Multi-Party Computation Coursework
Privacy Engineering 70018

Tuck Hong (tkh2017) and Preet Lalli (pl1516)

Thursday, 19 November 2020

1 Introduction

The Ben-Or, Goldwasser, Wigderson (BGW Protocol) is an example of Multi-Party Computation with Secret Sharing. In this coursework, the BGW protocol is implemented and used to evaluate functions that have been compiled into arithmetic circuits.

2 BGW Protocol

2.1 Overview

The BGW Protocol involves n parties evaluating a function of their n private inputs without requiring any party to disclose their private input values to any other party. This operates with perfect security provided that less than half of the parties involved are semi-honest.

Each party computes a polynomial with its private value as the constant value $P(0)$ and randomly generated numbers as the coefficients of the remaining powers of x. The party then sends a share using this polynomial to every party $i, i = 1, 2...n$ where the share sent is equal to $P(i)$. On receipt of the n shares, each party evaluates identical arithmetic circuits and broadcasts its output value to all the other parties. The output value of each party is in fact a share of the complete output value of the function. Using Lagrange Interpolation on the n received shares, each party can then reconstruct the true output value of the function.

For the 3 circuits evaluated in this coursework, the degree of the polynomials each party generates is 2 which means that a minimum of 5 parties are required to successfully evaluate the circuits as they all contain MUL gates.

2.2 ADD Gate

The ADD gate simply computes the modular addition over a prime number of the two input shares. This is a correct calculation for the addition of two polynomials as for any x:

$$h(x) = f(x) + g(x)$$

Therefore it follows that this is true for $x = i$. For example in the case of party 1, where $f(1)$ and $g(1)$ are 2 of its received shares:

$$h(1) = f(1) + g(1)$$

Since the two input shares are points on randomised polynomials whose constant terms are the private values, adding the two shares corresponds to adding the polynomials and hence, will result in a new randomised polynomial whose constant term is the sum of the private values.

2.3 MUL Gate

The MUL gate computes the modular multiplication over a prime number of the two input shares and then does degree reduction. This method is described in the code below.

```
def evaluate_mul(a, b, gate_no, network):
    share = mul(a,b)
    subshares = split_share(share)
    receivedshares = {}
    for p in ALL_PARTIES:
        network.send_share(subshares[p], gate_no, p)
        receivedshares[p] = network.receive_share(p, gate_no)
    outputshare= lagrange_interp(receivedshares)
    return outputshare
```

Degree reduction is necessary as multiplication produces an output polynomial with double the degree of the input polynomials. The degree would continue to increase exponentially if there are several MUL gates in sequence in the arithmetic circuit.

As such, the output of the modular multiplication is taken to be the constant value of a new polynomial of degree equal to the input polynomial. Similar to the treatment of the initial private value, random coefficients for this new polynomial are generated and then, n subshares are distributed to all parties. Finally, Lagrange Interpolation is performed on the n subshares received to construct the output of the MUL gate.

3 Circuits

3.1 Circuit 1

This circuit is the implementation of the function on p.445 of Cryptography Made Simple by Nigel Smart. It can be interpreted as the dot product of 2 vectors of 3 dimensions each.

$$f(x_1, ..., x_6) = x_1 \times x_2 + x_3 \times x_4 + x_5 \times x_6$$

As seen on line 1 of the log (Figure 1), the secret value of party 1 is 20 and it has generated a random polynomial $P(x) = 20 + 17x + 72x^2$ where the secret value is $P(0)$. After generating and distributing shares using this polynomial, party 1 receives the shares listed on line 2 and computes $x_1 \times x_2$ (mod p) through the first MUL gate.

$$8 \times 58 = 464$$

In this circuit $p = 101$ therefore:

$$464 \equiv 60$$

This value is the constant term of the random polynomial seen on line 4. To complete the degree reduction, party 1 once again generates, distributes and receives subshares. The result of the Lagrange Interpolation of the subshares on line 5 is 77 as seen on line 6.

After evaluating the intermediate MUL and ADD gates, the inputs to the final ADD gate are 80 and 44.

$$80 + 44 = 124$$

$$124 \equiv 23 \pmod{101}$$

This is the output share of party 1 which it broadcasts to the other parties. Finally, the final result to the function is obtained through Lagrange Interpolation of the output shares received from all parties which is 7 as seen on line 19. Using the private values of the parties:
$x_1 = 20, x_2 = 40, x_3 = 21, x_4 = 31, x_5 = 1, x_6 = 71$, it is easy to check that this is correct.

$$20 \times 40 + 21 \times 31 + 1 \times 71 = 1522$$

$$1522 \equiv 7 \pmod{101}$$

```
01-001: Random polynomial: 20 + 17x^1 + 72x^2
01-002: Received shares are: {1: 8, 2: 58, 3: 25, 4: 99, 5: 11, 6: 53}
01-003: Evaluating MUL gate
01-004: Random polynomial: 60 + 97x^1 + 8x^2
01-005: Received shares are: {1: 64, 2: 40, 3: 86, 4: 69, 5: 87, 6: 5}
01-006: MUL result is: 77
01-007: Evaluating MUL gate
01-008: Random polynomial: 51 + 32x^1 + 15x^2
01-009: Received shares are: {1: 98, 2: 81, 3: 38, 4: 52, 5: 96, 6: 63}
01-010: MUL result is: 68
01-011: Evaluating MUL gate
01-012: Random polynomial: 78 + 63x^1 + 97x^2
01-013: Received shares are: {1: 36, 2: 52, 3: 40, 4: 4, 5: 82, 6: 83}
01-014: MUL result is: 80
01-015: ADD result is: 44
01-016: ADD result is: 23
01-017: Received output shares are: {1: 23, 2: 40, 3: 58, 4: 77, 5: 97, 6: 17}
01-018: The recombination vector is: {1: 6, 2: 86, 3: 20, 4: 86, 5: 6, 6: 100}
01-019: The final output is: 7
```

Figure 1: Log of party 1's evaluation of circuit 1

3.2 Circuit 2

The second circuit calculates the factorial of a power of 2. In the specific example below, 8! is calculated using 8 parties where each party has a private value corresponding to its party number.

```
INPUTS = 2 ** 3
PRIVATE_VALUES = {k: k for k in range(1, INPUTS+1)}
```

As seen in Figure 2, the only gates that are used in this circuit are MUL gates, structured as a tree.

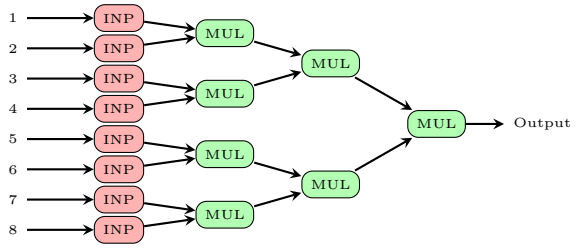

Figure 2: Factorial tree for 8 parties.

As before, each party must first split their own private value into shares and then distribute these shares to the other parties. For example, looking at line 1 in Figure 3, since party 1 has a private value of 1 and the degree chosen is 2, the random polynomial that party 1 generates is $1 + 17611x + 74606x^2$. Using this polynomial, party 1 then generates and distributes shares by plugging into x, values from 1 to 8. As seen in line 2, party 1 receives a share of 92218 for its own private value (1 + 17611 + 74606).

After each party has received their shares from the other parties, they can then begin evaluating the circuit. The first gate to evaluate is a MUL gate, multiplying the inputs of party 1 and party 2. Each party does the multiplication using their share of party 1 and 2's inputs. For example, party 1 does the following computation (mod 100003):

$$92218 \times 19418 \equiv 1790689124 \equiv 35406$$

As described earlier, when evaluating a MUL gate, a degree reduction step is needed which requires each party to split and distribute their multiplication result. For example, in line 4, party 1 uses 35406 to generate a new random polynomial of degree 2, $35406 + 8271x + 33432x^2$, which is then used to generate subshares.

After receiving the subshares of the multiplication result, each party has to recombine the subshares using Lagrange interpolation. For example, in line 5, party 1 received these subshares: {1: 77109, 2: 74230, 3: 52746, 4: 57645, 5: 29776, 6: 95037, 7: 29388, 8: 83716}. Using the recombination vector shown in line 32, party 1's result of the first MUL gate is found to be (mod 100003):

$$8 \times 77109 + 99975 \times 74230 + 56 \times 52746 + 99933 \times 57645$$

$$+56 \times 29776 + 99975 \times 95037 + 8 \times 29388 + 100002 \times 83716 \equiv 14968$$

```
01-001: Random polynomial: 1 + 17611x^1 + 74606x^2
01-002: Received shares are: {1: 92218, 2: 19418, 3: 8868, 4: 70696, 5: 15127, 6: 85777, 7: 62224, 8: 78272}
01-003: Evaluating MUL gate
01-004: Random polynomial: 35406 + 8271x^1 + 33432x^2
01-005: Received shares are: {1: 77109, 2: 74230, 3: 52746, 4: 57645, 5: 29776, 6: 95037, 7: 29388, 8: 83716}
01-006: MUL result is: 14968
01-031: Received output shares are: {1: 37960, 2: 68682, 3: 32483, 4: 29366, 5: 59331, 6: 22375, 7: 18501, 8: 47709}
01-032: The recombination vector is: {1: 8, 2: 99975, 3: 56, 4: 99933, 5: 56, 6: 99975, 7: 8, 8: 100002}
01-033: The final output is: 40320
```

Figure 3: Log of evaluating circuit 2 from party 1's point of view.

The remaining MUL gates in the circuit are evaluated in a similar fashion and as seen in line 33, the final output from party 1 was 40320 which is the correct answer (8!).

3.3 Circuit 3

The final circuit is a circuit that evaluates the following function:

```
def function(x):
    return (div((x[1] + x[2] + x[3] + x[4] + x[5] + x[6]), 6)) % PRIME
```

This function can be interpreted to be the average of the private values of the 6 parties, and will yield the correct answer in modular arithmetic when the sum is a multiple of 6. In the specific example below, the sum of the private values is 24, and hence, the average should be 4.

```
PRIVATE_VALUES = {1:4, 2:5, 3:3, 4:2, 5:7, 6:3}
```

The circuit makes use of 5 ADD gates and a DIV-by-6 gate which is just a MUL-by-a-constant gate where the constant is the multiplicative inverse of 6. In this case, a prime of 101 is chosen, so the multiplicative inverse is 17.

As before, each party must first split and distribute their own private value. For example, in line 1 in Figure 4, party 1 uses the random polynomial $4 + 17x + 72x^2$ to generate a share of 93 for itself (4 + 17 +72). Once the shares from all other parties have been received, circuit evaluation can begin. In this circuit, all the received shares are simply added up and then passed into a DIV-by-6 gate.

```
01-001: Random polynomial: 4 + 17x^1 + 72x^2
01-002: Received shares are: {1: 93, 2: 23, 3: 7, 4: 70, 5: 17, 6: 86}
01-003: ADD result is: 15
01-004: ADD result is: 22
01-005: ADD result is: 92
01-006: ADD result is: 8
01-007: ADD result is: 94
01-008: Evaluating DIV gate
01-009: Random polynomial: 83 + 97x^1 + 8x^2
01-010: Received shares are: {1: 87, 2: 28, 3: 59, 4: 93, 5: 52, 6: 9}
01-011: DIV result is: 89
01-012: Received output shares are: {1: 89, 2: 45, 3: 74, 4: 75, 5: 48, 6: 94}
01-013: The recombination vector is: {1: 6, 2: 86, 3: 20, 4: 86, 5: 6, 6: 100}
01-014: The final output is: 4
```

Figure 4: Log of evaluating circuit 3 from party 1's point of view.

As seen in line 7, the result of the additions is 94. This value will then be passed into a DIV-by-6 gate which is just a MUL-by-17 gate for the chosen prime of 101. Hence, for party 1, the multiplication result is $94 \times 17 \equiv 83 \pmod{101}$. However, the MUL gate also has a degree reduction step, so each party has to split and distribute their result once again. For example, in line 9, party 1 splits the result of 83 using a random polynomial $83 + 97x + 8x^2$. After receiving all the subshares from all parties, the output of the DIV-by-6 gate is found to be 89.

Finally, after receiving all the output shares, the final output can be obtained using the recombination vector in line 13. For example, party 1's final output is calculated as follows (mod 101):

$$6 \times 89 + 86 \times 45 + 20 \times 74 + 86 \times 75 + 6 \times 48 + 100 \times 94 \equiv 4$$