Last time: FSS

$$\begin{tikzcd}
 & f_0 \arrow[r] & f_0(x) \arrow[dr] & \\
f \arrow[ur] \arrow[dr] & & & f(x) \\
 & f_1 \arrow[r] & f_1(x) \arrow[ur] &
\end{tikzcd}$$

This time: Homomorphic secret sharing (HSS)

$$\begin{tikzcd}
 & x_0 \arrow[r] & y_0 = \mathrm{Eval}(P, x_0) \arrow[dr] & \\
x \arrow[ur] \arrow[dr] & & & \mathrm{Decode}(y_0, y_1) = P(x) \\
 & x_1 \arrow[r] & y_1 = \mathrm{Eval}(P, x_1) \arrow[ur] &
\end{tikzcd}$$

<u>Properties (informal)</u>

- <u>correctness</u>:
- <u>security</u>: for $b \in \{0,1\}$ and any $x, \hat{x}$ (s.t. $|x| = |\hat{x}|$)

  $s_b \approx_c \hat{s}_b$ where $s_0, s_1 \leftarrow \mathrm{Share}(x)$
  $\hat{s}_0, \hat{s}_1 \leftarrow \mathrm{Share}(\hat{x})$

<u>Extra properties</u>:

compactness: output of Eval to depend on output length of P
(not the size of P)

additive: output shares are additive

<u>Restricted Multiplication Straight-line program</u> (RMS programs)

- <u>load</u> an input into memory
- <u>add</u> two memory locations
- <u>multiply</u> an <u>input</u> with a <u>memory</u> location
- <u>output</u> a memory location

branching, NC^1

(informal theorem) assuming DDH,
can build HSS* for RMS programs

---

group : set w/ an operation + inverses

interested cyclic groups, generator g

DDH : $|q| = 2^n$, $\mathbb{G}$ of order q, gen g

$$(\mathbb{G}, g, q, g^a, g^b, g^{ab}) \approx_c (\_\_\_, g^a, g^b, T)$$

$$a, b \leftarrow \mathbb{Z}_q$$
$$T \leftarrow \mathbb{G}$$

Warm up construction inputs : x, y

Encoding $(x) = g^x$

$x = x_0 - x_1$
$y = y_0 - y_1$

(A)	(B)
x_0, g^x	x_1, g^x
y_0, g^y	y_1, g^y

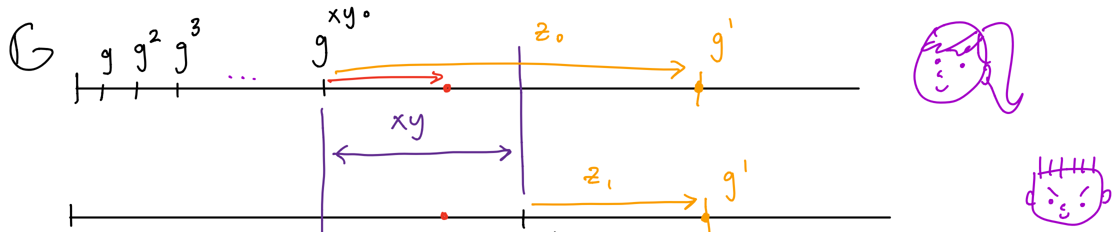
Add: $x + y = (x_0 + y_0) - (x_1 + y_1)$
$g^{x+y} = g^x / g^y$

Mult: want xy

$(g^x)^{y_0}$ $(g^x)^{y_1}$

$$g^{xy} = (g^x)^{y_0} / (g^x)^{y_1}$$

How to convert mult shares in exp
to additive shares?



some bound

$z_0 - z_1 = xy$ $xy < M$

$(g^{xy_0}) \cdot g^i$

G' "δ - sparse set" : $g' \in \mathbb{G}$ has δ - prob

parameter

of appearing in $\mathbb{G}'$

implement $\mathbb{G}'$ w/ PRF $\phi : \mathbb{G} \to \{0,1\}^{\log \delta}$

g' in $\mathbb{G}'$ if $\phi(g') = 0 \ldots 0$

$(g^x)^{y_0}$

$\phi\left((g^x)^{y_0} \cdot g^i\right) \overset{?}{=} 0^{\log \delta}$

if not $i{+}{+}$

if yes output i

ElGamal encryption:

$sk = c \qquad c \leftarrow \mathbb{Z}_q$
$pk = (\mathbb{G}, g, g^c)$

$Enc_{pk}(m) = \quad r \leftarrow \mathbb{Z}_q$
$\quad ct = (g^r, g^m \cdot g^{cr})$

$Dec_{sk}((ct_0, ct_1)) = m \quad$ s.t. $g^m = \dfrac{ct_1}{ct_0^c}$

assume $0 \leq m \leq poly(\lambda)$

ElGamal ct $[\![w]\!]_c$

additive $\langle y \rangle$

"The Whole Enchillada" protocol:

Share $(1^\lambda, w_1, \ldots, w_n)$

sharing of input w_i: $[\![w_i]\!]_c$ — need circular security
$\{[\![c^{(t)} w_i]\!]_c\}_{t \in [\ell]}$ } same for both parties "input" vals
$\langle w_i \rangle$
$\langle c w_i \rangle$ } individual shares "memory" vals

Eval:

Add mem vals:
$\langle w_i \rangle + \langle w_j \rangle = \langle w_i + w_j \rangle$
$\langle c w_i \rangle + \langle c w_j \rangle = \langle c(w_i + w_j) \rangle$

Multiply mem and input

$[\![x]\!]_c, \{[\![c^{(t)} x]\!]_c\}_{t \in [\ell]} \qquad \langle y \rangle, \langle c y \rangle$

$(g^r, g^x g^{rc})$

get mult share (in exponent) of xy as:

$$(g^x g^{rc})^{\langle y \rangle} (g^r)^{-\langle cy \rangle} = g^{x\langle y \rangle}$$

$\langle xy \rangle$

$\langle c^{(t)} xy \rangle$ $\qquad$ $c = \sum 2^i c^{(i)}$

$\langle c\, xy \rangle$

---

public key (aka parties can create shares themselves)

$[\![1]\!]_c, \{[\![c^{(t)}]\!]_c\}_{t \in [\ell]}$ ← added to pk

$\langle 1 \rangle, \langle c \rangle$

party wants to input w

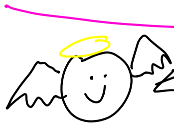
do other things not included here

$[\![w]\!]_c, [\![c^t w]\!]_c$

$[\![1]\!]_c = (g^r, g \cdot g^{rc})$

$[\![w]\!]_c = (g^{wr}, g^w \cdot g^{wrc}) = (g^{r'}, g^w g^{r'c})$

$[\![c^t]\!]_c^w$



makes keys

doing comp

$\langle x \rangle, \langle cx \rangle$: mult $[\![x]\!], [\![cx]\!]$

$\langle 1 \rangle, \langle c \rangle$