Back to FHE:

Sufficient to choose X such that $s_1^T X \approx (b_1^T - b_2^T) R$

Hint consists of

$$V_{ij} \leftarrow A_1 R_{ij}' + R_{ij} \cdot G \quad \text{where } R_{ij}' \xleftarrow{R} \{0,1\}^{m \times m}$$

(encryption of R_{ij} under A_1)

During evaluation time (when b_1^T and b_2^T are known), use $\{V_{ij}\}_{i,j \in [m]}$ and $b_1^T - b_2^T$ in above procedure to obtain $X \in \mathbb{Z}_q^{n \times m}$ where $s_1^T X = (b_1^T - b_2^T) R$

Then, the expanded ciphertext is

$$\hat{C} = \begin{bmatrix} C & X \\ 0 & C \end{bmatrix} \Rightarrow s^T \hat{C} = [s_1^T \mid s_2^T] \begin{bmatrix} C & X \\ 0 & C \end{bmatrix} \approx [x \cdot s_1^T G \mid s_1^T X + s_2^T C]$$

$$= x \cdot [s_1^T \mid s_2^T] \cdot G$$

which is a GSW ciphertext with respect to $[s_1^T \mid s_2^T]$

If C is an encryption under pk_2, then expanded ciphertext will be

$$\hat{C} = \begin{bmatrix} C & 0 \\ X & C \end{bmatrix} \Rightarrow s^T \hat{C} = [s_1^T C + s_2^T X \mid s_2^T C]$$

X is chosen so upon $s_2^T X = (b_2^T - b_1^T) \cdot R$ (by including encryptions of bits of R)

More generally, with N public keys $pk_1, \dots, pk_N$, expanded ciphertext encrypted under pk_i has the form

$$\begin{bmatrix} C & & & & & & \\ & \ddots & & & & & \\ X_1 & \cdots & X_{i-1} & C & X_{i+1} & \cdots & X_N \\ & & & & \ddots & & \\ & & & & & & C \end{bmatrix} \leftarrow \text{row } i$$

$$\therefore [s_1^T \mid \cdots \mid s_N^T] \hat{C} = [s_1^T C + s_i^T X_1 \mid \cdots \mid s_{i-1}^T C + s_i^T X_i \mid s_i^T C \mid s_{i+1}^T C + s_i^T X_{i+1} \mid \cdots \mid s_N^T C + s_N^T X_N]$$
$$\approx x \cdot [s_1^T \mid \cdots \mid s_N^T] \cdot G$$

Application to MPC (in CRS model):

1) Every party generates a public key pk_i and encrypts x_i using pk_i. It broadcasts ct_i to all parties
2) Every party homomorphically computes to get encryption of $f(x_1, \dots, x_N)$
3) Parties decrypt the final ciphertext(s)
   ↳ Requires combining the secret keys — would leak inputs if done naively!

Observation: Let $\hat{C} \in \mathbb{Z}_q^{Nn \times Nm}$ be final ciphertext. Goal is to compute

$$[s_1^T | \cdots | s_N^T] \cdot \hat{C} \quad \text{where party } i \text{ knows } s_i \text{ and } \hat{C}$$

$$\hookrightarrow [s_1^T | \cdots | s_N^T] \cdot \begin{bmatrix} \hat{C}_1 \\ \hline \vdots \\ \hline \hat{C}_N \end{bmatrix} = s_1^T \hat{C}_1 + \cdots + s_N^T \hat{C}_N$$

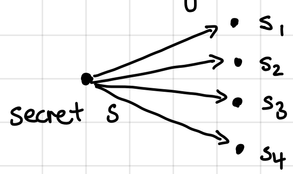
each party computes one term $s_i^T \hat{C}_i$

$\hat{C}_i \in \mathbb{Z}_q^{n \times Nm}$ (ith block of $\hat{C}$)

To decrypt in the MPC setting, each party simply decrypts "locally" and publishes their "share" of the output.
To reconstruct output, simply sum all of the shares together.

To prove simulation security for MPC protocol, parties add additional "smudging" noise to prevent partial decryption from leaking information.

Connection to secret sharing: In secret sharing scheme



"shares" of s

t-out-of-n secret sharing: any subset of t shares can be used to reconstruct the secret s

Security: Any subset with fewer than t-shares reveals no information about the secret s.
Namely, there exists a simulator S such that for all sets $T \subseteq [n]$ and all messages s:

$$\{(s_i)_{i \in T} : (s_1, \ldots, s_n) \leftarrow \text{Share}(1^\lambda, 1^n, s)\} \overset{s}{\approx} \{S(1^\lambda, 1^n, |s|, T)\}$$

Constructing n-out-of-n secret sharing:

Share $(1^\lambda, 1^n, s)$: To share a message $s \in \{0,1\}^\ell$, sample $s_1, \ldots, s_{n-1} \xleftarrow{R} \{0,1\}^{\ell-1}$ and set $s_n \leftarrow s_1 \oplus \cdots \oplus s_{n-1} \oplus s$

Reconstruct $(s_1, \ldots, s_n)$: Output $s_1 \oplus \cdots \oplus s_n$

Security is just one-time pad security.

will require that $p > n$

Constructing t-out-of-n secret sharing (Shamir secret sharing)

Share$(1^\lambda, 1^n, t, s)$: To share a message $s \in \mathbb{Z}_p$ (p is prime so $\mathbb{Z}_p$ is a field), sample a random polynomial $f \in \mathbb{Z}_p[x]$ of degree $t-1$ where $f(0) = s$. In other words, sample $f_1, \ldots, f_{t-1} \xleftarrow{R} \mathbb{Z}_p$ and let

$$f(x) = s + f_1 x + \cdots + f_{t-1} x^{t-1}$$

Output shares $s_i \leftarrow (i, f(i))$ for $i \in [n]$

Reconstruct$(\{s_i\}_{i \in T})$: Given at least t shares (i, z_i) for $i \in T$, interpolate the unique polynomial of degree $t-1$ such that $f(i) = z_i$. Output $f(0)$.

Security: Follows from the fact that it takes t points to define a polynomial of degree $t-1$.

When all of the shares provided to the Reconstruct algorithm are valid, then reconstruction is just polynomial interpolation (can also view as Reed-Solomon decoding – as an erasure code).

Homomorphic secret sharing (with additive reconstruction)

$$x \to x_1 \xrightarrow{f} t_1, \quad x \to x_2 \xrightarrow{f} t_2, \quad x \to x_3 \xrightarrow{f} t_3$$

Non-interactive evaluation procedure!

<u>Guarantee</u>: if x_1, x_2, x_3 are secret shares of x (i.e., hide x), then $t_1 + t_2 + t_3 = f(x)$

We will see some useful applications of this primitive soon.

Multi-key FHE $\Rightarrow$ 2-party HSS

$x \in \{0,1\}^\ell \longrightarrow$ Sample key-pairs (pk_1, sk_1) and (pk_2, sk_2)
Sample $x_1, x_2 \xleftarrow{R} \{0,1\}^\ell$ where $x_1 \oplus x_2 = x$
Compute $ct_1 \leftarrow \text{Encrypt}(pk_1, x_1)$
$ct_2 \leftarrow \text{Encrypt}(pk_2, x_2)$

$pk_1, pk_2, ct_1, ct_2, sk_1$
$pk_1, pk_2, ct_1, ct_2, sk_2$
} shares

Homomorphic computation on shares $\equiv$ homomorphic evaluation
$\hookrightarrow$ Remaining question: obtain <u>additive</u> secret shares of $f(x)$

<u>Currently</u>: after homomorphic computation: $[s_1^T \mid s_2^T] \cdot C \approx f(x) \cdot [s_1^T \mid s_2^T] \cdot G$

if $C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$, then $\underbrace{s_1^T C_1 + s_2^T C_2}_{\text{almost a secret share (but of a vector)}} = f(x) \cdot [s_1^T \mid s_2^T] \cdot G$

to obtain a scalar, observe that last component of secret key is 1. Let $w = [0, 0, \ldots, 0, q/2]^T$.

Then $[s_1^T \mid s_2^T] \, C G^{-1}(w) \approx f(x) \cdot [s_1^T \mid s_2^T] \cdot G \cdot G^{-1}(w) = \frac{q}{2} \cdot f(x)$

$\Rightarrow s_1^T C_1 G^{-1}(w) + s_2^T C_2 G^{-1}(w) = \frac{q}{2} f(x) + e$ for some small error e

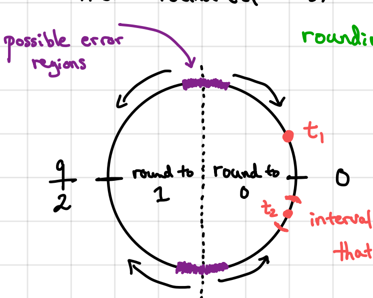
$\downarrow$

recover $f(x)$ by rounding (check if value closer to 0 or $\frac{q}{2}$)

Reconstruction is still not linear...

<u>Observe</u>: Suppose $t_1 + t_2 = \frac{q}{2} \cdot f(x) + e \pmod q$ where t_1, t_2 uniform over $\mathbb{Z}_q$ and e is small
Then $\text{round}(t_1 + t_2) = \text{round}(t_1) + \text{round}(t_2)$ with high probability $(= 1 - O(|e|)/q)$

rounding and addition commute with high probability (for the case of <u>two</u> shares)



interval of size $2|e|+1$ that contains t_2 (so $t_1 + t_2 = e$)

Suppose $f(x) = 0$. Then
$\text{round}(t_1) + \text{round}(t_2) = 0$ if t_1 and t_2 are on the same side as rounding boundary

error occurs only if t_1 and t_2 land on different sides of rounding boundary: prob. $\frac{2|e|+1}{q}$

(technically, this is smaller since only half of each interval can contribute to a rounding error — based on the sign of e)

Thus if $s_1^T C_1 G^{-1}(w)$ and $s_2^T C_2 G^{-1}(w)$ are individually uniform over $\mathbb{Z}_q$ and q sufficiently large $(q \sim (n \log q)^{O(d)})$

$$\begin{aligned}
\text{round}\left(s_1^T C_1 G^{-1}(w) + s_2^T C_2 G^{-1}(w)\right) &= \text{round}\left(s_1^T C_1 G^{-1}(w)\right) + \text{round}\left(s_2^T C_2 G^{-1}(w)\right) \\
\| \\
\text{round}\left(\tfrac{q}{2}\cdot f(x) + e\right) \\
\| \\
f(x)
\end{aligned}$$

But... $s_1^T C_1 G^{-1}(w)$ may not be uniform. So cannot apply above analysis.

<u>Solution</u>: Re-randomize using a secret share of 0.

Namely, sample $r \xleftarrow{R} \mathbb{Z}_q$ and give r to 1 party and $-r$ to the other:

$$\left.\begin{aligned} P_1 &\text{ now computes } s_1^T C_1 G^{-1}(w) + r \\ P_2 &\text{ now computes } s_2^T C_2 G^{-1}(w) - r \end{aligned}\right\} \text{still a secret share of } \tfrac{q}{2} f(x) + e$$

r is independent of s_1, s_2, C

2-party HSS from multi-key FHE:

Share $(1^\lambda, \underset{\in \{0,1\}^\ell}{x})$: Sample $crs \leftarrow \text{Setup}(1^\lambda)$

$(pk_i, sk_i) \leftarrow \text{KeyGen}(crs)$ for $i \in \{1,2\}$

$x_1 \xleftarrow{R} \{0,1\}^\ell$, $x_2 \leftarrow x \oplus x_1$

$ct_i \leftarrow \text{Encrypt}(pk_i, x_i)$

$\delta_1 \xleftarrow{R} \mathbb{Z}_q$, $\delta_2 \leftarrow -\delta_1$

Output shares

$$z_1 = (pk_1, pk_2, ct_1, ct_2, sk_1, \delta_1)$$
$$z_2 = (pk_1, pk_2, ct_1, ct_2, sk_2, \delta_2)$$

Eval (f, z_i): Define the bivariate function $g(x_1, x_2) := f(x_1 \oplus x_2)$

Homomorphically evaluate $C \leftarrow \text{Eval}(pk_1, pk_2, ct_1, ct_2, g)$

Let $sk_i = s_i$ and let C_i be i^{th} block of C.

Output $\text{round}(s_i^T C_i G^{-1}(w) + \delta_i)$

By above guarantee:
$$\begin{aligned}
\text{round}(s_1^T C_1 G^{-1}(w) + \delta_1) + \text{round}(s_2^T C_2 G^{-1}(w) + \delta_2) &= \text{round}\left(s_1^T C_1 G^{-1}(w) + s_2^T C_2 G^{-1}(w)\right) \\
&= \text{round}\left(\tfrac{q}{2}\cdot f(x) + \text{error}\right) \\
&= f(x)
\end{aligned}$$

Can extend from 2-party HSS to n-party HSS generically by relying on additive homomorphism