$CSE598\_HW4$

Kamal

April 2021

1 Question

Please see Python file for code Worked with Glen

2 Question

Main Source
The main phases for efficiently performing zero sharing with very larger numbers of parties are conditional zero-sharing phase in which an arbitrary number of parties have to create secret shares of zero and conditional reconstruction, where each party uses OPPRF to send its share to $party_1$ on a value that every party uses. An OPPRF builds on an Oblivious PRF, and a PRF is a puesdo-random-function that takes a seed and a label as input and outputs a value of length N (Source).

The main component that enables efficient sampling of zero shares is Cuckoo Hashing implemented on the Receiver's end in the OPPRF protocol, which in this case, each entry has only one element and when all the hashes are full, normally the extra conflicting elements are placed in a separate structure (the stash). However, in this implementation does not use a stash as the stash reduces performance as more searches are required to find an element and instead uses three hash functions. The result of the OPPRF with Cuckoo Hashing protocol are the following, where b is the sender's b_{th} bin in the hashmap and the Receiver generates a series of of inputs $(q_1, q_2, ..., q_t)$

- Sender receives a PRF key corresponding to the b_{th} bin and a hint for the b_{th} bin
- Receiver receives a hint for the b_{th} bin and F(k_b, $hint_b$, and the item in the Sender's b_{th} bin

3 Question

Vinayak and Anirudh presented CRYPTFLOW: Secure TensorFlow Interfence, which proposed a privacy preserving interface for TensorFlow and since deep neural networks are the most utilized ML model, privacy preserving DNN's have increased importance among other privacy preserving ML models. One of the main ideas of the paper was to compile otherwise default Tensorflow functions into privacy preserving equivalents. The first step was to change float variables to fixed variables and eventually, the model trains on fixed point data which does reduce accuracy, which is a tradeoff. The low level assembly code will integrate more crypto features and of the crypto topics covered in class, secret sharing is utilized in the paper for Matmul and RELU for example. Additionally, secure multiplication is included in the neural network, like in Convolutional Layers of a CNN. We did not cover how privacy machine learning models interact with hardware and what performance and security concerns may arise.

Later, the authors establish two general approaches (pure crypto and hardware) to make the model more resistant to malicious adversaries and proporses an approach that combines crypto and hardware security. A secure hardware approach provides data integrity and however, is potentially vulnerable to side channel attacks. On the other hand, a pure crypto approach has decent runtime performance when implemented to be resistant to purely semihonest adversaries and has a signifcant runtime performance cost when tailored to be additionally resistant against malicious adversaries as well.