates U_i: $f_t = f_{t-1} + \eta \frac{\sum_{i=1}^{m} U_i}{m}$. Data privacy can be further enhanced in FL by secure aggregation performed via Multi-Party Computation (MPC) [19].

Algorithm 1: Federated Averaging Protocol

Data: Clients $\mathcal{C} = \{D_i\}_{i=1}^n$, Federated Learning Server S, rounds T, clients per round m, aggregation learning rate η

Function FedLearn($S, \mathcal{C}$):
    // Function run by server
    $f_0 =$ INITIALIZEMODEL()
    for $t \in [1, T]$ **do**
        // Get updates from m participants in round i
        $M_t =$ SELECTPARTICIPANTS($\mathcal{C}, m$)
        REQUESTUPDATE(f_{t-1}, M_t)
        $U_t =$ RECEIVEUPDATE(M_t)
        // Update and send out new model
        $f_i =$ UPDATEMODEL(f_{t-1}, U_t, η)
        BROADCASTMODEL($f_t, \mathcal{C}$)

B. Threat Model

Adversarial goal: An attacker can target the accuracy for all the classes of the learning task – availability attacks, or target a particular class – targeted attacks. Targeted attacks are much more difficult to detect as the attacker strives to make the model retain its test accuracy for non-targeted points to avoid trivial detection. We consider targeted attacks and define a population to be one of the classes in the learning task, but this notion could be extended to sub-classes as well. We do not consider poisoning availability attacks which are detectable and can be addressed with existing defenses [5], [6], [9].

Attack strategies: The attacker can conduct poisoning attacks, or network-level attack. Poisoning attacks can target the training data – an adversary injects maliciously crafted data points to poison the local model [2], or the model - where the adversary compromises a set of clients and sends malicious updates to the protocol with the goal of achieving a certain objective in changing model's prediction [4], [7], [8]. Both these attacks are conducted by manipulating directly the inputs to the machine learning algorithm. We consider an attacker with model poisoning (POISON_MODEL) capability, using it to amplify network-level attacks.

In network-level attacks, based on their network-level capabilities, the adversary can observe communication sent over the network and prevent the machine learning algorithm from *receiving* the information needed to learn a good model. The basic action for a network-level attack is dropping traffic. We consider a smart attacker that selectively drops traffic to maximize the strength of the attack while minimizing detection. The attacker has several decisions to make: (1) what clients to select to drop their contributions, (2) when to start the attack given that federated learning is an iterative protocol, and (3) how many packets (i.e., local models) to drop. We focus on a targeted dropping attack in which the attacker selects a set of clients contributing highly to the target class for dropping their

contributions. We compare that with an attack which drops the traffic of a subset of randomly chosen clients.

Adversarial network-level knowledge: We consider encrypted communication and partial network visibility scenarios for the communication in the FL protocol. For comparison we also consider unencrypted communication.

1) **COMM_PLAIN**. All communication between clients and server is unencrypted. This is a baseline scenario where a network-level adversary obtains maximum information. We do not expect this to happen in practice and we study this model to understand the maximum possible damage.
2) **COMM_ENC**. All communication between clients and server is encrypted. This is a realistic scenario, where the network-level adversary could infer the set of clients participating in each round, but not the exact model updates they send. This is the typical deployment for FL, using IPSec, HTTPS, or application-level encryption.
3) **COMM_ENC_LIMITED**. A special case of COMM_ENC where communication is encrypted, and the adversary is limited to only observe a fixed subset of the clients participating in the protocol. This is the most constrained attacker we consider.

Adversarial global model knowledge: Since cross-device FL is an open system, the adversary can always participate in the FL protocol as one of the clients. We assume that the adversary obtains the global model updates f_t at each round t.

To summarize, we consider an adversary that has either the COMM_PLAIN, COMM_ENC or COMM_ENC_LIMITED network capability, has knowledge of the global model at each round, and targets a particular victim population of interest. We also consider an adversary that additionally has POISON_MODEL poisoning capabilities.

C. Feasibility of Network-level Attacks for Federated Learning

FL protocols such as Federated Averaging [1] usually abstract the communication protocol between the server and clients. In practice, clients will communicate with the server either directly through a TCP connection, or through a multi-hop overlay network of hop-by-hop TCP or customized transport services. Finally, the last hop connecting the client to the Internet is often in the form of wireless communication. Network infrastructure and routing protocols facilitate all this communication through physical or logical connectivity. By exploiting the underlying routing protocols and network topology an attacker can position themselves to observe and interfere with data of interest; they can then further impact the accuracy of the global model and create an effect similar to that created through a data or model poisoning attack.

Packet dropping attacks can be achieved in multiple ways, and have been studied for network-level adversaries who perform physical-layer attacks in wireless networks [10], router compromise [12], or transport-level attacks [14]. Each of these methods will require different attacker capabilities. For example for physical layers attacks for wireless clients, the attacker needs to be in their proximity - this is both a powerful

attack since jamming is usually difficult to defend against, and limiting since it will be difficult for the attacker to attack geographically distributed clients without significant resources. In the case of routing, while a router might be more difficult to compromise, in practice the impact can be bigger as it might have control over the traffic of multiple clients. Last, but not least, if customized overlays are used for communication, compromising nodes in the overlay is slightly easier as they are typically hosts, and the number of clients that can be impacted depends on the scalability of the service.

III. NETWORK-LEVEL ATTACKS ON FEDERATED LEARNING

In this section we describe in detail our targeted network attacks against a population in federated learning.

A. Identification of Highest-Contributing Clients

The key insight of our attack is that only a small number of clients contribute to the accuracy of a target population of interest. Thus, the most effective and difficult to detect strategy is to drop the traffic just for those highly contributing clients. However, the attacker will first need to identify such clients. We design a Client Identification algorithm aimed at determining the set of clients whose model updates lead to the largest improvements in target population accuracy. Our method, described in Algorithm 2, supports both plain and encrypted network communication, and is very effective even if the attacker has limited observability of the network. We demonstrate in Section V that if the adversary has high success at identifying the clients contributing to the task of interest, then the targeted dropping attack impact is much higher compared to random dropping.

Algorithm 2: Loss Difference Client Identification

Data: Target Population Dataset D^*, loss function ℓ, rounds T_N, count of clients to drop k_N, visibility parameter v.

Function ClientIdentification(D^*, ℓ, T_N, k_N):
    $f_0 = $ GETGLOBALMODEL(0)
    $\Delta = []$
    for $t \in [1, T_N]$ **do**
        $M_t = $ GETPARTICIPANTS(v)
        $f_t = $ GETGLOBALMODEL(t)
        if COMM_ENC **then**
            // Get loss differences for this round
            $L_t = \ell(f_{t-1}, D^*) - \ell(f_t, D^*)$
            for $j \in M_t$ **do**
                // Associate L_t with update members
                $\Delta[j] = $ CONCATENATE$(\Delta[j], [L_t])$

        if COMM_PLAIN **then**
            for $j \in M_t$ **do**
                $f_t^j = $ GETLOCALMODEL(j)
                // Get participant's loss difference
                $L_t^j = \ell(f_{t-1}, D^*) - \ell(f_t^j, D^*)$
                // Associate L_t^j with this participant
                $\Delta[j] = $ CONCATENATE$(\Delta[j], [L_t^j])$

    // Compute the average loss change for each client
    for $j \in \Delta$ **do**
        $\Delta[j] = \frac{1}{|\Delta[j]|} \sum \Delta[j]$
    $Z = $ GETLARGESTVALUES(Δ, k_N)
    return Z

The main intuition is that the adversary computes the loss of the model before and after the updates on the target class for

a number of rounds T_N. Rounds in which the loss decreases correspond to an increase in accuracy on the target class. The adversary tracks all the clients participating in those rounds, and computes a loss difference metric per client. In the plain communication case, the adversary will determine exactly which clients decrease the loss, while in the encrypted communication case the adversary only observes the aggregated updates and considers all clients participating in the round to be collectively responsible for the loss improvement.

In more detail, to measure each client's contribution to the target population, the adversary computes the loss difference between successive model updates on the target population (using a representative dataset D^* from the population):

$$L_t^j = \ell(f_{t-1}, D^*) - \ell(f_t^j, D^*)$$

Note that the second term $\ell(f_t^j, D^*)$ is the loss of the local update of client j in round t for COMM_PLAIN, but becomes the loss of the global model $\ell(f_t, D^*)$ when the adversary only has access to aggregated updates in COMM_ENC or COMM_ENC_LIMITED. The value L_t^j measures the decrease in the target class loss before and after a given round t.

To more reliably estimate client contributions, especially with aggregation of the model updates, the values of L_t^j are accumulated over time. The adversary maintains a list for each client j of the values L_t^j for rounds t in which it participated. This allows the adversary to update the list of clients most relevant to the target class accuracy at every round. Empirically, computing the mean of L_t^j for each client j across all rounds it participates in works well at identifying the clients contributing most to the target class.

B. Dropping Attack

The targeted dropping attack starts with the adversary performing the Client Identification procedure, by running Algorithm 2 during the first T_N rounds of the protocol, in which training happens as usual. The adversary identifies in expectation k_N clients contributing updates for the target class. After Client Identification is performed, the network adversary drops contributions from the k_N clients in every round in which they participate after round T_N. As the adversary can expect an increase in identification accuracy by monitoring more rounds of the protocol, they can repeat the Client Identification procedure in each subsequent round, and, if necessary, update the list of selected clients. We assume that the adversary drops the identified clients' updates, and the server simply updates the model using all received updates. If the Client Identification protocol is not completely accurate, the attacker might drop traffic from non-target clients but we found this has a minimal impact on the trained FL model.

C. Attack Analysis

Algorithm 2 uses several parameters: the number of clients to identify k_N and the number of rounds to wait before identification is successful T_N, which we analyze here.

1) How many clients to drop?: In the FL protocol, m out of n clients are selected at random in each round. Setting the number of dropping clients k_N is mainly a tradeoff between maximally damaging accuracy on the target data and remaining stealthy by not significantly compromising the overall model performance. If k_N is too large, significant benign updates may be removed, preventing the model from achieving good accuracy, and potentially allowing the server to identify that an active adversary, rather than standard packet loss, is to blame for the dropped updates. If k_N is too small, however, not enough clients will be dropped to have a significant impact on the model. We consider a range of values $k_N \leq k$, where k is the number of clients holding examples from the targeted class, and show the attack effectiveness for each.

2) How many rounds are needed to identify the clients?: Here, we discuss how to set the number of rounds T_N for client identification for both plain and encrypted communication. When setting T_N, if the value is set too small, then the adversary will not have enough observations to reliably identify the contributing clients. However, if set too large, the adversary will allow benign training too much time, resulting in high accuracy on the target population, making it more difficult to mount the attack later. Suppose the adversary wants to wait until all n clients have participated in the protocol at least once. This is a well-studied problem, known as the *coupon collector's* [20] (our setting additionally considers batched arrivals). In this variation of the problem, the adversary must observe an expected number of $cn \log(n)/m$ batches before observing each client, for a small constant c. With values of $n = 100$, $m = 10$, roughly 46 rounds are necessary [21]. Having established a connection to the coupon collector's problem, we will extend it to model the adversary in each setting:

Plain communication COMM_PLAIN. The baseline case is where the adversary receives *every* client's updates. We carry out a modification of the coupon collector analysis to identify the setting of T_N where a batch of m out of n distinct clients participate in each round, and the goal is to identify a set of k_N of k target clients. The number of batches to wait for the i-th client from the k target clients is $t_i = \frac{n}{m(k-i+1)}$ in expectation, as the probability that any target client which has been unobserved appears is $\frac{m(k-i+1)}{n}$. By summing this expected batch count over i from 1 to k_N (using the linearity of expectation), we find that the expected number of batches to wait for the first k_N clients is $\frac{n}{m}(H_k - H_{k-k_N})$, where H_i is the i-th harmonic number (and using $H_0 = 0$ if $k_N = k$). As k increases, this value tends towards $\approx \frac{n}{m}(\ln(k) - \ln(k - k_N))$ rounds (when $k = k_N$, we replace $\ln(0)$ with 0). To use a setting that is common in our experiments, where $n = 60$, $m = 10$, $k = 15$, and $k_N = 15$, we expect to wait roughly $\frac{n}{m} \ln(k) = 6 \ln(15)$ rounds, or roughly 16 rounds. Note that waiting for fewer clients k_N requires fewer rounds.

Encrypted communication COMM_ENC and COMM_ENC_LIMITED. In the realistic case of encrypted communication, the adversary only has access to mean updates and cannot localize an improved target accuracy to a particular

client, as is possible in COMM_PLAIN. However, clients who repeatedly participate in rounds where the target accuracy improves, can be considered as more likely targets. Moreover, clients who participate in any round where the target accuracy does not improve can be identified as not targets, and we analyze only rounds without target clients. An α precision level at identification can be reached by removing $n - k/\alpha$ clients which are known non-target clients. To analyze how many rounds it takes to collect $n - k/\alpha$ non-target clients, we compute the probability that a batch will contain non-target clients, and then compute how many non-target batches are required to collect $n - k/\alpha$ clients. To compute the probability that a batch contains non-target clients, we notice that there are $\binom{n-k}{m}$ possible batches which have non-target clients, and there are a total of $\binom{n}{m}$ batches which are selected uniformly at random. Then the probability a batch has non-target clients is $\binom{n-k}{m} / \binom{n}{m} \approx \left(1 - \frac{k}{n}\right)^m$.

To compute the number of non-target batches required to accumulate $n - k/\alpha$ non-target clients, we can use comparable coupon collector analysis as before, making the observation that each non-target batch is guaranteed to have m non-target clients. On average, an upper bound of $\frac{n}{b}(H_{n-k} - H_{k/\alpha - k})$ batches is sufficient. As an example, in a setting we use in our experiments, $n = 60$, $k = 15$, and $m = 10$, the probability that a batch contains non-target clients is roughly 5.6%. To reach a precision of $\alpha = 0.3$, we obtained a total of 26 batches on average. In practice, it is likely that precision can be even higher than this value due to overestimation from our analysis.

We note that our analysis is similar to analysis for the *group testing problem* [22], introduced by [23], used for pool testing. However, a key difference is that our algorithm must be capable of identifying members from noisy aggregate information, rather than the clean signal which is typically provided during group testing. It is possible that more sophisticated group testing algorithms can be used to improve Algorithm 2 further by overcoming this constraint.

D. Amplifying Dropping Attack with Model Poisoning

In order to amplify the effectiveness of the targeted dropping attack, the adversary may also collude with, or inject, malicious clients, whose presence in training is designed to further compromise the performance on the target distribution. Following Bagdasaryan et al. [4] and Sun et al. [7], we use a targeted model poisoning attack known as model replacement. Writing θ_t for the parameter of the global model f_t, in this attack, the adversary seeks to replace θ_t with a selected target θ_t^* (as is done in [4], [7], we use the θ_t^* resulting from a data poisoning attack on a compromised client's local training procedure). The poisoned clients' local training sets are sampled identically to clients with access to the target class, with the difference of changing the labels of target class samples to another, fixed class. The update that the adversary sends is $\theta_t^* - \theta_{t-1}$. The server aggregation then weights this update with an η/m factor. In a model replacement attack, a boosting factor β is applied to the update, so the update which is sent is $\beta(\theta_t^* - \theta_{t-1})$, increasing the contribution to overcome the η/m factor decrease.

IV. DEFENSES AGAINST NETWORK-LEVEL ADVERSARIES

Several defenses against network-level attacks are known. For instance, defenders could monitor and detect faulty paths in the network [24], create resilient overlays [25], [26], or secure the routing protocols [27]. These defenses might increase robustness, but are generally not effective against stealthy attacks, such as targeted update dropping, or edge-level attacks, such as model poisoning. Often, the FL owner might not control the network, so we design FL-specific server defenses that complement existing network-level defenses against the attacks introduced in Section III.

Model Poisoning Countermeasures. A popular form of defense against data and model poisoning attacks in FL is to replace the Federated Averaging protocol with a secure aggregation scheme [28]–[31]. It was shown, however, that an adversary can evade this defense (e.g., [4]–[6]), and finding a secure aggregation method remains an open problem. Sun et al. [7] observed that model poisoning attacks with larger gradient norms are more effective, and therefore a natural defense is to reduce the norms. With this method, an update g sent from a client is clipped to a maximum norm C and becomes $\min\left(1, \frac{C}{\|\Delta\|}\right) g$. Clipping works particularly well against model poisoning attacks in which the local client update is boosted by the attacker [4].

Algorithm 3: Server Defensive UpSampling Strategy

Data: Target Dataset D_S^*, rounds T_S, loss function ℓ, client count n, count of clients to upsample k_S, up-sample factor λ
Function UpSampling(D_S^*, ℓ, k_S, T_S, n):
  $Z = \text{ClientIdentification}(D_S^*, \ell, T_S, k_S)$
  // Reduce sampling probability for most clients
  $p = [\frac{n - k_S \lambda}{n^2 - k_S n} \text{ for } c \in [n]]$
  for $i \in Z$ **do**
    // Increase sampling probability for highly contributors
    $p[i] = \lambda/n$
  return p

UpSampling Defense Strategy. While Clipping reduces the impact of model poisoning attack, we still need to defend against targeted update dropping. When a dropping attack is performed, the number of clients contributing legitimate updates to the target class is reduced, leading to a larger impact on naturally under-represented classes (i.e., available in a small set of clients). Thus, it is essential for the server to first identify clients contributing to the target class, and second, leverage them to improve the accuracy of the target population. For the first component, we can use directly Algorithm 2 for Client Identification, using the knowledge available to the server. In standard FL implementations, servers receive individual model updates from the clients, while in privacy-preserving FL implementations based on MPC [19] servers only receive aggregated updates. For the second task, the server can run Algorithm 3 (UpSampling defense) to modify the client sampling strategy in the FL protocol. With this modification, the sampling probability of identified clients is increased by a factor of λ and the sampling probability of all others is decreased.

TABLE I
THE PARAMETERS USED IN OUR EXPERIMENTS.

Party	Param	Definition	Setting
Dataset	n	Total Clients	{100, 60}
Dataset	k	Clients with Target Class	{9, 12, 15}
Dataset	α_D	Dirichlet Distribution Param	1.0
Dataset	α_T	Target Class Dataset Fraction	{0.5, 0.6}
Client	T_L	Local Training Epochs	2
Client	η_L	Local Learning Rate	{ 0.1, 0.001 }
Server	η	Aggregation Learning Rate	0.25
Server	m	Clients per Round	10
Server	T	Total Training Rounds	Varied
Server	T_S	Rounds before up-sampling	Varied
Server	k_S	Up-sampled Client Count	Varied
Server	λ	Up-sampling factor	2
Server	C	Aggregation Clipping Norm	1
Adversary	T_N	Rounds before Dropping	Varied
Adversary	k_N	Number of Dropped Clients	Varied
Adversary	k_P	Number of Poisoning Clients	Varied
Adversary	β	Poisoning Boost Factor	10

Dataset	k	Number of Clients Dropped k_N			
		0	$k/3$	$2k/3$	k
Perfect Knowledge					
	9	0.47/0.66	0.21/0.50	0.01/0.23	0.0/0.0
EMNIST	12	0.58/0.78	0.36/0.61	0.06/0.31	0.0/0.0
	15	0.65/0.80	0.48/0.66	0.15/0.40	0.0/0.0
FashionMNIST	15	0.40/0.55	0.17/0.32	0.02/0.09	0.0/0.0
DBPedia	15	0.36/0.53	0.03/0.06	0.00/0.00	0.0/0.0
Random Drop					
	9	0.47/0.66	0.40/0.68	0.39/0.68	0.35/0.64
EMNIST	12	0.58/0.78	0.58/0.78	0.57/0.77	0.53/0.76
	15	0.65/0.80	0.66/0.82	0.64/0.81	0.65/0.82
FashionMNIST	15	0.40/0.55	0.39/0.53	0.38/0.53	0.35/0.50
DBPedia	15	0.36/0.53	0.39/0.54	0.31/0.47	0.34/0.45

The server will determine how many clients to identify, k_S, and how many rounds to monitor, T_S, using its own target dataset D_S^* to estimate contributions. As with the attacker, the server will repeat the Client Identification process at each successive round to refine its list of clients to up-sample. Interestingly, the UpSampling strategy can help even if there is no dropping attack, but there are simply too few clients from some target population on which the model accuracy is low.

V. EXPERIEMENTAL EVALUATION

Here, we evaluate our attacks, show how model poisoning amplifies targeted update dropping damage, and conclude by looking at the performance of the UpSampling defense.

A. Experiment Setup

We use three well known datasets (EMNIST, FashionMNIST, and DBPedia-14), which we adapt to the *non i.i.d.* setting by controlling the class distribution across clients. In all cases, the target population is represented by samples of an entire class selected from the dataset. Our datasets have balanced classes, and we use classes 0, 1, 9 for our evaluation. All reported results are averages of 4 trials, with different randomness seeds. The list of parameters used in our FL protocol is shown in Table I.

EMNIST [32] is a handwritten digits recognition dataset with 344K samples. We use approximately 100K images of numbers, partitioned equally, without overlap, among 100 clients. To enforce heterogeneity, we allocate samples from the target victim class to k fixed clients and vary k. For those k clients, we allocate $\alpha_T = 50\%$ of the local dataset to be target class points, while the remainder is sampled from the remaining classes according to a Dirichlet distribution with parameter $\alpha_D = 1.0$. For clients without the target class, we sample 100% of the local training set from a Dirichlet distribution with $\alpha_D = 1.0$, following [33]. We train a convolutional model (two 2D convolution layers and two linear layers, with learning rate $\eta_L = 0.1$) for 100 rounds, selecting 10 clients at each round, using mean aggregation with a learning rate $\eta = 0.25$.

FashionMNIST [34] is an image classification dataset, consisting of 70K gray-scale images of 10 types of clothing

articles. It is more complex than EMNIST, but smaller. Thus, we increase the number of training rounds to $T \in \{200, 300\}$, reduce the number of clients n to 60, and set α_T to 0.6. We set the local dataset size to 400. We use a convolutional model similar to the one used before, and fix all other parameters.

DBPedia-14 [35] is an NLP text classification dataset consisting of 560K articles from 14 ontological categories, such as "Company", "Animal", "Film". DBPedia-14 is a relatively complex dataset, so we use the same k, T, and α_T as in FashionMNIST, and a local dataset size of 1000. The model is trained starting from pre-trained GloVe embeddings [36], followed by two 1D convolution layers and two linear layers, and optimized with Adam, with $\eta_L = 0.001$.

B. Baselines: Perfect Knowledge and Random Dropping

To demonstrate the potential severity of a dropping attack, we evaluate the best possible case, where the adversary has perfect knowledge of a subset of k_N target clients from the beginning of the protocol, and drops every update originating from them throughout training. The results in Table II demonstrate the power of update dropping, and provide a baseline to compare our full attack pipeline against. We also evaluate the effect of an adversary that selects victim clients uniformly at random.

C. Client Identification Evaluation

Table III shows the average number of targeted clients correctly identified by Algorithm 2 for encrypted communication with plain as comparison. Client Identification works very well for plain communication, and maintains a reasonable performance even when the adversary only sees aggregated updates (COMM_ENC). For instance, on DBPedia an average of 11.75 out of 15 clients are identified at 70 rounds under COMM_ENC. Interestingly, for FashionMNIST and DBPedia, it is easier to identify target clients than for EMNIST (where an average of 7 out of 15 clients are identified at round 70). One hypothesis for this phenomenon is that classes are more distinct in complex datasets, leading the global model to forget the target class in rounds with no participating target clients, resulting in significant loss increases for those rounds.

We also use Table III to select parameters for the targeted dropping attack and validate our analysis from Section III-C.

TABLE III
TARGET CLIENT IDENTIFICATION. AVERAGE NUMBER OF CLIENTS CORRECTLY IDENTIFIED BY ALGORITHM 2 AT DIFFERENT ROUNDS UNDER
COMM_PLAIN AND COMM_ENC, $k_N = k$. ON FASHIONMNIST AND DBPEDIA ALL 15 TARGET CLIENTS ARE IDENTIFIED AT 50 AND 20 ROUNDS,
RESPECTIVELY, FOR COMM_PLAIN, WHILE THE MAXIMUM NUMBER OF CLIENTS IDENTIFIED UNDER COMM_ENC IS 11.75 AT 70 ROUNDS FOR DBPEDIA.

Dataset	Communication	k	Round Number T_N							
			5	10	15	20	30	40	50	70
EMNIST	COMM_PLAIN	9	1.75	5	7.50	7.75	8.50	8.75	8.75	8.50
		12	5	8.25	9.75	10.25	10.75	10.75	11.25	11.25
		15	4.25	9.50	11.50	12	14.25	14.25	14	14
	COMM_ENC	9	0.50	2.25	3.25	2.75	3.25	3.75	4	5.25
		12	2	2.75	3.25	3	3	4	5	5.25
		15	3	4	4	3.75	4.75	5.75	6.25	7
FashionMNIST	COMM_PLAIN	15	9	12	14	14.75	14.75	14.75	15	15
	COMM_ENC	15	5.5	6.50	8.0	8.50	9	10	10.50	11
DBPedia	COMM_PLAIN	15	8	13.25	13.75	15	15	15	15	15
	COMM_ENC	15	5.25	7	8	9	10	11.25	11.25	11.75

In the COMM_ENC scenario, we see that more rounds are necessary for successful identification, as expected from Section III-C. To identify between $k/3$ and $2k/3$ of the target clients, we need to wait between 30 and 50 rounds, and, in many cases, the identification accuracy tends to plateau in successive rounds. Thus, we select round 30 as the starting point for the dropping attack in our COMM_ENC experiments.

D. Targeted Dropping Evaluation

We measure our targeted dropping attack's performance in Table IV. Under COMM_PLAIN, it significantly compromise target population accuracy, closely approximating the perfect knowledge adversary for increasing values of k_N. We also observe that our attack in the COMM_PLAIN scenario vastly outperforms the strategy of randomly dropping the same number of clients, in all situations. Our attacks' performance decreases when the adversary's knowledge is limited in COMM_ENC, which is expected from the reduced Client Identification performance. We still observe a significant advantage in using our identification pipeline, over the random selection baseline. For instance, with $k_N = 5$, the target accuracy on DBPedia drops drastically from 53% to 6%. Moreover, these results highlight the trend mentioned in Section V-C: on more complex tasks, such as FashionMNIST and DBPedia, the high identification accuracy leads to noticeably larger attack performance, than in EMNIST. Given the targeted nature of our attack, in all considered scenarios, dropping the victim clients generally leads to very little degradation of the overall model performance – on average 3.88% for COMM_PLAIN and 1% for COMM_ENC.

Tables VI, VIII, X, show the accuracy of the global model on the full test set under the same settings as the other experiments we run.

E. Impact of Model Poisoning and Targeted Dropping

We compared the effects of the model poisoning strategy and targeted dropping on the EMNIST dataset for the cases of perfect knowledge, COMM_PLAIN, and COMM_ENC in Figures 1a, 1c, and 1e respectively. These heatmaps show that, for different levels of k_N (number of dropped clients)

TABLE IV
TARGETED DROPPING ATTACK, UNDER COMM_PLAIN AND COMM_ENC.
ACCURACY ON TARGET POPULATION AT ROUNDS $T/2$ AND T. $T = 100$
FOR EMNIST, $T = 200$ FOR FASHIONMNIST AND DBPEDIA.

COMM_PLAIN

Dataset	k	Number of Clients Dropped k_N			
		0	$k/3$	$2k/3$	k
EMNIST	9	0.47/0.66	0.25/0.49	0.09/0.18	0.00/0.00
	12	0.58/0.78	0.39/0.65	0.19/0.33	0.00/0.00
	15	0.65/0.80	0.50/0.74	0.26/0.50	0.02/0.02
FashionMNIST	15	0.40/0.55	0.24/0.23	0.02/0.03	0.00/0.00
DBPedia	15	0.36/0.53	0.10/0.01	0.00/0.00	0.00/0.00

COMM_ENC

Dataset	k	Number of Clients Dropped k_N			
		0	$k/3$	$2k/3$	k
EMNIST	9	0.47/0.66	0.35/0.58	0.32/0.52	0.32/0.52
	12	0.58/0.78	0.50/0.72	0.48/0.69	0.45/0.67
	15	0.65/0.80	0.63/0.78	0.60/0.76	0.56/0.71
FashionMNIST	15	0.40/0.55	0.34/0.38	0.14/0.13	0.03/0.01
DBPedia	15	0.36/0.53	0.19/0.06	0.06/0.00	0.00/0.00

and k_P (number of poisoned clients), the model replacement attack is more effective than targeted dropping, even when the adversary has perfect knowledge. The results, however, are significantly different when Clipping, a standard poisoning defense [7], [37], [38], is applied to local model updates. Figures 1b, 1d, and 1f show the same set of experiments repeated with a clipping norm of 1. These heatmaps highlight that, while clipping lowers the impact of model poisoning, the combination of targeted dropping and model poisoning still results in very noticeable targeted performance degradation. For instance, under COMM_PLAIN, the original model accuracy on class 0 is 0.8. This becomes 0.59 when dropping 7 clients out of 15, and 0.66 when poisoning 7 clients, while combining both attacks leads to 0.4 accuracy.

By repeating the experiments with $T = 50$, we can observe that it appears that targeted dropping is more effective than poisoning at low round counts, as shown in Fig. 2 When the model is in its early phase of learning, it needs to see some examples of the target class to begin learning it, and dropping significantly delays the arrival of these examples. Later in training, however, the difference between poisoning and dropping decreases.

(a) Perfect knowledge

(b) Perfect knowledge, clipping

(c) COMM_PLAIN

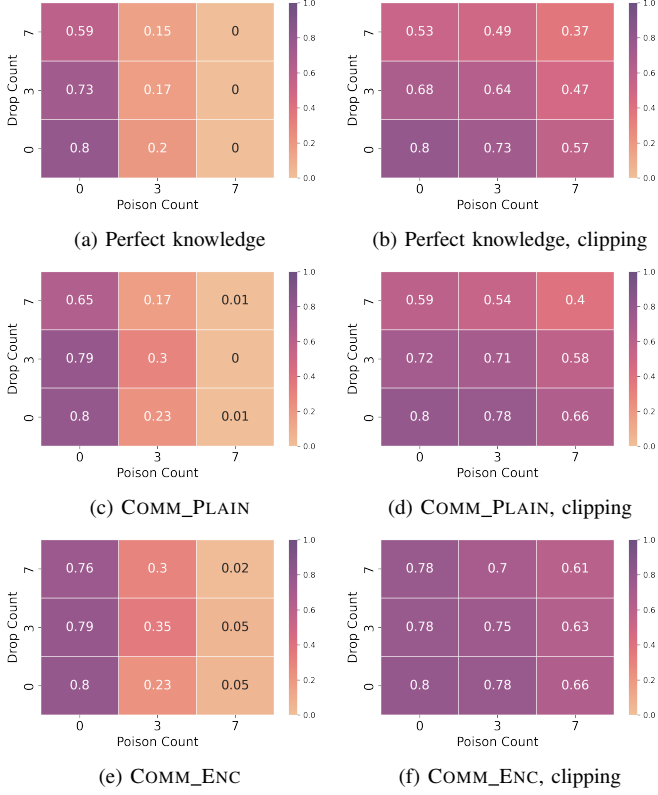(d) COMM_PLAIN, clipping

(e) COMM_ENC

(f) COMM_ENC, clipping

Fig. 1. Accuracy on target class 0 on EMNIST, for $k = 15$, $T = 100$, and varying number of dropped and poisoned clients under 3 scenarios: Perfect Knowledge, COMM_PLAIN, and COMM_ENC. Left results are without Clipping, and right results use a Clipping norm of 1.

F. Impact of Adversarial Visibility

We model a targeted, resource-limited, adversary, COMM_ENC_LIMITED, by restricting their ability to observe the clients participating to the protocol to a fixed-size subset of the clients, chosen before the attack starts, sampled from a Dirichlet distribution with concentration parameter α_V. α_V controls how likely it is for the sampled visible set to include clients containing data of the target population, with larger values of α_V leading to higher likelihood. We run the attack on FashionMNIST with a similar setup as in Table IV under the COMM_ENC conditions. The results reported in Figure 3a show that, as expected, higher visibility fractions, and larger α_V lead to smaller target accuracy values. The network adversary is still quite effective even when observing a small fraction of the clients, for instance achieving a $\approx 43.6\%$ relative target accuracy decrease when observing 20 out of 60 clients with $\alpha_V = 2$. Similarly, Figure 3b shows that adding poisoning always leads to better accuracy degradation.

G. Defense Evaluation

In Tables V, VII, we evaluate the defense strategies under COMM_PLAIN and COMM_ENC, using targeted dropping and poisoning attacks. We set the number of dropped (k_N) and poisoned clients (k_P) to $2k/3$ for EMNIST, and $k/3$ for FashionMNIST and DBPedia. These parameters result in strong

(a) Perfect knowledge

(b) Perfect knowledge, clipping

(c) COMM_PLAIN

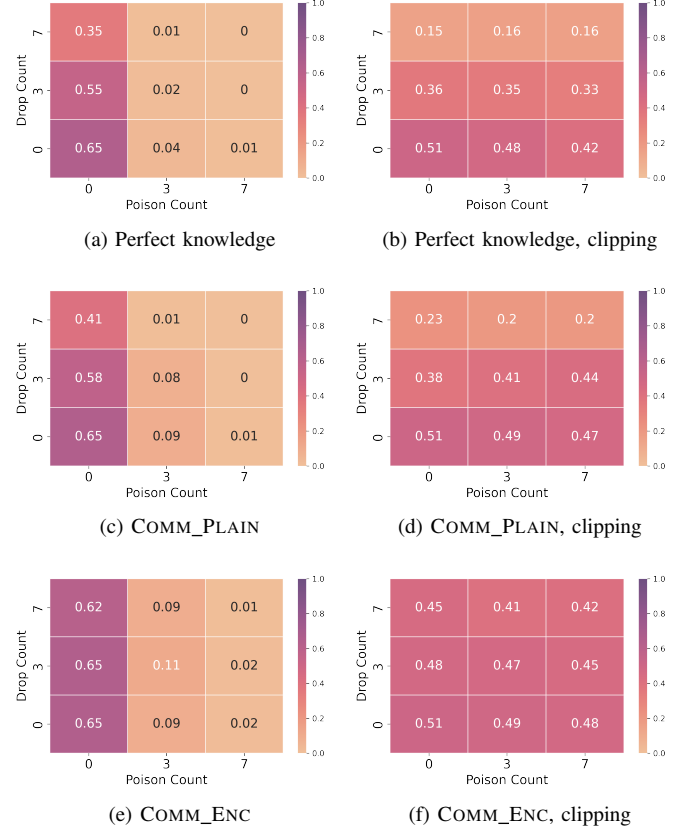(d) COMM_PLAIN, clipping

(e) COMM_ENC

(f) COMM_ENC, clipping

Fig. 2. Accuracy on target class 0 on EMNIST, for $k = 15$, $T = 50$, and varying number of dropped and poisoned clients under 3 scenarios: perfect knowledge, COMM_PLAIN, and COMM_ENC. Left results are without clipping, and right results use a clipping norm of 1. When no clipping is used, model poisoning attack is devastating at small number of poisoned clients. With clipping, model poisoning. Results are similar with those in Figure 1.

(a) Drop

(b) Drop + Poison

Fig. 3. COMM_ENC_LIMITED. Accuracy on target class 0 on FashionMNIST, for $k = 15$, $T = 200$, $k_N = 10$, $k_P = 10$, varying visibility and α_V.

attacks, as the target accuracy is below 4% under targeted dropping and poisoning, when no mitigation is used. UpSampling is successful in mitigating targeted dropping, and, when combined with Clipping, achieves high accuracy against the powerful combined attack. For instance, on EMNIST's class 1 with $k = 9$, the update dropping attack causes accuracy to decrease from 92% to 25% under COMM_PLAIN, and UpSampling manages to restore the target accuracy to 76%.

The UpSampling defense is very effective under the COMM_ENC setting, due to knowledge asymmetry, i.e. the attacker receives aggregated updates, while the defender observes

TABLE V
ACCURACY ON TARGET CLASS PRESENTED AT ROUNDS $T/2$ AND T, UNDER COMM_PLAIN SETTING. $T = 100$ FOR EMNIST, $T = 300$ FOR FASHIONMNIST AND DBPEDIA. WE CONSIDER BOTH TARGETED DROPPING AND DROPPING + POISONING SCENARIOS.

k	Target Class	No Attack FedAvg	Targeted Drop FedAvg	UpSample	Targeted Drop + Poison FedAvg	Clip	UpSample	Clip + UpSample
					EMNIST			
	0	0.47/ 0.66	0.09/ 0.18	0.18/ 0.44	0.00/ 0.00	0.00/ 0.06	0.01/ 0.00	0.03/ 0.31
9	1	0.75/ 0.92	0.09/ 0.25	0.39/ 0.76	0.00/ 0.00	0.00/ 0.04	0.01/ 0.00	0.04/ 0.30
	9	0.43/ 0.56	0.01/ 0.17	0.09/ 0.40	0.00/ 0.00	0.00/ 0.01	0.01/ 0.00	0.00/ 0.23
	0	0.58/ 0.78	0.19/ 0.33	0.40/ 0.66	0.00/ 0.00	0.04/ 0.09	0.00/ 0.01	0.12/ 0.40
12	1	0.86/ 0.95	0.25/ 0.55	0.49/ 0.85	0.00/ 0.00	0.01/ 0.04	0.04/ 0.01	0.17/ 0.60
	9	0.53/ 0.67	0.06/ 0.31	0.28/ 0.52	0.00/ 0.00	0.00/ 0.02	0.02/ 0.01	0.07/ 0.29
	0	0.65/ 0.80	0.26/ 0.50	0.47/ 0.71	0.00/ 0.00	0.04/ 0.06	0.00/ 0.03	0.22/ 0.35
15	1	0.91/ 0.96	0.47/ 0.79	0.83/ 0.94	0.00/ 0.00	0.06/ 0.18	0.05/ 0.04	0.53/ 0.51
	9	0.65/ 0.75	0.15/ 0.39	0.39/ 0.59	0.00/ 0.00	0.02/ 0.03	0.02/ 0.07	0.19/ 0.40
					FashionMNIST			
	0	0.44/ 0.47	0.30/ 0.36	0.60/ 0.62	0.09/ 0.03	0.29/ 0.38	0.13/ 0.19	0.58/ 0.58
15	1	0.93/ 0.95	0.89/ 0.93	0.95/ 0.96	0.12/ 0.02	0.77/ 0.77	0.16/ 0.28	0.92/ 0.93
	9	0.88/ 0.87	0.81/ 0.81	0.92/ 0.93	0.13/ 0.04	0.70/ 0.62	0.09/ 0.19	0.90/ 0.89
					DBPedia			
	0	0.45/ 0.54	0.11/ 0.10	0.75/ 0.82	0.00/ 0.00	0.00/ 0.00	0.02/ 0.01	0.65/ 0.77
15	1	0.77/ 0.87	0.37/ 0.17	0.93/ 0.96	0.00/ 0.00	0.08/ 0.00	0.02/ 0.03	0.89/ 0.94
	9	0.52/ 0.60	0.24/ 0.17	0.84/ 0.91	0.00/ 0.00	0.00/ 0.00	0.02/ 0.02	0.64/ 0.62

TABLE VI
ACCURACY ON FULL TEST SET PRESENTED AT ROUNDS $T/2$ AND T, UNDER COMM_PLAIN SETTING. $T = 100$ FOR EMNIST, $T = 300$ FOR FASHIONMNIST AND DBPEDIA. WE CONSIDER BOTH TARGETED DROPPING AND DROPPING + POISONING SCENARIOS.

k	Target Class	No Attack FedAvg	Targeted Drop FedAvg	UpSample	Targeted Drop + Poison FedAvg	Clip	UpSample	Clip + UpSample
					EMNIST			
	0	0.92/ 0.95	0.88/ 0.90	0.89/ 0.93	0.86/ 0.87	0.85/ 0.88	0.86/ 0.87	0.86/ 0.91
9	1	0.94/ 0.96	0.87/ 0.90	0.90/ 0.95	0.85/ 0.85	0.84/ 0.86	0.85/ 0.86	0.85/ 0.89
	9	0.91/ 0.93	0.87/ 0.89	0.88/ 0.92	0.86/ 0.87	0.86/ 0.87	0.86/ 0.87	0.86/ 0.90
	0	0.93/ 0.96	0.89/ 0.92	0.91/ 0.95	0.86/ 0.87	0.86/ 0.89	0.86/ 0.87	0.87/ 0.92
12	1	0.95/ 0.97	0.89/ 0.93	0.91/ 0.96	0.85/ 0.86	0.84/ 0.86	0.85/ 0.86	0.86/ 0.92
	9	0.92/ 0.95	0.87/ 0.91	0.90/ 0.93	0.86/ 0.87	0.86/ 0.87	0.68/ 0.68	0.87/ 0.90
	0	0.94/ 0.96	0.90/ 0.93	0.92/ 0.95	0.86/ 0.87	0.86/ 0.88	0.86/ 0.88	0.89/ 0.91
15	1	0.95/ 0.97	0.91/ 0.95	0.94/ 0.96	0.85/ 0.86	0.85/ 0.88	0.85/ 0.86	0.90/ 0.91
	9	0.94/ 0.95	0.88/ 0.92	0.91/ 0.94	0.86/ 0.87	0.86/ 0.87	0.87/ 0.88	0.88/ 0.92
					FashionMNIST			
	0	0.81/ 0.83	0.79/ 0.82	0.81/ 0.84	0.76/ 0.78	0.79/ 0.82	0.76/ 0.79	0.81/ 0.84
15	1	0.83/ 0.85	0.82/ 0.85	0.82/ 0.85	0.74/ 0.76	0.81/ 0.83	0.75/ 0.78	0.82/ 0.84
	9	0.82/ 0.85	0.82/ 0.84	0.83/ 0.85	0.74/ 0.76	0.80/ 0.82	0.74/ 0.77	0.82/ 0.84
					DBPedia			
	0	0.93/ 0.94	0.91/ 0.91	0.95/ 0.96	0.89/ 0.90	0.90/ 0.90	0.89/ 0.90	0.94/ 0.96
15	1	0.95/ 0.96	0.92/ 0.91	0.96/ 0.97	0.88/ 0.89	0.90/ 0.90	0.89/ 0.89	0.95/ 0.96
	9	0.93/ 0.94	0.91/ 0.91	0.95/ 0.96	0.89/ 0.90	0.89/ 0.90	0.89/ 0.90	0.93/ 0.94

individual updates. Thus, UpSampling, defending against the dropping attack, improves the target model accuracy by an average (over the 3 classes) of 6.33% on EMNIST, 10.66% on FashionMNIST, and 24.66% on DBPedia for $k = 15$ compared to the original accuracy. Even under both targeted dropping and model poisoning attacks, the combined UpSampling defense and Clipping results in an average decrease of 5% on EMNIST and average increase of 9% on FashionMNIST and 16.66% on DBPedia over the 3 classes compared to the original accuracy. Interestingly, classes with lower original accuracy benefit more from the UpSampling strategy, with improvements as high as 34% (on DBPedia for class 9, original accuracy is increased from 60% to 94% with UpSampling under targeted dropping).

We observed that under-represented classes (in terms of number of clients holding samples from those classes) are impacted more by our attacks. To alleviate this problem, the server could identify new clients with data from the populations of interest, and add them to the set of clients participating in the FL protocol. In cross-device FL settings, servers typically have access to a large number of clients, and can make decisions on expanding the set of participating clients to improve accuracy on under-represented populations.

TABLE VII
ACCURACY ON TARGET CLASS PRESENTED AT ROUNDS $T/2$ AND T, UNDER COMM_ENC SETTING. $T = 100$ FOR EMNIST, $T = 300$ FOR FASHIONMNIST AND DBPEDIA. WE CONSIDER BOTH TARGETED DROPPING AND DROPPING + POISONING SCENARIOS.

k	Target Class	No Attack FedAvg	Targeted Drop		Targeted Drop + Poison			
			FedAvg	UpSample	FedAvg	Clip	UpSample	Clip + UpSample
EMNIST								
9	0	0.47/ 0.66	0.32/ 0.52	0.59/ 0.76	0.01/ 0.00	0.14/ 0.40	0.10/ 0.04	0.46/ 0.68
	1	0.75/ 0.92	0.58/ 0.76	0.88/ 0.96	0.04/ 0.00	0.24/ 0.52	0.16/ 0.04	0.67/ 0.88
	9	0.43/ 0.56	0.31/ 0.46	0.54/ 0.70	0.01/ 0.00	0.03/ 0.25	0.06/ 0.05	0.37/ 0.57
12	0	0.58/ 0.78	0.48/ 0.69	0.75/ 0.85	0.00/ 0.00	0.29/ 0.36	0.02/ 0.06	0.60/ 0.77
	1	0.86/ 0.95	0.77/ 0.91	0.94/ 0.97	0.00/ 0.00	0.47/ 0.43	0.20/ 0.12	0.82/ 0.92
	9	0.53/ 0.67	0.41/ 0.56	0.68/ 0.78	0.00/ 0.01	0.18/ 0.27	0.15/ 0.07	0.50/ 0.67
15	0	0.65/ 0.80	0.60/ 0.76	0.81/ 0.89	0.01/ 0.00	0.37/ 0.44	0.01/ 0.05	0.62/ 0.71
	1	0.91/ 0.96	0.88/ 0.94	0.96/ 0.97	0.04/ 0.00	0.60/ 0.48	0.12/ 0.06	0.90/ 0.94
	9	0.65/ 0.75	0.50/ 0.58	0.76/ 0.85	0.03/ 0.01	0.30/ 0.35	0.10/ 0.07	0.53/ 0.71
FashionMNIST								
15	0	0.44/ 0.47	0.40/ 0.39	0.69/ 0.71	0.13/ 0.02	0.38/ 0.36	0.25/ 0.17	0.64/ 0.69
	1	0.93/ 0.95	0.92/ 0.94	0.95/ 0.96	0.14/ 0.03	0.86/ 0.81	0.26/ 0.24	0.94/ 0.95
	9	0.88/ 0.87	0.85/ 0.82	0.93/ 0.94	0.20/ 0.05	0.77/ 0.59	0.19/ 0.23	0.91/ 0.92
DBPedia								
15	0	0.45/ 0.54	0.16/ 0.12	0.79/ 0.84	0.00/ 0.00	0.00/ 0.00	0.03/ 0.01	0.73/ 0.79
	1	0.77/ 0.87	0.39/ 0.22	0.95/ 0.97	0.00/ 0.00	0.07/ 0.00	0.10/ 0.05	0.92/ 0.95
	9	0.52/ 0.60	0.31/ 0.12	0.88/ 0.94	0.00/ 0.00	0.00/ 0.00	0.02/ 0.01	0.75/ 0.76

TABLE VIII
ACCURACY ON FULL TEST SET PRESENTED AT ROUNDS $T/2$ AND T, UNDER COMM_ENC SETTING. $T = 100$ FOR EMNIST, $T = 300$ FOR FASHIONMNIST AND DBPEDIA. WE CONSIDER BOTH TARGETED DROPPING AND DROPPING + POISONING SCENARIOS.

k	Target Class	No Attack FedAvg	Targeted Drop		Targeted Drop + Poison			
			FedAvg	UpSample	FedAvg	Clip	UpSample	Clip + UpSample
EMNIST								
9	0	0.92/ 0.95	0.91/ 0.94	0.93/ 0.95	0.86/ 0.87	0.88/ 0.92	0.87/ 0.87	0.91/ 0.94
	1	0.94/ 0.96	0.92/ 0.95	0.95/ 0.97	0.85/ 0.86	0.88/ 0.91	0.86/ 0.86	0.92/ 0.95
	9	0.91/ 0.93	0.90/ 0.92	0.93/ 0.95	0.86/ 0.87	0.86/ 0.90	0.87/ 0.88	0.90/ 0.93
12	0	0.93/ 0.96	0.92/ 0.95	0.94/ 0.96	0.86/ 0.87	0.90/ 0.92	0.87/ 0.88	0.92/ 0.95
	1	0.95/ 0.97	0.94/ 0.96	0.95/ 0.97	0.85/ 0.86	0.90/ 0.90	0.87/ 0.87	0.93/ 0.95
	9	0.92/ 0.95	0.91/ 0.94	0.94/ 0.96	0.86/ 0.87	0.88/ 0.90	0.88/ 0.88	0.91/ 0.94
15	0	0.94/ 0.96	0.93/ 0.95	0.95/ 0.96	0.86/ 0.87	0.90/ 0.92	0.87/ 0.88	0.92/ 0.95
	1	0.95/ 0.97	0.95/ 0.96	0.95/ 0.97	0.85/ 0.86	0.91/ 0.90	0.86/ 0.86	0.94/ 0.96
	9	0.94/ 0.95	0.92/ 0.94	0.94/ 0.96	0.87/ 0.87	0.89/ 0.91	0.88/ 0.88	0.92/ 0.95
FashionMNIST								
15	0	0.81/ 0.83	0.80/ 0.83	0.82/ 0.85	0.76/ 0.78	0.80/ 0.82	0.77/ 0.78	0.81/ 0.84
	1	0.83/ 0.85	0.83/ 0.85	0.82/ 0.85	0.75/ 0.76	0.82/ 0.83	0.76/ 0.78	0.82/ 0.85
	9	0.82/ 0.85	0.82/ 0.84	0.83/ 0.85	0.74/ 0.76	0.80/ 0.82	0.75/ 0.78	0.82/ 0.85
DBPedia								
15	0	0.93/ 0.94	0.91/ 0.91	0.95/ 0.96	0.89/ 0.90	0.90/ 0.90	0.90/ 0.90	0.95/ 0.96
	1	0.95/ 0.96	0.92/ 0.92	0.96/ 0.97	0.88/ 0.89	0.89/ 0.90	0.89/ 0.89	0.95/ 0.97
	9	0.93/ 0.94	0.92/ 0.91	0.95/ 0.96	0.89/ 0.90	0.89/ 0.90	0.89/ 0.89	0.93/ 0.95

Privacy-preserving FL. So far, we have discussed settings in which the server receives local model updates in all rounds of the FL protocol. However, to protect client privacy, it is common to deploy privacy-preserving FL protocols, based on Multi-Party Computation (MPC), such as [19], [39]. In MPC implementations, multiple parties will be involved in aggregation and the server only receives the global model at the end of each iteration. The server has therefore the same knowledge as the network-level adversary under encrypted communication when running the client identification protocol from Algorithm 2.

In Table IX, we present attack and defense results for this challenging setting. Similarly to the previous tables, we set the parameters to $k_N = k_P = 2k/3$ for EMNIST, and $k_N = k_P = k/3$ for both FashionMNIST and DBPedia. While under this setting there is essentially no difference in the effectiveness of the attacker models we consider, the defensive performance varies due to the limited knowledge. The results we observe for the UpSample and Clip + UpSample defensive mechanisms are, as we would expect, somewhere in between the COMM_ENC and COMM_PLAIN scenarios. For instance for EMNIST, with $k = 15$ on class 1 we obtain an average accuracy

level of 0.84, considerably higher than the 0.51 obtained under COMM_PLAIN, but also not as high as the average of 0.94 obtained with COMM_ENC, which is essentially the same result obtained without any attack.

VI. RELATED WORK

Given the distributed training process in FL, poisoning attacks represent an even larger threat than in traditional ML systems. For instance, poisoning availability attacks have been shown effective in Federated Learning in recent work [5], [6]. Targeted model poisoning attacks impact a small population, or introduce a backdoor in the models to mis-classify instances containing the trigger [3], [4], [7], [8]. Researchers also proposed methods to defend the FL protocol from adversaries, such as Byzantine-resilient or trust-based aggregation rules [28]–[31], [40]–[44]. However, [5] and [6] systematically analyzed Byzantine-robust aggregation schemes, showing that an adversary controlling compromised clients can bypass these defenses with poisoning availability attacks. Also, targeted poisoning attacks can bypass Byzantine-resilient aggregation, such as Krum [4]. Methods protecting from model poisoning include filtering of malicious gradients for availability attacks [5], [6], [9] and gradient clipping for targeted attacks [4], [7].

VII. CONCLUSION

We examined the effects that a network-level adversary can have on the final model's accuracy on a target population in cross-device FL. We proposed a new attack, based on a procedure for identifying the set of clients mostly contributing towards a target class of interest to the adversary, and showed that it can be amplified by coordinated model poisoning attacks. We showed that our attacks are effective for realistic scenarios where the communication is encrypted and the attacker has limited network visibility. We also explored defensive approaches, and found that our UpSampling mechanism can be extremely successful when paired with clipping of model updates, against very powerful network-level attacks.

ACKNOWLEDGEMENTS

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *AISTATS*. PMLR, 2017.

[2] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data Poisoning Attacks Against Federated Learning Systems," *arXiv:2007.08432*, 2020.

[3] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Model Poisoning Attacks in Federated Learning," in *NeurIPS SECML*, 2018.

[4] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*. PMLR, 2020.

[5] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," in *USENIX Security*, 2020.

[6] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[7] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv:1911.07963*, 2019.

[8] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the Tails: Yes, You Really Can Backdoor Federated Learning," in *NeurIPS*, 2020.

[9] J. Xu, S. Huang, L. Song, and T. Lan, "Signguard: Byzantine-robust federated learning through collaborative malicious gradient filtering," *CoRR abs/2109.05872*, 2021.

[10] Y. O. Basciftci, F. Chen, J. Weston, R. Burton, and C. E. Koksal, "How vulnerable is vehicular communication to physical layer jamming attacks?" in *IEEE VTC*, 2015.

[11] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill, "Bgp hijacking classification," in *TMA*, 2019.

[12] "Russia compromised core router in us energy attacks," https://www.computerweekly.com/news/252437089/Russia-compromised-core-router-in-US-energy-attacks, 2018.

[13] A. Walters, D. Zage, and C. Nita Rotaru, "A framework for mitigating attacks against measurement-based adaptation mechanisms in unstructured multicast overlay networks," *IEEE/ACM TON*, 2008.

[14] S. Jero, E. Hoque, D. Choffness, A. Mislove, and C. Nita-Rotaru, "Automated attack discovery in TCP congestion control using a model-guided approach," in *NDSS*, 2018.

[15] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *IEEE S&P*, 2017.

[16] B. Weintraub, C. Nita-Rotaru, and S. Roos, "Structural attacks on local routing in payment channel networks," in *IEEE S&B*, 2021.

[17] M. Jagielski, N. Jones, C.-W. Lin, C. Nita-Rotaru, and S. Shiraishi, "Threat detection for collaborative adaptive cruise control in connected cars," in *WiSec*. ACM, 2018.

[18] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," *Found. Trends Mach. Learn.*, 2021.

[19] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical Secure Aggregation for Privacy-Preserving Machine Learning," in *CCS*. ACM, 2017.

[20] "Coupon collector's problem," https://en.wikipedia.org/wiki/Coupon_collectors_problem.

[21] W. Xu and A. K. Tang, "A generalized coupon collector problem," *J. Appl. Probab.*, 2011.

[22] "Group testing," https://en.wikipedia.org/wiki/Group_testing.

[23] R. Dorfman, "The detection of defective members of large populations," *Ann. Math. Stat.*, 1943.

[24] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "ODSBR: an on-demand secure byzantine resilient routing protocol for wireless ad hoc networks," *ACM TISSEC*, 2008.

[25] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS OSR*, 2003.

[26] D. Obenshain, T. Tantillo, A. Babay, J. Schultz, A. Newell, M. E. Hoque, Y. Amir, and C. Nita-Rotaru, "Practical intrusion-tolerant networks," in *IEEE ICDCS*, 2016.

[27] M. Hollick, C. Nita-Rotaru, P. Papadimitratos, A. Perrig, and S. Schmid, "Toward a taxonomy and attacker model for secure routing protocols," *SIGCOMM CCR*, 2017.

[28] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," in *NeurIPS*, 2017.

[29] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The Hidden Vulnerability of Distributed Learning in Byzantium," in *ICML*, 2018.

[30] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates," in *ICML*. PMLR, 2018.

[31] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine Stochastic Gradient Descent," in *NeurIPS*, 2018.

[32] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *IJCNN*, 2017.

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

[34] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," in *arXiv:1708.07747*, 2017.

[35] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, 2015.

TABLE IX

Accuracy on target class presented at rounds $T/2$ and T, under the MPC scenario. $T = 100$ for EMNIST, $T = 300$ for FashionMNIST and DBPedia. We consider both Targeted Dropping and Dropping + Poisoning scenarios.

k	Target Class	No Attack FedAvg	Targeted Drop FedAvg	UpSample	Targeted Drop + Poison FedAvg	Clip	UpSample	Clip + UpSample
					EMNIST			
	0	0.47/ 0.66	0.32/ 0.52	0.39/ 0.67	0.01/ 0.00	0.14/ 0.40	0.02/ 0.00	0.25/ 0.55
9	1	0.75/ 0.92	0.58/ 0.76	0.72/ 0.93	0.04/ 0.00	0.24/ 0.52	0.01/ 0.02	0.49/ 0.73
	9	0.43/ 0.56	0.31/ 0.46	0.38/ 0.60	0.01/ 0.00	0.03/ 0.25	0.03/ 0.00	0.14/ 0.50
	0	0.58/ 0.78	0.48/ 0.69	0.50/ 0.75	0.00/ 0.00	0.29/ 0.36	0.02/ 0.02	0.40/ 0.62
12	1	0.86/ 0.95	0.77/ 0.91	0.85/ 0.96	0.00/ 0.00	0.47/ 0.43	0.00/ 0.03	0.63/ 0.78
	9	0.53/ 0.67	0.41/ 0.56	0.54/ 0.72	0.00/ 0.01	0.18/ 0.27	0.02/ 0.00	0.27/ 0.51
	0	0.65/ 0.80	0.60/ 0.76	0.69/ 0.84	0.01/ 0.00	0.37/ 0.44	0.01/ 0.04	0.43/ 0.63
15	1	0.91/ 0.96	0.87/ 0.93	0.92/ 0.96	0.04/ 0.00	0.60/ 0.48	0.01/ 0.00	0.73/ 0.84
	9	0.64/ 0.75	0.50/ 0.58	0.59/ 0.80	0.03/ 0.01	0.30/ 0.35	0.02/ 0.00	0.38/ 0.61
					FashionMNIST			
	0	0.44/ 0.46	0.40/ 0.41	0.61/ 0.70	0.12/ 0.04	0.39/ 0.35	0.04/ 0.05	0.60/ 0.65
15	1	0.93/ 0.95	0.92/ 0.94	0.95/ 0.96	0.11/ 0.03	0.86/ 0.81	0.14/ 0.06	0.93/ 0.95
	9	0.88/ 0.87	0.85/ 0.82	0.93/ 0.93	0.19/ 0.05	0.75/ 0.60	0.08/ 0.04	0.86/ 0.85
					DBPedia			
	0	0.45/ 0.54	0.14/ 0.12	0.70/ 0.81	0.00/ 0.00	0.00/ 0.00	0.00/ 0.00	0.61/ 0.77
15	1	0.77/ 0.87	0.48/ 0.32	0.92/ 0.96	0.00/ 0.00	0.07/ 0.00	0.00/ 0.03	0.87/ 0.93
	9	0.52/ 0.60	0.30/ 0.13	0.82/ 0.91	0.00/ 0.00	0.00/ 0.00	0.00/ 0.00	0.53/ 0.70

TABLE X

Accuracy on full test set presented at rounds $T/2$ and T, under the MPC scenario. $T = 100$ for EMNIST, $T = 300$ for FashionMNIST and DBPedia. We consider both Targeted Dropping and Dropping + Poisoning scenarios.

k	Target Class	No Attack FedAvg	Targeted Drop FedAvg	UpSample	Targeted Drop + Poison FedAvg	Clip	UpSample	Clip + UpSample
					EMNIST			
	0	0.92/ 0.95	0.91/ 0.94	0.91/ 0.95	0.86/ 0.87	0.88/ 0.92	0.87/ 0.87	0.89/ 0.93
9	1	0.94/ 0.96	0.92/ 0.95	0.93/ 0.96	0.85/ 0.86	0.88/ 0.91	0.85/ 0.86	0.90/ 0.93
	9	0.91/ 0.93	0.90/ 0.92	0.91/ 0.94	0.86/ 0.87	0.86/ 0.90	0.87/ 0.87	0.88/ 0.93
	0	0.93/ 0.96	0.92/ 0.95	0.92/ 0.95	0.86/ 0.87	0.90/ 0.92	0.86/ 0.87	0.91/ 0.94
12	1	0.95/ 0.97	0.94/ 0.96	0.95/ 0.97	0.85/ 0.86	0.90/ 0.90	0.85/ 0.86	0.91/ 0.94
	9	0.92/ 0.95	0.91/ 0.94	0.93/ 0.95	0.86/ 0.87	0.88/ 0.90	0.87/ 0.68	0.89/ 0.93
	0	0.94/ 0.96	0.93/ 0.95	0.94/ 0.96	0.86/ 0.87	0.90/ 0.92	0.87/ 0.88	0.91/ 0.94
15	1	0.95/ 0.97	0.95/ 0.96	0.95/ 0.97	0.85/ 0.86	0.91/ 0.90	0.85/ 0.86	0.92/ 0.95
	9	0.94/ 0.95	0.92/ 0.94	0.93/ 0.96	0.87/ 0.87	0.89/ 0.91	0.87/ 0.87	0.90/ 0.94
					FashionMNIST			
	0	0.81/ 0.83	0.80/ 0.83	0.82/ 0.85	0.76/ 0.78	0.80/ 0.82	0.75/ 0.77	0.81/ 0.84
15	1	0.83/ 0.85	0.83/ 0.85	0.82/ 0.85	0.74/ 0.76	0.82/ 0.83	0.75/ 0.76	0.82/ 0.85
	9	0.82/ 0.85	0.82/ 0.84	0.83/ 0.85	0.74/ 0.76	0.80/ 0.82	0.74/ 0.76	0.81/ 0.84
					DBPedia			
	0	0.93/ 0.94	0.91/ 0.91	0.95/ 0.96	0.89/ 0.89	0.90/ 0.90	0.89/ 0.90	0.94/ 0.96
15	1	0.95/ 0.96	0.93/ 0.92	0.96/ 0.97	0.88/ 0.89	0.89/ 0.90	0.88/ 0.89	0.95/ 0.96
	9	0.93/ 0.94	0.92/ 0.91	0.95/ 0.96	0.89/ 0.90	0.89/ 0.90	0.89/ 0.89	0.92/ 0.95

[36] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *EMNLP*. Doha, Qatar: ACL, 2014.

[37] N. Gupta, S. Liu, and N. H. Vaidya, "Byzantine fault-tolerant distributed machine learning using stochastic gradient descent (sgd) and norm-based comparative gradient elimination (cge)," *arXiv:2008.04699*, 2020.

[38] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *arXiv:2002.11497*, 2020.

[39] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, and S. Zeitouni, "SAFELearn: Secure Aggregation for private FEderated Learning," in *IEEE SPW*, 2021.

[40] R. Guerraoui, A. Guirguis, J. Plassmann, A. Ragot, and S. Rouault, "Garfield: System support for byzantine machine learning (regular pa-per)," in *DSN*. IEEE, 2021.

[41] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, and H. Li, "FL-WBC: Enhancing robustness against model poisoning attacks in federated learning from a client perspective," in *NeurIPS*, 2021.

[42] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *NDSS*, 2021.

[43] E. M. E. Mhamdi, S. Farhadkhani, R. Guerraoui, A. Guirguis, L.-N. Hoang, and S. Rouault, "Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning)," in *NeurIPS*, 2021.

[44] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "Distributed momentum for byzantine-resilient stochastic gradient descent," in *ICLR*, 2021.