MPC Review

March 21, 2025

Recall Shamir Secret Sharing

1) Share:
    a) Choose random poly with degree at most t-1 over GF[p] s.t. $f(0) = m$
    b) $f(x) = m + {}_{i=1}\Sigma^{t-1}\, a_i\, x^i$
    c) Send $f(i)$ to party i
2) Reconstruct:
    a) Given t shares, we have t linear equations with t unknowns
    b) Solve the system over each i: $s_i = {}_{j=0}\Sigma^{t-1}\, a_j\, (i)^j$
    c) a_0 is the message
    d) Lagrange Interpolation

BGW

1) Secret share your input with a *(t+1)*-out-of-*n* scheme (i.e. Shamir)
2) Compute on each share
3) Reconstruct the result

BGW Invariant

1) At every stage, we require our shares to be:
   a) Random polynomials
   b) Degree at most t
   c) Post-computation are random shares of the output given the inputs from the input shares

BGW Computation: Addition

1) Given share g_a and g_b:
   a) $g_{a+b} = g_a + g_b$
2) Total sum is:
   a) $= f_a(i) + f_b(i)$
   b) Random (random + random is random)
   c) Degree at most t
   d) Constant term is $a + b$

BGW Computation: Scalar Multiplication

1) Given share g_a and constant c:
   a) $g_{ca} = cg_a$
2) Result is:
   a) $= cf_a(i)$
   b) Random (random times constant is random)
   c) Degree at most t
   d) Constant term is ca

BGW Computation: Multiplication

1) Requires interaction
    a) Rerandomization
    b) Degree reduction
2) To start, we have:
    a) $g_{ab} = g_a g_b$
    b) Degree *2t* poly
    c) Not necessarily random

Rerandomization

1) Each party i chooses random $2t$-degree poly g_i' such that $g_i'(0) = 0$
2) Sends $g_i'(j)$ to party j
3) Each party j rerandomizes by computing $g_{ab} + {}_{i=1}\Sigma^n\, g_i'(j)$

Degree Reduction

1) There is a matrix A that can reduce *2t*-degree polynomials and reduce them to degree *t*
2) $(g_{ab}'(1), \ldots, g_{ab}'(n))^T$ = A $(g_{ab}(1), \ldots, g_{ab}'n))^T$
3) Can compute this securely with MPC
   a) All linear operations so it is not an infinite loop
4) A = V P V^{-1}
   a) P is 1s on the first *t* diagonals and 0 elsewhere
   b) V is the Vandermonde matrix
   c) Essentially, deconstruct the poly into the coefficients, select the coefficients with degree at most *t*, and then reconstruct the poly

Private Analytics

Goal: Compute aggregate statistics on millions of users

Idea 1: Use simple operations in MPC (i.e. no multiplication)

Idea 2: Use client-server model to remove interaction between all users and eliminate member churn

Example: Identify home page

Protocol

1) 2 Servers that are always on (Server A and B)
2) Clients secret share data to both servers
    a) Shares are a one-hot vector indicating what home page you are using
3) A and B compute the sum of their shares and publish the results
4) Joining them together will produce the desired results

Problems

1) Consistency - ensure every share is received by both servers
    a) Send share A and enc(Share B) to server A and delegate them to distribute
2) Defend against malicious client - Ensure that the share is indeed one hot
    a) Next slide

Verification of One Hot Share

1) Compute random vector r
2) Set R to be r^2 (pointwise squared)
3) Server i computes:
    a) $t_i = \langle r, x_i \rangle$
    b) $T_i = \langle R, x_i \rangle$
4) MPC to check that $(t_a + t_b)^2 = (T_a + T_b)$
5) Does not protect privacy against a malicious server:
    a) Can add a minus 1 to each slot and 1 elsewhere, when protocol accepts, found the index of the 1!

Problems

(Majority as an arithmetic circuit). For odd n, the majority function takes as input n binary values $x1, \ldots, xn \in \{0, 1\}$. It outputs 1 if the majority of the inputs is 1, and outputs zero otherwise. Show how to implement the majority function as an arithmetic circuit over Zq, for some prime $q > n$.

$$p(x) = \sum_{k=\frac{n+1}{2}}^{n} \prod_{j=\frac{n+1}{2}, j \neq k}^{n} \frac{x-j}{k-j}$$