Practical Secure Aggregation for Privacy-Preserving Machine Learning

Why do we need Federated Learning?

Federated learning is a machine learning approach that allows a model to be trained across multiple decentralized edge devices (such as smartphones, IoT devices) without exchanging raw data.

In many scenarios, especially with sensitive or personal data (e.g., healthcare records, user behavior on devices), privacy is a significant concern. Federated learning allows the training of models directly on user devices without the need to share raw data, addressing privacy issues associated with centralized data storage, and saving the computing and storage resources of the centralized cloud server.

Scenario

When we use a smartphone's keyboard to type words, there is a useful function which predicts the most possible word we want to type, and this is actually an ai model behind this function. The ai model needs to use our typing words as the training data to make its prediction more accurate. However, text messages frequently contain sensitive information; users may be reluctant to upload a copy of them to the modeler’s servers.Therefore, we consider training such a model in a Federated Learning setting, wherein each user maintains a private database of her text messages securely on her own mobile device, and a shared global model is trained under the coordination of a central server based upon highly processed, minimally scoped, ephemeral updates from users.

From Federated Learning to Secure aggregation

Secure aggregation is specifically designed to implement federated learning on mobile devices.Mobile devices have only sporadic access to power and network connectivity, so the set of users participating in each update step is unpredictable and the system must be robust to users dropping out. Because the neural network may be parameterized by millions of values, updates may be large, representing a direct cost to users on metered network plans. Mobile devices also generally cannot establish direct communications channels with other mobile devices (relying on a server or service provider to mediate such communication) nor can they natively authenticate other mobile devices. Thus, Federated Learning motivates a need for a Secure Aggregation protocol that:
(1) operates on high-dimensional vectors
(2) is highly communication efficient, even with a novel set of users on each instantiation
(3) is robust to users dropping out
(4) provides the strongest possible security under the constraints of a server-mediated, unauthenticated network model

Cryptographic Primitives of Secure Aggregation

1. secret sharing:

This technology involves splitting a secret "s" into "n" shares and setting a threshold value "t" The original secret "s" can only be reconstructed if one has obtained equal to or more than "t" shares. It is designed to address the issue of disconnections— even if some users go offline, it is still possible to reaggregate parameter values and share private keys when the number of available shares meets or exceeds the threshold.

2. key agreement:

This discusses methods of key exchange, which involves how encryption keys for transmitting information among participants are exchanged.Which is actually the use of Diffie-Hellman key agreement method.

3. Authenticated Encryption
   (Symmetric) Authenticated Encryption combines confidentiality and integrity guarantees for messages exchanged between two parties.
4. Pseudorandom Generator
5. Signature Scheme
6. Public Key Infrastructure
   To prevent the server from simulating an arbitrary number of clients (in the active-adversary model), we require the support of a public key infrastructure that allows clients to register identities, and sign messages using their identity

Algorithm&Proof

This paper's secure aggregation is achieved by a 5-rounds protocol.

- Round 0 (AdvertiseKeys):
  for every user u(client in the federated learning) , it will use key agreement protocol like diffie hellman to generate 2 pairs of key $< C_u^{PK}, C_u^{SK} >$ and $< S_u^{PK}, S_u^{SK} >$, key C will be used to encrypt the message between any two of the users, key S will be used to generate one of the masks which can protect our ai models' weights.

  And use u use a private key d_u^{SK} (this key is the private key of asymmetric key encryption like RSA, and it's public key d_u^{PK} is sign by Certificate Authority, so other one can use d_u^{PK} to make sure the signature is made by u) to sign C_u^{SK} , generate the signature σ_u.

  every user u send <C_u^{PK}, S_u^{PK}, σ_u> to the server, and then the server will collect those messages and then send them to every user.

- Round 1 (ShareKeys):
  every user u receive <C_v^{PK}, S_v^{PK}, σ_v> of every other user v from the server, and use d_u^{PK} and σ_v to make sure C_v^{PK} and S_v^{PK} are correct and not be tampered.
  every user u generate the share t-out-of-n shares of S_u^{SK}, and t-out-of-n shares of b_uThese two are the seeds to put into the pseudo random generator to generate masks to protect ai models' weights.
  So, for every other user v, we generate $S_{u,v}^{SK}$ and $b_{u,v}$, means the share of S_u^{SK} which is give to user v and the share of b_u which is give to user v.
  Then use C_v^{PK} and C_u^{SK} (Diffie–Hellman) to get the key $C_{u,v}$, and use $C_{u,v}$ to encrypt $S_{u,v}^{SK}$ and $b_{u,v}$, which ensure only user v can know the value of $S_{u,v}^{SK}$ and $b_{u,v}$, so user u generate ciphertexts $e_{u,v} = ENCRYPT(key = DH(C_v^{PK}, C_u^{SK}),\ text = < S_{u,v}^{SK}, b_{u,v} >)$ for every user v.

user u send $e_{u,v}$ for every user v to the server

- Round 2 (MaskedInputCollection):
  user u receive $e_{v,u}$, the ciphertexts from every user v, from the server, and
  $< S_{v,u}^{SK}, b_{v,u} > \ = \ DECRYPT(key = DH(C_u^{SK}, C_v^{PK}),\ text \ = e_{v,u})$ get the two shares $S_{v,u}^{SK}$ and $b_{v,u}$
  user u have an ai model weights x_u, calculate

  $$y_u = \ x_u + PRG(b_u) + \sum_{v\ in\ all\ other\ user} PRG(DH(S_v^{PK}, S_u^{SK})) * (1\ if\ u < v\ else\ -1)$$

  the propose of $(1\ if\ u < v\ else\ -1)$ is to make the below equation hold.

  $$\sum_{u\ in\ all\ user} \sum_{v\ in\ all\ other\ user} PRG(DH(S_v^{PK}, S_u^{SK})) * (1\ if\ u < v\ else\ -1) = 0$$

  the propose of PRG (Pseudo Random Generator) is to take one seed as input, generate a huge amount of random numbers to add on every weight of an ai model.
  And every user u send y_u to the server.
- Round 3 (ConsistencyCheck):
  server receive y_u from every user u and make a set U represent the remain user, and
  send this U to every user u.
  every user u use d_u^{SK} to sign the remain user set U, make the signature σ'_u , and send σ'_u to the server.
  server receive signature σ'_u and collect all σ'_u and the remain user set U, then send all signatures and the remain user set U to every user

  This step's target is to let every user have a consensus on who remains and who drops out.
- Round 4 (Unmasking):
  user u Receive a set U from the server. check all of the signatures σ'_v append with the U by d_v^{SK} .
  for all users remain, send the share $b_{v,u}$ to server, for all users drop out, send the share $S_{v,u}^{SK}$ to server
  server receive at least t user return the corresponding $b_{v,u}$ and $S_{v,u}^{SK}$.
  for those users v still remain, reconstruct the b_v by at least t shares $b_{v,u}$, for those users v drop out, reconstruct the S_v^{SK} by at least t shares $S_{v,u}^{SK}$
  calculate the summation of x_u by

  $$\sum_{u\ in\ all\ user} y_u - \sum_{u\ in\ all\ remain} PRG(b_u) - \sum_{u\ in\ all\ drop\ out} PRG(DH(S_v^{PK}, S_u^{SK})) * (1\ if\ u < v\ else\ -1)$$

  and then securely get the summation of the models.