- Scribes?
- BGW protocol
  - multiparty protocol
  - round complexity linear in circuit depth
  - unconditionally secure if $t < n/2$ parties corrupted
  - arithmetic circuits over $\mathbb{F}_p$
    - wires hold field elements
    - gates are addition/multiplication over $\mathbb{F}_p$

---

Polynomials over $\mathbb{F}_p$

- degree-$t$ ~~polynomial~~ non-zero polynomial has at most $t$ roots
- Any collection of $t+1$ pairs $\{\underline{(x_1, y_1), (x_2, y_2), \ldots, (x_{t+1}, y_{t+1})}\}$,
  where $x_i$ are distinct,
  defines a unique degree-$t$ polynomial $f$
  such that $\underline{f(x_i) = y_i}$ for all $i$

---

$l_i$ – want $f(x_i) = 1$
$f(x_j) = 0$ for $j \neq i$

$$l_i(X) = \frac{\prod_{j \neq i} (X - x_j)}{\prod_{j \neq i} (x_i - x_j)}$$

– degree-$t$ polynomial

$$\underline{f(X) = \sum_{i=1}^{t+1} l_i(X) \cdot y_i}$$

Say $g$ is a degree-$t$ polynomial
w/ $g(x_i) = y_i$ for all $i$.
Then $f(X) - g(X) = h(X)$ satisfies
$h(x_i) = 0$ for all $i$ $\Rightarrow$ $h$ is 0-poly

For any $x \in \mathbb{F}_p$, $f(x)$ can be written as a linear combination of the $\{y_i\}$.

$$f(x) = \sum_{i=1}^{t+1} l_i(x) \cdot y_i$$

<u>$(t+1)$-out-of-$n$ secret sharing (Shamir)</u>

Given a secret value $s$

- define $f(X) = f_t X^t + f_{t-1} X^{t-1} + \dots + f_1 X + s$,

  where $f_t, \dots, f_1 \leftarrow \mathbb{F}_p$, $p > n$
- give share $f(i)$ to $P_i$, for $i = 1, \dots, n$

- Reconstruction?
- Secrecy?

<u>BGW protocol</u>

Invariant: parties will hold $(t+1)$-out-of-$n$ shares of the values on the wires of the circuit.

<u>Step 1</u>: Input sharing

if $P_i$ holds input $x_i$, they use Shamir secret sharing to obtain shares $x_1, \dots, x_n$ and send $x_i$ to $P_i$.

I.e., $P_i$ chooses $f$ s.t. $f(0) = x$; sets $x_i = f(i)$ for $i = 1, \dots, n$.

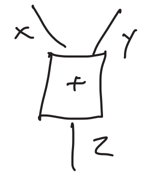
$[x] \stackrel{\text{df}}{=} x$ is appropriately shared

$= (x_1, \dots, x_n)$, w/ $x_i$ known to $P_i$, that lie on a degree-$t$ polynomial $f$ with $f(0) = x$

Step 2: evaluation

<u>multiplication by a public constant</u> – easy

addition gate




parties hold $[x] = (x_1, \ldots, x_n)$ s.t. $f(i) = x_i$, $f(0) = x$

$[y] = (y_1, \ldots, y_n)$ s.t. $g(i) = y_i$, $g(0) = y$

$f, g$ degree-$t$

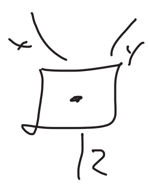
want to compute $\underline{[z] = (z_1, \ldots, z_n)}$ s.t. $h(i) = z_i$, $h(0) = z = x + y$

$h$ degree-$t$

each party locally computes $\underline{z_i = x_i + y_i}$

this defines the polynomial $h(X) = f(X) + g(X)$

multiplication gate




(also possible to handle division)

parties hold $(x_1, \ldots, x_n)$, $f(i) = x_i$, $f(0) = x$

$(y_1, \ldots, y_n)$, $g(i) = y_i$, $g(0) = y$

parties can locally set $h_i = x_i \cdot y_i$

let $\boxed{h(X) = f(X) \cdot g(X)}$ then $h(i) = h_i$ for all $i$

$h(0) = f(0) \cdot g(0) = x \cdot y$

degree of $h$ is $2t$

now, parties hold $(h_1, \ldots, h_n)$ s.t. $h(i) = h_i$, $h(0) = z$,

degree of $h$ is $2t$

we know that $z = h(0) = \sum_{i=1}^{2t+1} L_i \cdot h_i$, note $t < n/2$

each party will share $h_i$ $\Rightarrow$ parties (collectively) hold

$[h_1], [h_2], \ldots, [h_n]$

using local computation, parties can compute

$[L_1 \cdot h_1], [L_2 \cdot h_2], \ldots, [L_n \cdot h_n]$

using local computation, parties can compute

$$\left[\sum_{i=1}^{2t+1} L_i h_i\right]$$

---

$(x_1, \ldots, x_n)$ - degree-$t$ poly $f$, $f(0)=x$

$(y_1, \ldots, y_n)$ - degree-$t$ poly $g$, $g(0)=y$

$\Downarrow$

$(h_1, \ldots, h_n) = (x_1 y_1, x_2 y_2, \ldots, x_n y_n)$ - degree-$2t$ poly $h = g \cdot f$, $h(0) = xy = \underline{z}$

$h_1 \rightarrow (h_{1,1}, \ldots, h_{1,n})$ - degree-$t$ poly $h_1$ s.t. $h_1(0) = h_1$

$\vdots$

$h_n \rightarrow (h_{n,1}, \ldots, h_{n,n})$ - degree-$t$ poly $h_n$ s.t. $h_n(0) = h_n$

<u>Step 3</u> Output reconstruction

if $P_i$ is supposed to learn some output value $y$, parties send their shares of $y$ to $P_i$ . . .

## Beaver triples

have $[x]$, $[y]$

want to compute $[z]$, where $z = xy$ ← Beaver triples

assume parties hold $\boxed{[a],\ [b],\ [c]}$,

where $c = ab$ and $a, b$ are uniform in $\mathbb{F}_p$

**Step 1:** publicly reconstruct the values

$$x-a, \quad y-b$$

$$[x], [a] \Rightarrow [x-a] \xrightarrow{\text{reveal shares}} x-a$$

$$[y], [b] \Rightarrow [y-b] \xrightarrow{\text{reveal shares}} y-b$$

**Step 2:** parties locally compute

$$[z] = [c] + \underbrace{(y-b)\cdot[x]} + \underbrace{(x-a)\cdot[y]} - \underbrace{(x-a)\cdot(y-b)}$$

$$[c] + [(y-b)\cdot x] + [(x-a)\cdot y] - [(x-a)\cdot(y-b)]$$

$$[c + (y-b)\cdot x + (x-a)\cdot y - (x-a)\cdot(y-b)]$$

$$[\cancel{ab} + xy - \cancel{bx} + \cancel{xy} - \cancel{ay} - \cancel{xy} + \cancel{ay} + \cancel{bx} - \cancel{ab}]$$

$$= [xy]$$