SplitOut: Out-of-the-Box Training-Hijacking Detection in Split Learning via Outlier Detection

Ege Erdoğan, Unat Tekşen, M. Salih Çeliktenyıldız, Alptekin Küpçü, A. Ercüment Çiçek

23rd International Conference on Cryptology and Network Security (CANS)

September 26, 2024

Overview

In SplitNN the model is divided into two parts between the clients and the server.

⇒ What the client models learn is fully determined by the server.

A malicious server can launch a training-hijacking attack to

- infer private training data
- implement backdoor in client models

Methods for clients to detect such attacks have been proposed, but they rely on either

- the client tampering with the protocol, or
- fine-tuning of a large number of hyperparameters.

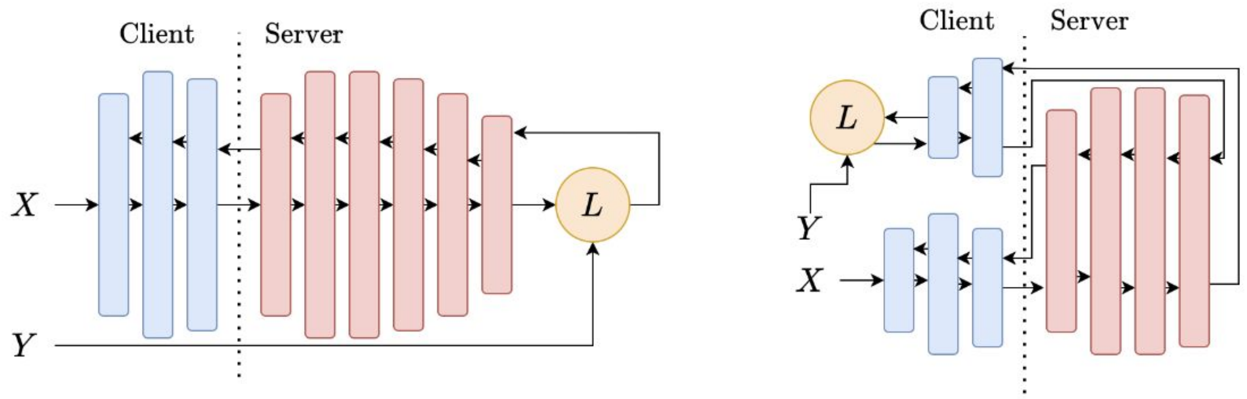


Fig 1. Potential SplitNN setups (with label sharing and without label sharing).

Under modest assumptions on clients' compute capabilities, an out-of-the-box outlier detection algorithm can achieve almost perfect accuracy in detecting training-hijacking attacks

Training-Hijacking

General attack paradigm: *feature-space-alignment + attack,* **although the client model is only trained during the alignment step.**

<u>Feature-Space Alignment</u>

The attacker has access to public data X_{pub} similar to client's private X_{priv}. Creates shadow model f_s to mimic the client model f_c.

Discriminator D trained to distinguish between $f_s(X_{pub})$ from $f_c(X_{priv})$.

Meanwhile f_c is trained to maximize D's error, i.e. make it harder to distinguish $f_s(X_{pub})$ from $f_c(X_{priv})$.

$\Rightarrow$ $f_s(X_{pub})$ and $f_c(X_{priv})$ overlap to a large extent, so that the attacker has "white-box" access to the client model.

<u>Attack Task</u>

Train an auto-encoder to reconstruct the inputs from the intermediate outputs [1,2], or train a binary backdoor classifier to detect backdoored samples for the backdoor attack [3].

[1] Pasquini, D. et al. Unleashing the tiger: Inference attacks on split learning in ACM CCS (2021), 2113–2129.
[2] Fu, J. et al. Focusing on Pinocchio's Nose: A Gradients Scrutinizer to Thwart Split-Learning Hijacking Attacks Using Intrinsic Attributes. in NDSS (2023).
[3] Yu, F. et al. How to Backdoor Split Learning. Neural Networks 168, 326–336 (Nov. 2023).

Previous Detection Methods

SplitGuard [4]: Clients randomize labels and measure how the gradients received from the server differ between those and non-randomized labels.

- Visible difference if the server is honest, none if malicious (attack discards labels).
  - **Weakness 1:** Critically dependent on some hyperparameters.
  - **Weakness 2:** Tampers with the training process, potentially detectable by the attacker, e.g. as in [2] by tracking the errors for different batches.

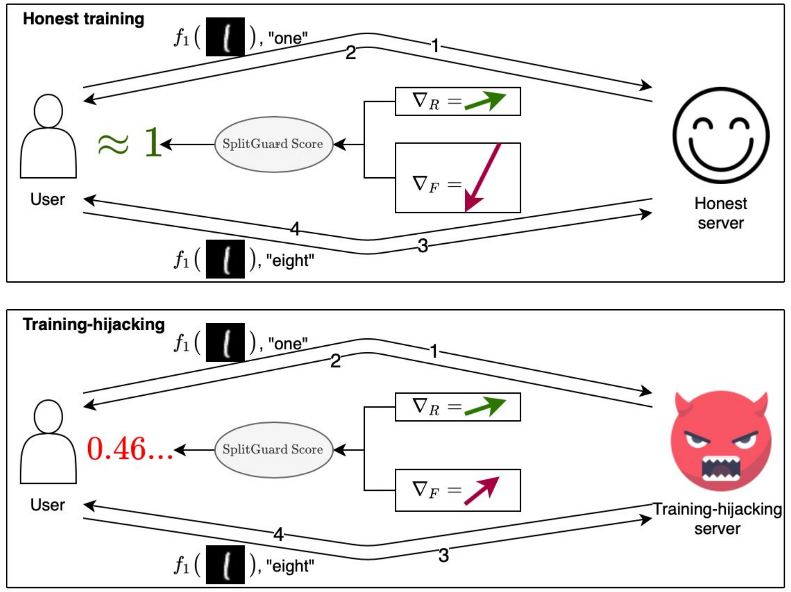


Fig 2. Overview of SplitGuard, from [4].

Gradients Scrutinizer [2] claims that *"gradients of the samples with the same label should be more similar than those of the samples with different labels."*

- Pairwise cosine similarities of the gradients from samples with two sets (same and different labels).
- Formula based on hyperparameters and these two sets, determines the attack according to pre-selected threshold.
  - **Weakness 1:** Abundance of hyperparameters.
  - **Weakness 2:** Lack of a learnable structure to them.

[2] Fu, J. et al. Focusing on Pinocchio's Nose: A Gradients Scrutinizer to Thwart Split-Learning Hijacking Attacks Using Intrinsic Attributes. in NDSS (2023).
[4] Erdogan, E. et al. SplitGuard: Detecting and Mitigating Training-Hijacking Attacks in Split Learning in ACM WPES (2022), 125–137.

Local Outlier Factor (LOF)

LOF:

- density-based,
- no distribution assumptions,
- unsupervised,
- assigns *local outlier scores* to points, points with lower local densities have lower scores,
- score ≤ 1 if inlier, > 1 if outlier

$$r(p,q) = \max(d(p,q), d_k(p))$$

Reachability distance

Euclidean distance

Distance to k'th closest neighbor

$$\mathrm{LRD}(p) = \left(\frac{1}{k} \sum_{q \in \mathrm{kNN(p)}} r(p,q) \right)^{-1}$$

Local reachability density

Inverse of the average reachability distance to k nearest neighbors

$$\mathrm{LOF}(p) = \frac{\sum_{q \in \mathrm{kNN}(p)} \mathrm{LRD(q)}}{k \cdot \mathrm{LRD}(p)}$$

LOF score

Ratio between average LRD of k nearest neighbors and the point itself

Since LOF only considers local densities, no global assumptions are made on attacker behavior, such as 'malicious gradients are distributed normally', 'malicious and honest gradients are linearly separable' etc.

SplitOut

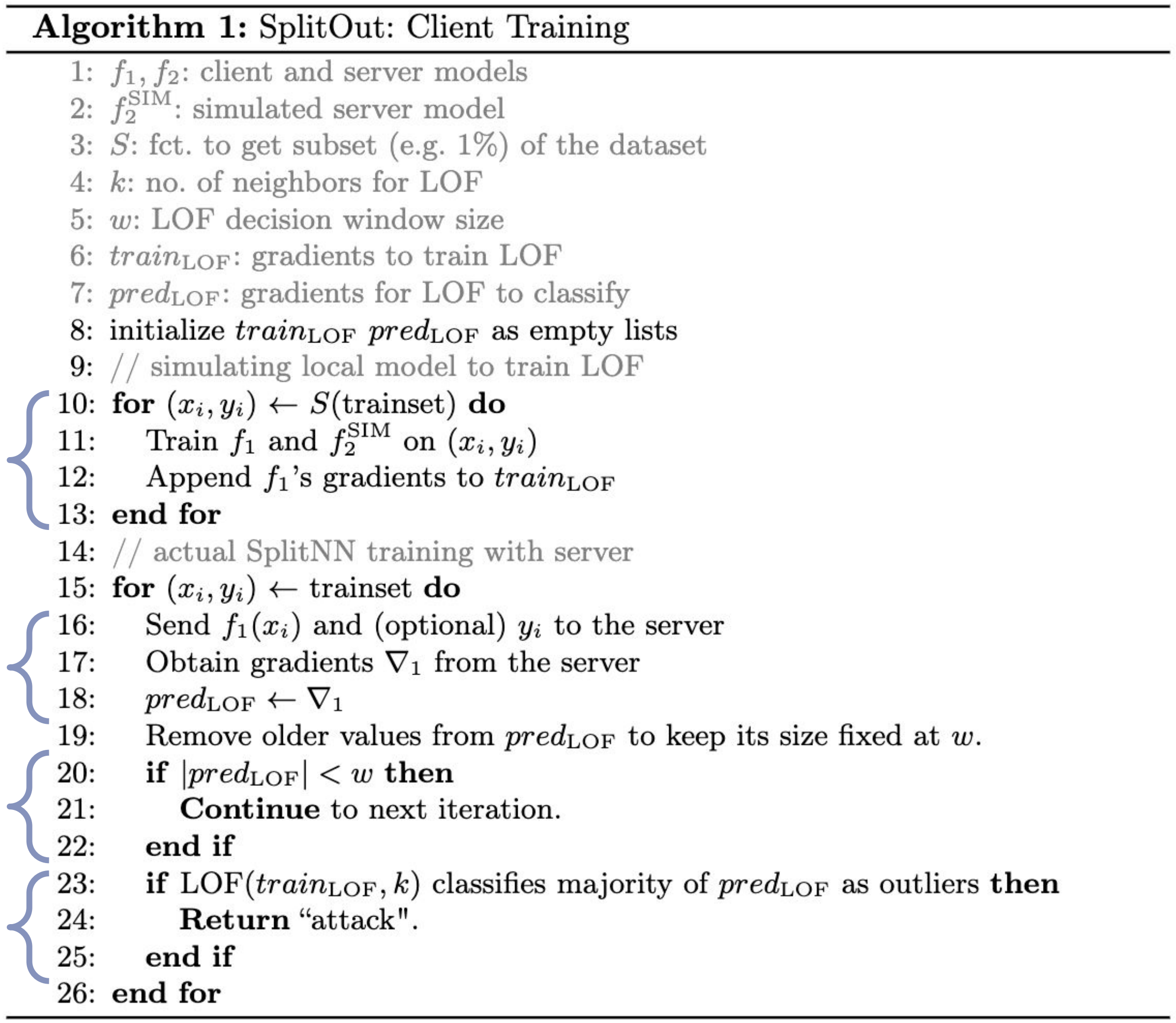
Core idea: Clients train their model on a very small subset of their data, using gradients as “normal” data to train an outlier detection method. Malicious gradients are then expected to be outliers.

1. Collect gradients by training the client model on a very small subset of data

2. Input the gradients received by the server to the outlier detection method.

3. Continue if not majority of gradients are classified as outliers.

4. Report attack if majority classified as outlier (malicious) in the window.



Algorithm 1: SplitOut: Client Training

```
$f_1, f_2$: client and server models
$f_2^{\text{SIM}}$: simulated server model
$S$: fct. to get subset (e.g. 1%) of the dataset
$k$: no. of neighbors for LOF
$w$: LOF decision window size
$train_{\text{LOF}}$: gradients to train LOF
$pred_{\text{LOF}}$: gradients for LOF to classify
initialize $train_{\text{LOF}}$ $pred_{\text{LOF}}$ as empty lists
// simulating local model to train LOF
for $(x_i, y_i) \leftarrow S(\text{trainset})$ do
    Train $f_1$ and $f_2^{\text{SIM}}$ on $(x_i, y_i)$
    Append $f_1$'s gradients to $train_{\text{LOF}}$
end for
// actual SplitNN training with server
for $(x_i, y_i) \leftarrow$ trainset do
    Send $f_1(x_i)$ and (optional) $y_i$ to the server
    Obtain gradients $\nabla_1$ from the server
    $pred_{\text{LOF}} \leftarrow \nabla_1$
    Remove older values from $pred_{\text{LOF}}$ to keep its size fixed at $w$.
    if $|pred_{\text{LOF}}| < w$ then
        Continue to next iteration.
    end if
    if LOF$(train_{\text{LOF}}, k)$ classifies majority of $pred_{\text{LOF}}$ as outliers then
        Return “attack".
    end if
end for
```

Why Does LOF Work? (1/2)

LOF is a distance-based method, so the curse of dimensionality dictates it should struggle in high dimensions.

However, honest and malicious gradients have noticeably different neighborhood characteristics, making them easy to distinguish by such a method.

Pairwise average distances between random sets of honest (H) and malicious gradients (M):

- Honest gradients are closer to honest gradients (top left quadrant darker than top right)
- Honest gradients more tightly grouped (top left darker than bottom right)
- Malicious gradients are closer to honest gradients than they are to malicious gradients (bottom left darker than bottom right).

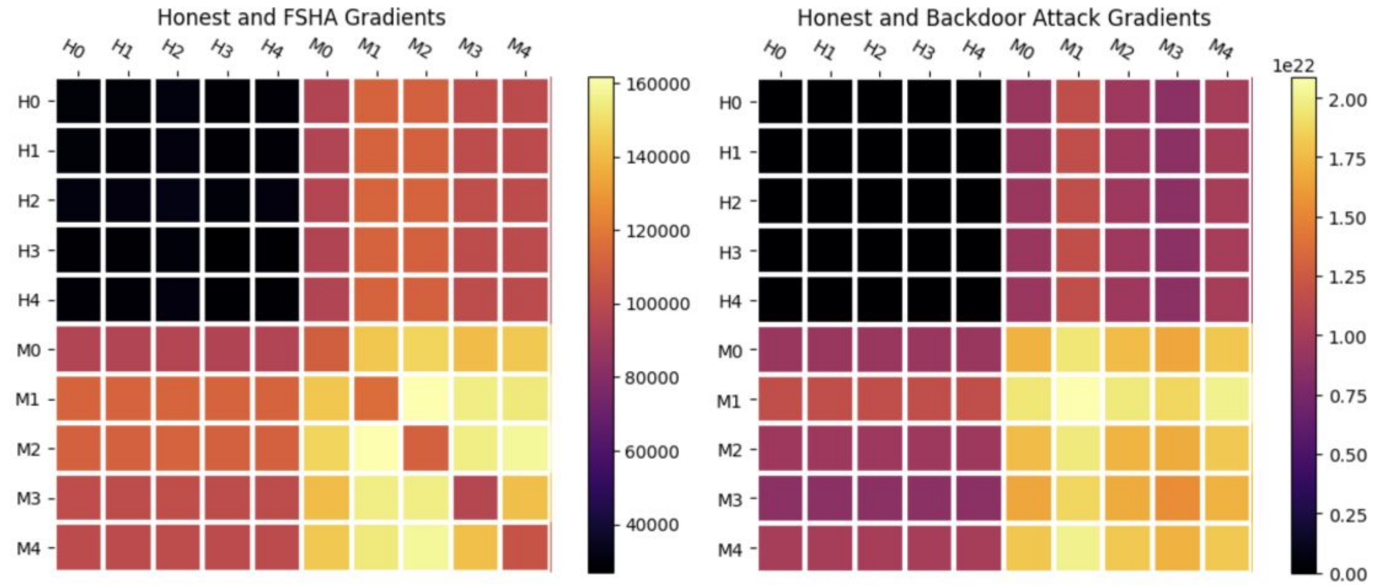


Fig 4. Pairwise average L2 distances between five randomly-chosen sets of gradients for honest training, FSHA and the backdoor attack.

Why Does LOF Work? (2/2)

The observations imply neighborhoods of honest gradients are densely populated with honest gradients, while the neighborhoods of malicious gradients are sparser, containing both malicious and honest gradients.

The t-SNE plots also support this hypothesis:

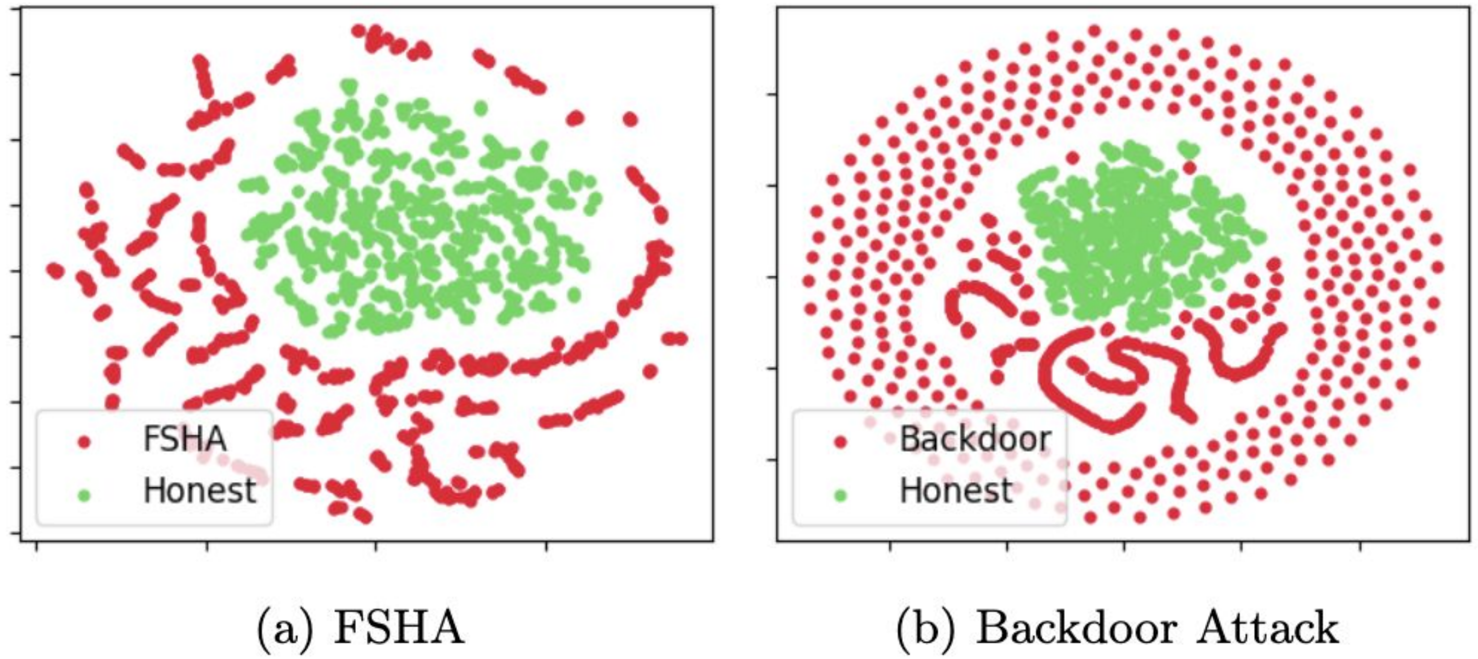


Fig 5. T-SNE dimension reduction comparing honest and malicious (FSHA & the backdoor attack) gradients (randomly chosen on first epoch) on CIFAR10.

Detection Accuracy

Results collected over 100 runs:

- All instances of [1, 2, 3] are detected using **1%** of the client's data to train LOF.
- Min. **98%** of attacks detected against a multitasking FSHA server.
- **No** honest servers are classified as malicious in (F-)MNIST, **2%** and **14%** in CIFAR10/100.
- Detection is early enough to prevent high-quality reconstructions by the attacker (Figure 3).

	FSHA		SplitSpy		Backdoor		FSHA-MT		
	TPR	*t*	TPR	*t*	TPR	*t*	TPR	*t*	FPR
MNIST	1	.011	1	.011	1	.011	1.00	.012	0
F-MNIST	1	.011	1	.011	1	.011	0.98	.029	0
CIFAR10	1	.013	1	.013	1	.013	0.98	.026	.02
CIFAR100	1	.013	1	.013	1	.013	1.00	.064	.14

Even 0.1% of client data might be enough, e.g. on EMNIST-digits against FSHA:

Data %	TPR	t (± SE)	FPR
.10	1	$4.1 \cdot 10^{-5} \pm 0$	0
.25	1	$4.1 \cdot 10^{-5} \pm 0$	0
1	1	$4.1 \cdot 10^{-5} \pm 0$	0

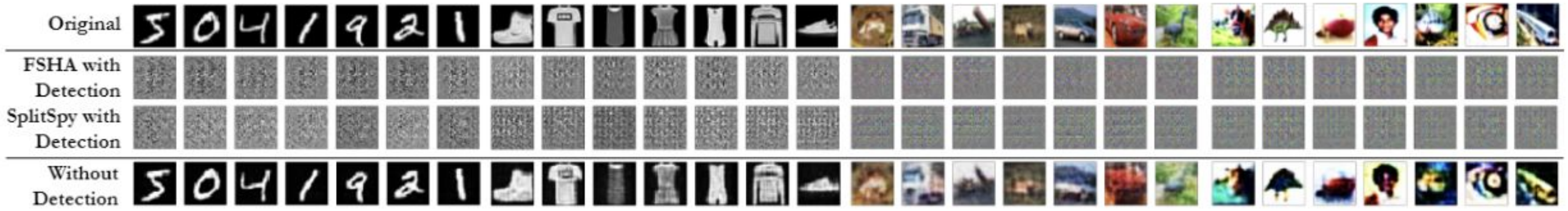


Fig 3. Results obtained by attackers for the MNIST, F-MNIST, CIFAR10, and CIFAR100

[1] Pasquini, D. et al. Unleashing the tiger: Inference attacks on split learning in ACM CCS (2021), 2113–2129.
[2] Fu, J. et al. Focusing on Pinocchio's Nose: A Gradients Scrutinizer to Thwart Split-Learning Hijacking Attacks Using Intrinsic Attributes. in NDSS (2023).
[3] Yu, F. et al. How to Backdoor Split Learning. Neural Networks 168, 326–336 (Nov. 2023).

Conclusion

SplitOut, using readily available algorithms, can achieve almost perfect accuracy in detecting training-hijacking attacks.

Stronger assumptions are required for the clients' compute capabilities, but this leads to a simpler method to implement and use.

Since our method does not rely on individual attacks, it can potentially detect future training-hijacking attacks.

Further experiments in the paper on:

- Attacks starting with the second epoch
- Clients not knowing the exact server model architecture
- Detection results for dataset (EMNIST-digits) with larger batch sizes

Questions/Limitations

Can there be training-hijacking attacks not doing feature-space alignment? Would those be harder to detect?

Although all attacks so far begin with the start of training, detection methods so far do not consider an attack starting at a much later time in training. Could those attacks starting perhaps after the client model has converged on the main task, be harder to detect?

Thank you! We are happy to take questions.

Paper: **https://arxiv.org/abs/2302.08618**
Code: **https://github.com/ege-erdogan/splitout**

Cryptography, Security, and Privacy Research Group

CICEKLAB

https://crypto.ku.edu.tr/
@crypto_koc_univ

http://ciceklab.cs.bilkent.edu.tr/
@crashius

GitHub

arXiv

We acknowledge the support of the Scientific and Technological Research Council of Turkey (TÜBİTAK) under project 119E088.