Crypto Final Project Report

Brendon Pon

Link to Git Repo: https://github.com/brendonpon/Crypto.git

Implementation of Edit Distance:

My implementation of edit distance was pretty simple I simply choose to have the characters atgc represented by the numbers 0, 1, 2, 3 respectively. Although the exact association doesn't actually matter. Thus the inputs from the parties are two strings of length 5 made up of numbers. The program just outputs a number between 0 and 5 inclusive that represents the edit distance. I also only implemented substitution edit distance so there is no shifting or deletions involved (i.e. ED(12345, 23451) = 5, instead of 2).

Basic Lookup:

The second function I implemented was a lookup method. The idea behind this is that one party has a list of keys and corresponding data. The second party has a key. They want to know the corresponding data or if no such data exists. This is akin to a variable size oblivious transfer but the second party will receive all matching data corresponding to the provided key or if no corresponding data exists, the equivalent of a null response. In my implementation. The data and keys are all just numbers but this could easily be extended to arbitrary data types. Perhaps the most direct real world application of this is the government or other agencies searching databases, records, or manifests. If the government wants to know which flight a particular person was on. They could provide the name of the person and the airlines would be providing the list of keys (names) and data (flights). This way the government could get the information they needed without knowing the information of everyone else, and the airlines would not know which person the government was interested in. This would work similarly with other types of databases and queries. It's possible that this (or something similar) could even work towards general searches. Say if you want to google something but you don't want anyone to know what you're looking for. You could hypothetically perform a similar secure computation and get your search results without the other party knowing more than needed. However the issue with this is that as you increase the size of the database or pool of information you're trying to draw from, the amount of complexity needed to encrypt that data becomes worse and worse. So while this may work in theory, chances are it wouldn't be practical for any sufficiently large process. Thus this application is much more useful for searching through smaller lists (such as flight records etc).

I did start to try and write a machine learning protocol. The issue that I ran into was that it was nearly impossible to do anything meaningful with the computation. The program was taking to long to compute with any reasonable size data set. I was only using a relatively small data set as well (just over a hundred data points with under 10 dimension data points). I also was just

trying to code a basic perceptron. The circuit blowup seemed kind of ridiculous. It’s possible that I could have reduced some of the time through smarter use of smaller values than just floats or actually trying to code at the circuit level to reduce redundancies. I also think that the program was running exceedingly slow by some aspect of the docker. But even the other simple programs that I would sometimes hang for a while. But in any case it seems that im you want to efficiently implement secure machine learning for anything meaningful, you need to either reduce definitions of security of use a different method to secure computing to reduce the amount of computations necessary. Because I was not able to fully test my protol and get everything working I didn't include the code for the perceptron. But I found it interesting and thought it deserved a mention.