Gm Folks,

I’m applying to the EigenLayer fellowship and am interested in Federated learning after the reading the blog by AVS research.

https://www.blog.eigenlayer.xyz/eigenlayer-universe-15-unicorn-ideas/

Im going to describe my thought process below. Please note Im not an expert in Federated learning and am learning about it constantly so feel free to point out any mistakes or give suggestions.

One the major issues that federated learning currently faces is the existence of malicious clients that can severely decrease the performance of a model. I feel that EigenLayer can help solve that problem by establishing crypto-economic security.

There is a scheme called BGFLS (https://ieeexplore.ieee.org/document/9682070) that can backtrack malicious clients in an FL network. Clients that wish to train an FL model can opt in as AVS and need to restake some amount to participate. If we can efficiently and consistently identify malicious clients then we can slash their stake to deter them from sabotaging the model.

As @kydo pointed out in this post (https://forum.eigenlayer.xyz/t/federated-learning-protocol-secured-by-eigenlayer/402/3), a deterministic method to identify malicious behaviour and slashing them so the model has economic security would be better implemented on EigenLayer than a statistical approach.

Working of BGFLS

BGFLS is an architecture that groups clients and establishes a Model Parameters Sharing (MPS) Chain in each group to identify and backtrack malicious clients. This is done as backtracking algorithms are complicated and time-consuming with a large network of clients.

First, we randomly select clients by location and form a group. A MPS Chain is built on top of each group using Hyperledger fabric. This is done because we need a special block structure for the MPS Chain. Each client in a group gets a base model at genesis and locally trains the model to generate a set of update parameters. A random client in the group (Ci) initiates a model sharing request. A client Cj shares the latest parameters with Ci which is recorded in the MPS Chain as a transaction. After all parameters are shared and consensus is reached within the group, a new block is minted with model parameters of all clients. Ci also aggregates the local parameters and shares a new base model in the same block which is then trained upon by the rest of the clients.

The block structure has several bloom filters added by Ci to improve efficiency of backtracking malicious clients. A bloom filter is a probabilistic and efficient data structure to identify if an element is a part of a set. This part is used to identify the malicious parameters inside a group which we can verify and compare to find the malicious client.

A central server receives the aggregated parameters and checks similarity between different group parameters. If group A send parameters a = [a0, a1, a2, …] and group B send parameters b = [b0, b1, b2, …] then similarity is calculated by $\cos\theta = a.b / |a||b|$

.

If similarity to a particular group is very low compared to other groups then it’s classified as a malicious group. After identification, bloom filters can be used to check which parameters are malicious and verify to make sure which client is sending malicious parameters.

This scheme is explained in more detail in this research paper: A Blockchain-based Grouped Federated Learning Scheme Against Malicious Clients

(https://ieeexplore.ieee.org/document/9682070)

Privacy-Preserving FL Schemes

I also came across another scheme called BPFL (https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9685821) that uses homomorphic encryption to ensure that local model parameters are also private and hence no sensitive dataset is leaked.

Conclusion

As @sreeramkannan pointed out in this reply, we need to answer the following questions:

i) How is eigenlayer leveraged in FL schemes?

ii) How EL nodes can collect data to train local models?

iii) What kind of applications would an FL scheme be suitable for?

I think the BGFL Scheme illustrates how it can leverage economic security from Eigenlayer to prevent Byzantine attacks and

I would love to implement a V0 in the next few days to test out this architecture.

I, however, am curious to know how the other two questions can be answered. Please let me know your thoughts on this idea and correct me if I'm wrong anywhere.

Telegram: yash2399

Twitter: 0xpanicError