CS 203.4860 **Secure Multi-Party Computation** Fall 2021

Homework 5: Report

Lecturer: Dr. Adi Akavia *Student(s): ()*

Work space:

We used Python to implement BeDOZa Protocol to achieve security against malicious adversaries using MACs to authenticate secret shares.
This implementation is the same one as homework 4, it's an extended protocol for general functionality.

Classes,Functions and implementation:

TVK class:

In this class we used MACs to authenticate secret shares.
init function: in this function we get the keys of both Bob and Alice and value and prime p and we calculate Tag and save it in variable t.
add function:loading of the adding symbol, used as the additive secret sharing schema.
mul const function: Multiplying the share by a constant, the input parameter i is the indicator of which player running this commend (Alice or Bob).
add const function: Adding the share by a constant, the input parameter i is the indicator of which player running this commend (Alice or Bob).
str function:Helper function for printing the TVK object which consist of key,value and tag.
and function:Loading of the and symbol for two values and returns null/none if not valid.
kless function: returns TVK object with same tag and value and p.
TKVpair function: returns the pairs $t_{A,x}$, $t_{B,x}$ as taught in class.

BeDOZa Dealer:

We deal with this class as the dealer in this class through the function init we define random alpha1, alpha 2 (random values between 0 and p-1) and define aTag0, aTag1 to be $t_{A,x}$, $t_{B,x}$ and add it to our tags arrays for Alice and Bob. and we define c that is equal to a*b mod p and we called TVK to add $t_{A,x}$, $t_{B,x}$ for Alice and Bob arrays. We defined also RandA and RandB functions which return the arrays of random values to Alice and Bob and the tags .

BeDOZa Agent:

Instead of defining classes as Alice and Bob we defined BeDOZa Agent which has same functionality as Alice and Bob, We added init function to this class that initializing the circle and all the inputs. Same Send and Receive functions

test circuit:

circuit is just layers of arrays of tuples (op,inp1,inp2),this circuit test most of the features we support.
gt4Circ compares if $value \geq 4 (\text{mod } 251)$.
the rest functions are for implementing the circuit we defined in previous assignment but with p=251.
mainTest:Gets the inputs and ciruit and check the protocol with p=59.

Test and complexity:

We tried all the possible inputs of the Equation3 with 100 repetitions of the tests (To eliminate the effect of randomness) and checked if all the circuit outputs are right through these inputs, and measured the communication complexity , we always got 43 for average and maximal communication per activation.

Execution Time:

We also calculated the executing time by using package time and subtract the start time from the time when we end running the tests and we got different times of execution such as 27.04992651939392 seconds, 31.670392274856567 seconds, 25.458529949188232 seconds (for $100 \cdot 4^4$ repetitions).