Secure multi-party computation

Assignment 3

Najah Kamal 325829133

Salam Qais 327876116

Sami Serhan 327876298

Semester 1

We wrote the code according to Equation 3:

For $d = 2$ the function is $f_{\vec{a},4}\colon \{0,1,2,3\}^2 \to \{0,1\}$ defined by

$$f_{\vec{a},4}(x_1, x_2) = \begin{cases} 1 & \text{if } a_1x_1 + a_2x_2 \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

In addition, we use the Boolean circuit that we have wrote in assignment 1.

Now, after we have the Boolean circuit we can write this circuit using a code, such that the computation will be secure. We do so according to the algorithm that we have learned in the lecture. Therefore, we created 3 classes: the dealer, Alice and Bob.

- Offline Phase:
  Now we are talking about the dealer class:
  - Firstly, we sample a “Beaver triples”: u,v ← R {0,1}, then we find w, which is w = u · v mod 2. We do this action t times, when t is the number of AND gates in our Boolean circuit.
  - After that, for each u, v and w (t variables of u, t variables of u and t variables of w), we do secret share by choosing u, v, and w for Alice randomly, and then giving Bob Alice values XOR the original values of u, v, w.
  - Then, we have two functions rand_a which sends massage to Alice, sends the shares $(u.A, u.A, w.A)$, and rand_b send to bob $(u.B, v.B, w.B)$.
  - In addition, we have an array where we define our Boolean circuit.
- We have for class Alice many functions, such as:
  - Initial function that initializes class Alice according to the inputs.
  - A function that sends Bob Alice's shares.
  - A function that receives Bob's shares.
  - A function that receives list of the result that bob computes.
  - A function that sends Bob Alice's computation.
  - A function that check if we are in the last layer.
  - A function that returns an output.
- Bob has functions some functions that are similar to Alice's function but not all of them.

The correctness of BeDoza:

In order to prove the correctness of BeDoza the entire protocol, we have first to show the correctness of each one of the subprotocols:

- Sharing input wires:
  - $Share(A, x_i)$: $Alice\ computes\ (x_{iA}, x_{iB}) \leftarrow Shr(x_i)\ and\ sends\ x_{iB} to\ Bob$
    Alice sample x_{iA} randomly, and sends Bob $x_{iB} = x_i \oplus x_{iA}$.
  - $Share(B, x_i)$: $Bob\ computes\ (x_{iA}, x_{iB}) \leftarrow Shr(x_i)\ and\ sends\ x_{iA} to\ Alice$
    Bob sample x_{iB} randomly, and sends Alice $x_{iA} = x_i \oplus x_{iB}$.

- Opening secret shared values:
  - $OpenTo(A, [x])$: $Bob\ sends\ x_B\ to\ Alice$
    $Alice\ outputs\ x = x_A \oplus x_B$
    We didn't write a OpenTo function by itself, but we wrote the things that it do in our code, by using the functions send and receive, so Alice and Bob can send each other their input, and the receiver get the value and he compute the real value.
    Note: It is possible to copy the lines of code that: receive from Bob the input and calculates XOR between the input and its value (what OpenTo does), and make a new function and call it OpenTo.

  - $OpenTo(B, [x])$: $Analogus$

  - $Open([x])$:
    We didn't write a Open function by itself, rather in the main function in order to open secret shared values, we call send and receive values, it is equal to OpenTo functions, and then we can computes what Open function do.
    Note: It is possible to copy the lines of code that: do OpenTo for Alice and foe Bob and put it in one function, and call it Open.

- $XOR([x], c)$:
  $Alice\ outputs\ z_A = x_A \oplus c$

  $Bob\ outputs\ z_B = x_B$

  $\rightarrow z_A \oplus z_B = x_A \oplus c \oplus x_B = x_A \oplus x_B \oplus c$

- $XOR([x], [y])$:
  $ALice\ outputs\ z_A = x_A \oplus y_A$
  $Bob\ outputs\ z_B = x_B \oplus y_B$
  $\rightarrow z_A \oplus z_B = (x_A \oplus y_A) \oplus (x_B \oplus y_B)$
  $= x_A \oplus x_B \oplus y_A \oplus y_B$

  $\rightarrow= x \oplus y$

- $AND([x], c)$:
  $Alice\ outputs\ z_A = c * x_A$
  $Bob\ outputs\ z_B = c * x_B$
  - If c=0: $\rightarrow AND([x], c) = 0$

$z_A = c * x_A = 0,$
$z_B = c * X_B = 0$
$\rightarrow z_A * z_B = 0 * 0 = 0 = AND([x], c)$

- If c=1: $\rightarrow AND([x], c) = [x]$
  $z_A = c * x_A = x_A,$
  $z_B = c * x_B = x_B$
  $\rightarrow z_A * z_B = x_A * x_B = AND([x], c)$

- $AND([x], [y])$:
  $[d] \leftarrow XOR([x]. [u])$
  $d \leftarrow Open([d])$
  $[e] \leftarrow XOR([y], [v])$
  $e \leftarrow Open([e])$
  Compute: $[z] = [w] \oplus (e * [x]) \oplus (d * [y]) \oplus (e * d)$
  We do so using subprotocols $XOR\ and\ AND$.

The correctness of Share, OpenTo, Open: follows from the correctness of secret sharing scheme.

The correctness of $XOR([x], c), XOR([x], [y]), AND([x], c)$ we have shown above:

- The correctness of $AND([x], [y])$:
  $AND([x], [y]) = w \oplus ex \oplus dy \oplus\ ed$
  $= uv \oplus (yx + vx) \oplus (xy + uy) \oplus (xy + uy + xv + uv)$
  $= xy$

Now, the correctness of entire protocols:

Theorem: For every wire w in the circuit, the parties holds a secret-sharing $[v_w]$of the value v_w on that wire.

We will prove it by induction on the number of wires in the circuit:

Base case: Input wires: in the input wires it is true, because our share at first is true.

Induction step – XOR & AND gates: In every gate (XOR or AND) that we add, the condition of the induction is maintained according to the lemma of every gate (XOR or AND).

Therefore, we get that it is true for every gate in the circuit → it is true for all the circuit and all the wires, in particular it is true for the output wire, so when we do open to the output wire we get the true value.

Corollary: Opening the output wire returns the correct circuit output.

The privacy of the BeDoza:

We have to show that whatever Alice and Bob see when they participate in the protocol, we can simulate without participating in the protocol and without additional information.

- Simulated view for Alice:
  $S_A(x, f(x,y))$ output a simulated view consisting of:
  1. Alice's input x.
  2. Alice's randomness: S_A run $Share(A, x_i)$, add used randomness to vies.
  3. Alice's receives messages:
     S_A plays dealer's role:
     Sample t Beaver triples, secret-share and add to vies Alice's shares (u_A, v_A, w_A).
     S_A plays Bob's role in input sharing Share(B,x_i):
     Sample and add to view: n random bits $r_1.r_2, \dots, r_n$ in place of Alice's shares for Bob's input.
     S_A plays Bob's role in $Open(A, [d])$ and $Open(A, [e])$ during $AND([x],[y])$ gate evaluation:
     Sample and add to view random bits d_B and e_B.
     S_A plays Bob's in output wire opening:
     Add to view the value $x_B^L = x^L \oplus X_A^L$

  - Simulates View ≡ Real View:
    We next argue that simulated $\equiv_{prefect}$ real view.
    Input and randomness are sampled identically to the real view.
    Message receives:
    Simulated messages for the t Beaver Triples shares (u_A, v_A, w_A) and Bob's input shares $r_1, \dots, r_n$ are generated as in real protocol – identically distributed.
    Simulated messages (shares) for d_B and e_B are random bits; the real messages are $d_B = x_B \oplus u_B$ and $e_B = y_B \oplus v_B$ for random bits u_B, v_B – identically distributed.
    Simulated message (share) for Bob's output wire is $x_B^L = x^L \oplus x_A^L$; in the real protocol: $x^L = x_B^L \oplus x_A^L$ – identically distributed.

- Simulated view for Bob:
  $S_B(y, f(x,y))$ output a simulated view consisting of:
  1. Bob's input y.
  2. Bob's randomness: S_B run $Share\ (B, y)$, add used randomness to view.
  3. Bob's received messages:
     S_B plays dealer's role:
     Sample t Beaver triples, secret-share and add to view Bob's shares (u_B, v_B, w_B).
     S_B plays Alice's role in input sharing Share(A.y):
     Sample and add to view n random bits $r_1, .., r_n$ in place of Bob's shares for Alice's input.
     S_B plays Alice's role in Open(B,[d]) and Open(B,[e]) during AND([x],[y]) gate evaluation:
     Sample and add to view random bits d_A and e_A.

S_B plays Alice's in output wire opening:
Add to view the value $x_A^L = x^L \oplus x_B^L$.

- Simulates View ≡ Real View:
  We next argue that simulated $\equiv_{prefect}$ real view.
  Input and randomness are sampled identically to the real view.
  Message receives:
  Simulated messages for the t Beaver Triples shares (u_B, v_B, w_B) and Alice's input shares $r_1, \dots, r_n$ are generated as in real protocol – identically distributed.
  Simulated messages (shares) for d_A and e_A are random bits; the real messages are $d_A = x_A \oplus u_A$ and $e_A = y_A \oplus v_A$ for random bits u_A, v_A – identically distributed.
  Simulated message (share) for Bob's output wire is $x_A^L = x^L \oplus x_B^L$; in the real protocol: $x^L = x_B^L \oplus x_A^L$ – identically distributed.