Multi-party Homomorphic Secret Sharing & Sub-linear MPC from Sparse LPN

Quang Dao   Aayush Jain   Yuval Ishai   Huijia Lin

Carnegie Mellon University

TECHNION Israel Institute of Technology

UNIVERSITY of WASHINGTON

Crypto 2023

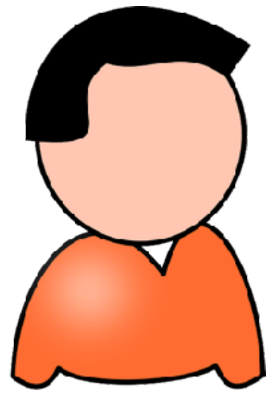
Homomorphic Secret Sharing (HSS)

Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)

Homomorphic Secret Sharing (HSS)

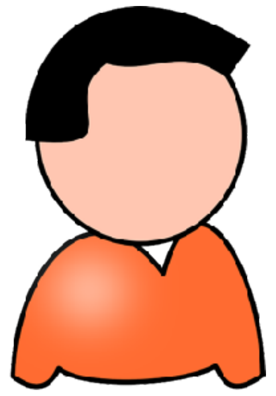
(distributed / secret-shared version of homomorphic encryption)



Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)

Share *Eval* *Rec*

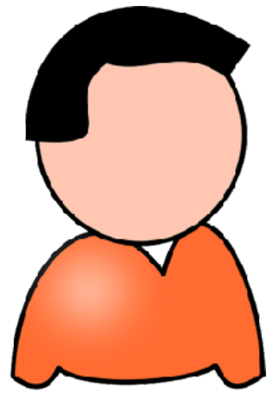


Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)

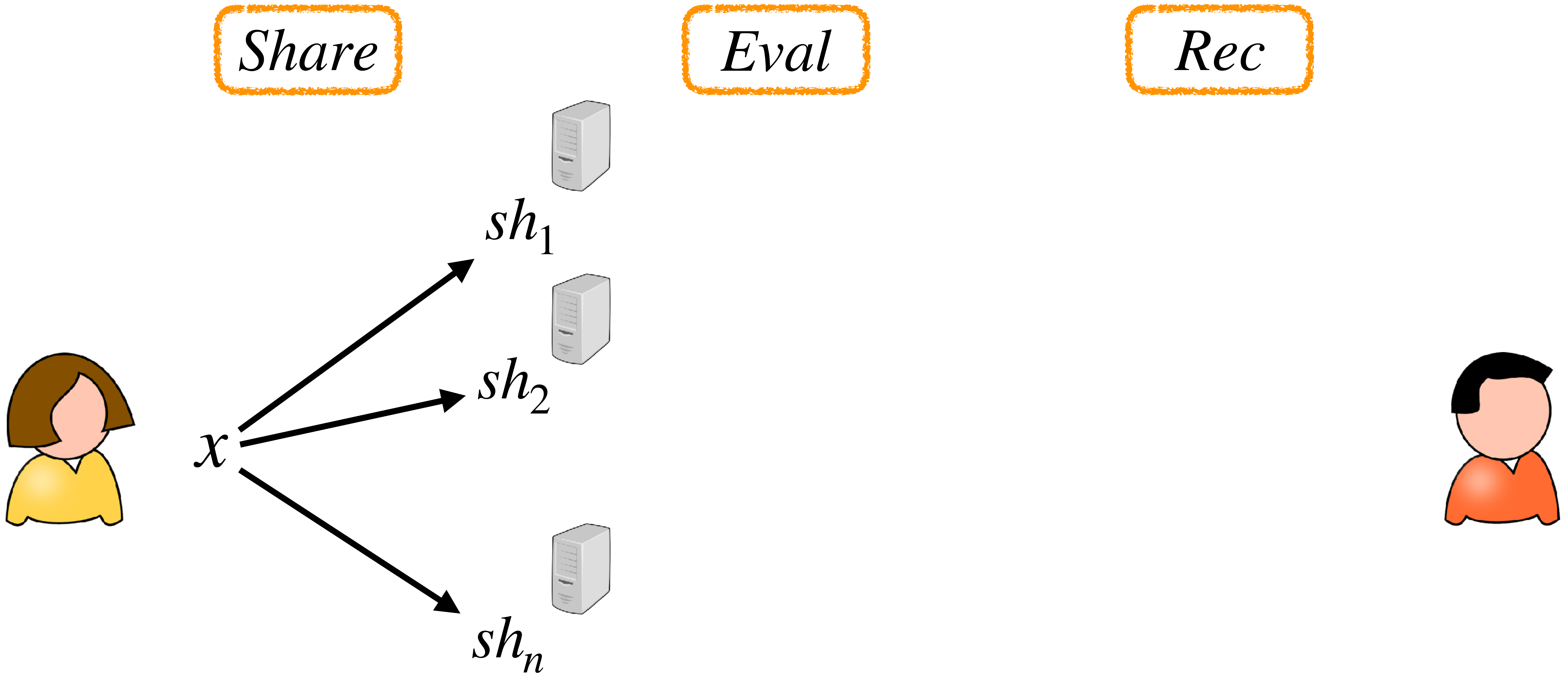
$\boxed{Share}$ $\boxed{Eval}$ $\boxed{Rec}$

x



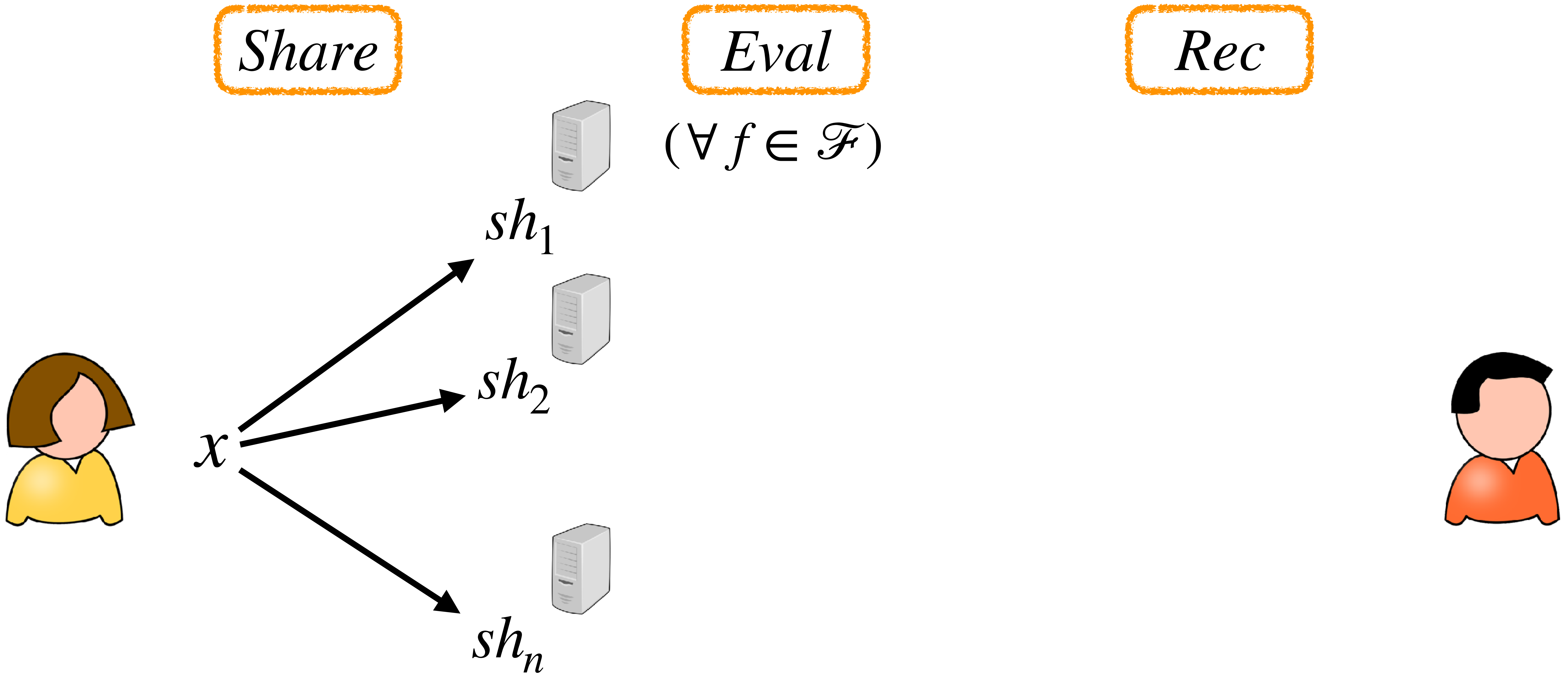
Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)



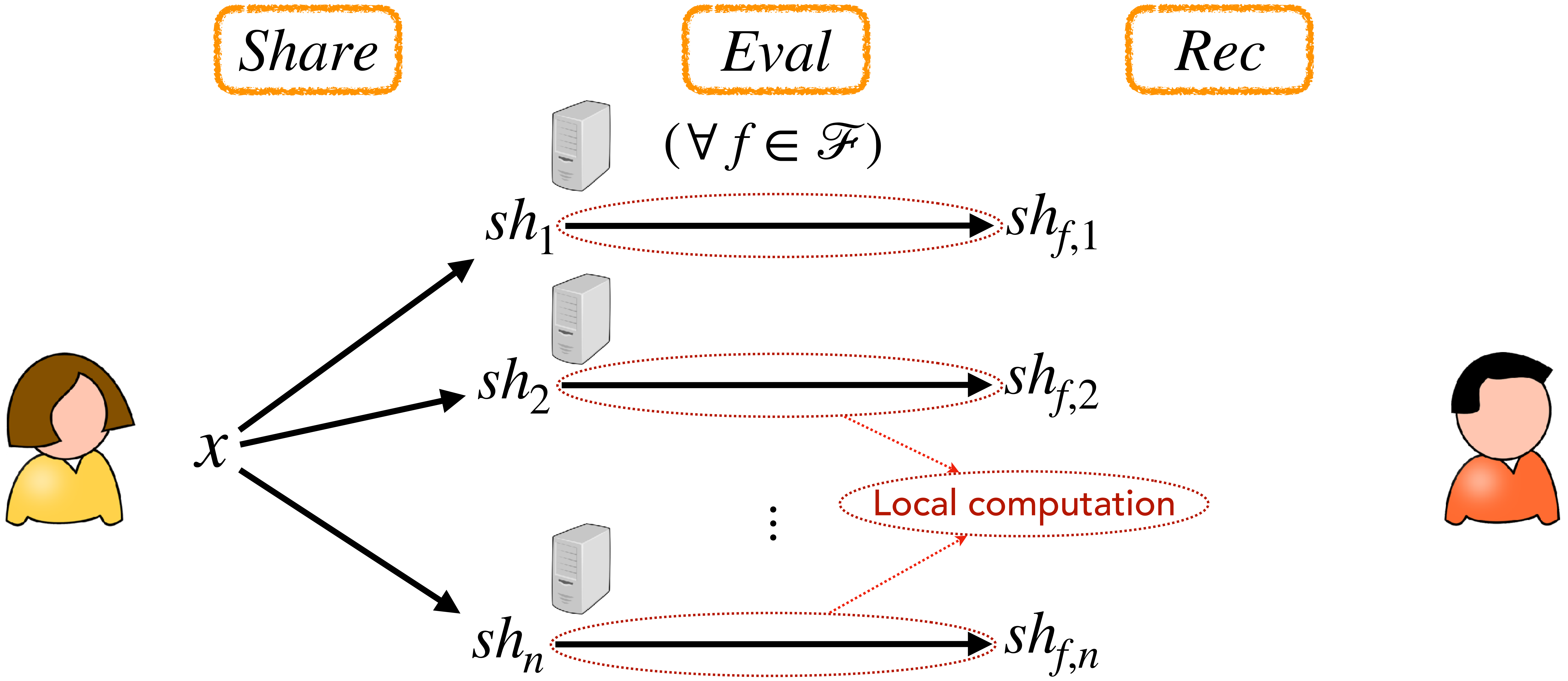
Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)



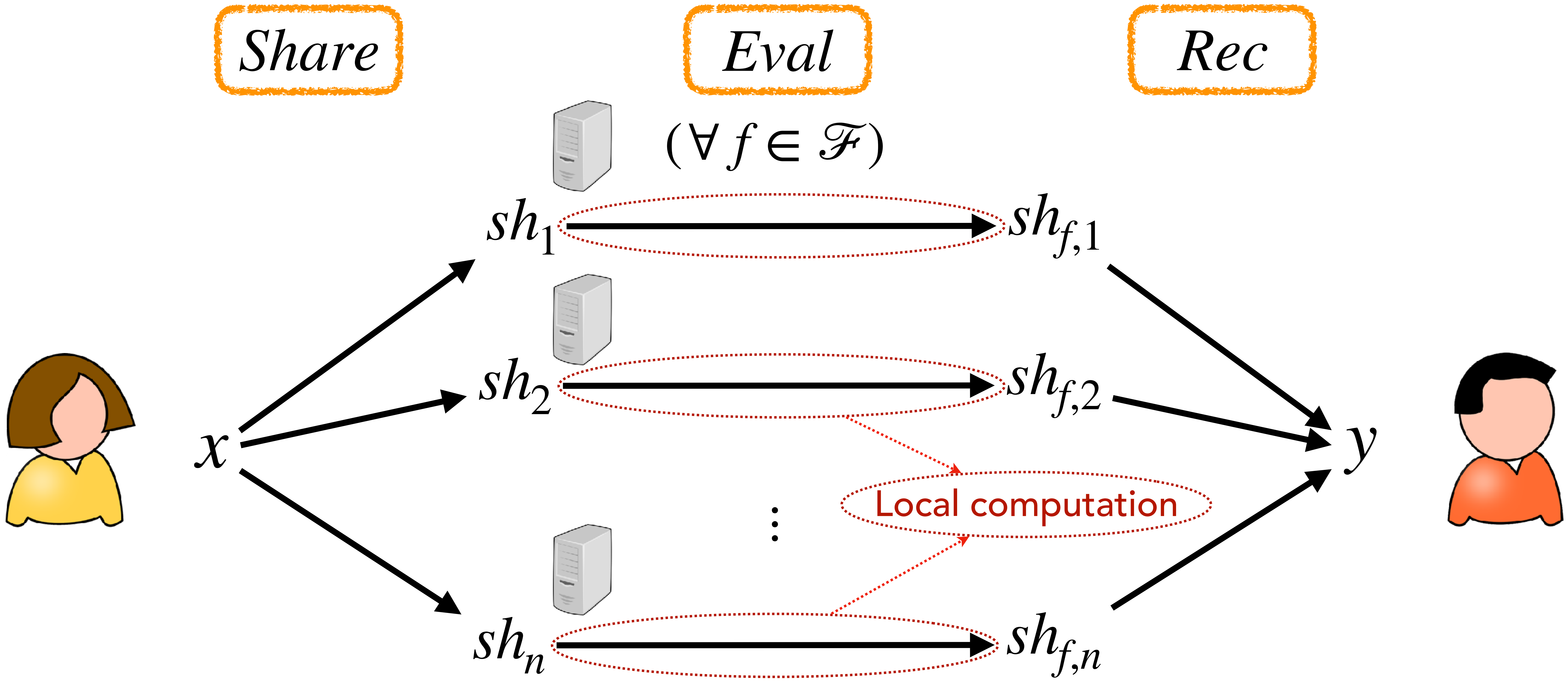
Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)



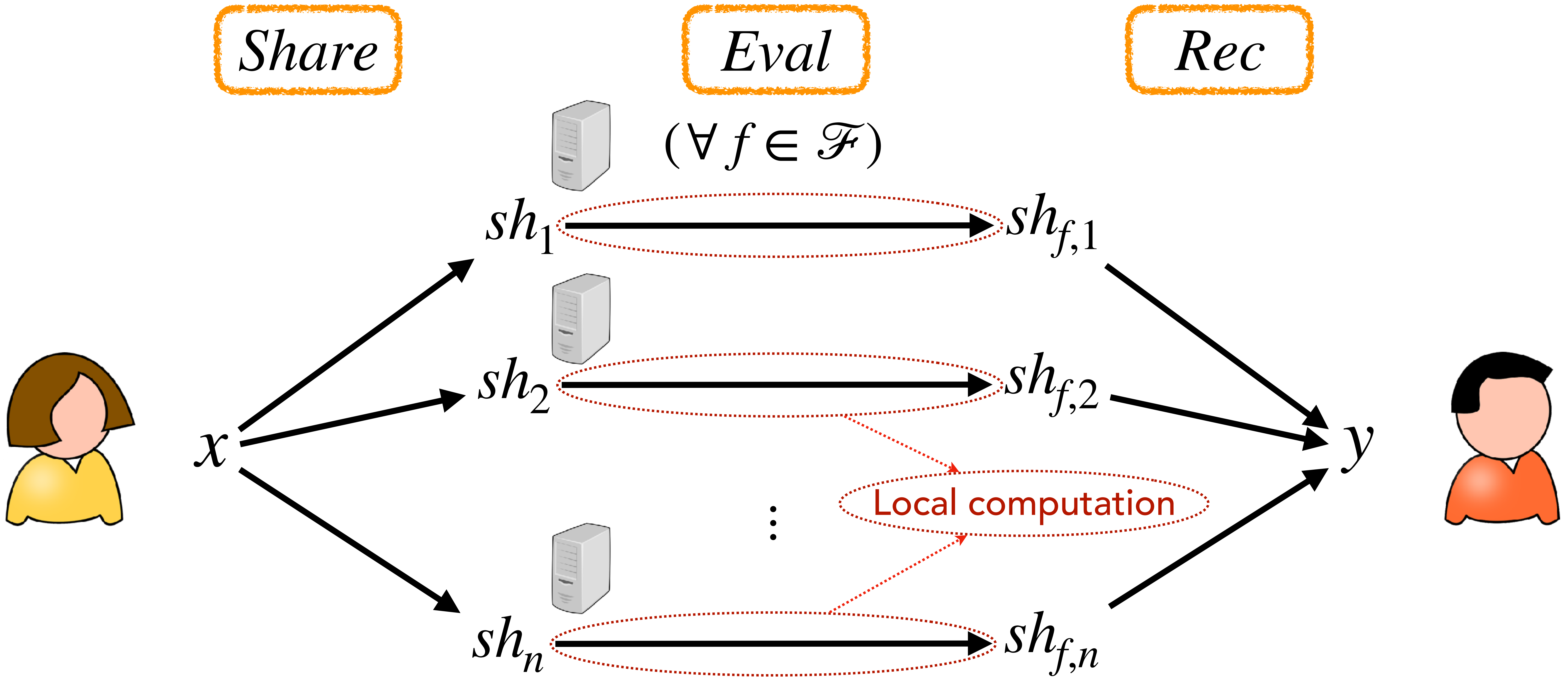
Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)



Homomorphic Secret Sharing (HSS)

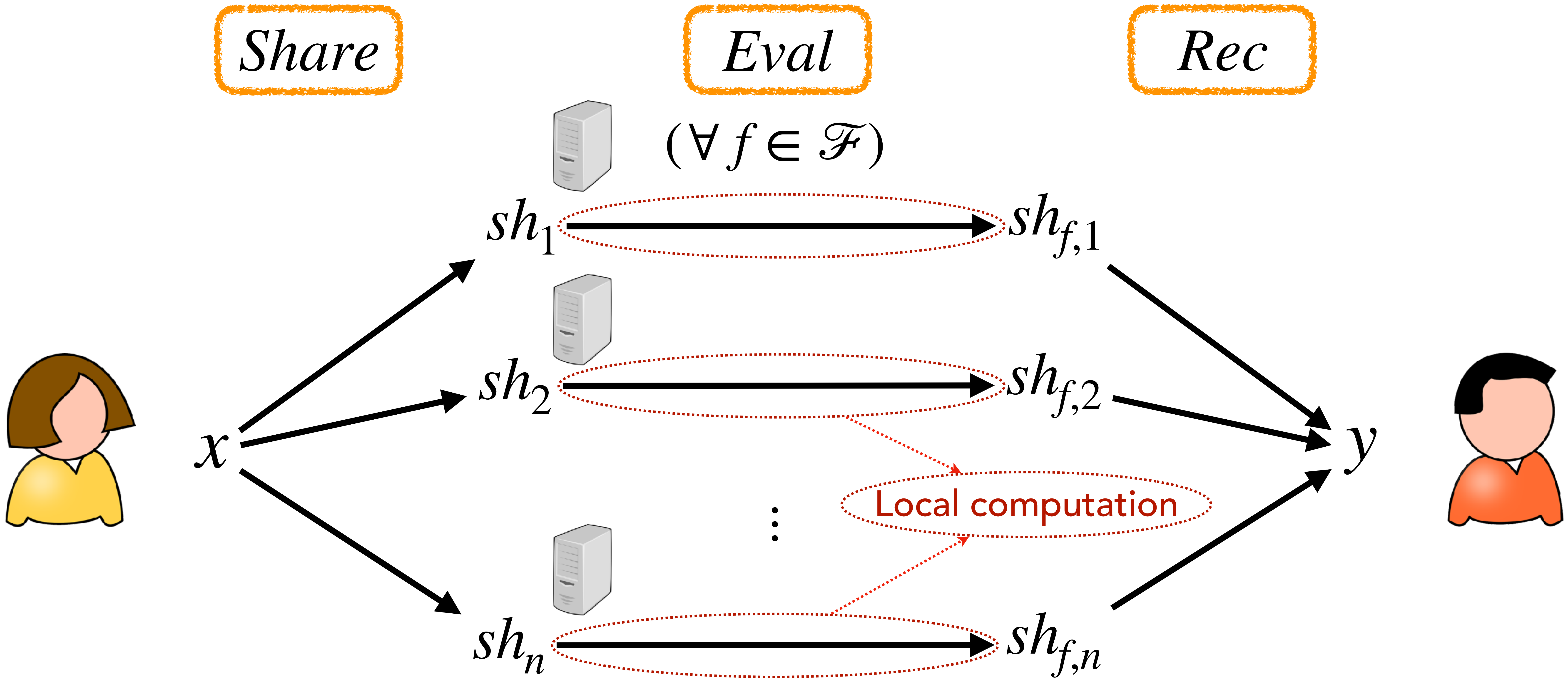
(distributed / secret-shared version of homomorphic encryption)



- δ-**Correctness:** $Pr\left[y = f(x)\right] \geq 1 - \delta$.
- t-**Privacy:** any $\leq t$ shares hide x.

Homomorphic Secret Sharing (HSS)

(distributed / secret-shared version of homomorphic encryption)



- **δ-Correctness:** $Pr\left[y = f(x)\right] \geq 1 - \delta.$
- **t-Privacy:** any $\leq t$ shares hide x.
- **Compactness:** $|\mathsf{sh}_{f,i}| \ll |f|$
- **Linear reconstruction:** *(Default)*
  Rec is a linear function

HSS Application: Sublinear MPC

(secure multi-party computation)

HSS Application: Sublinear MPC

(secure multi-party computation)

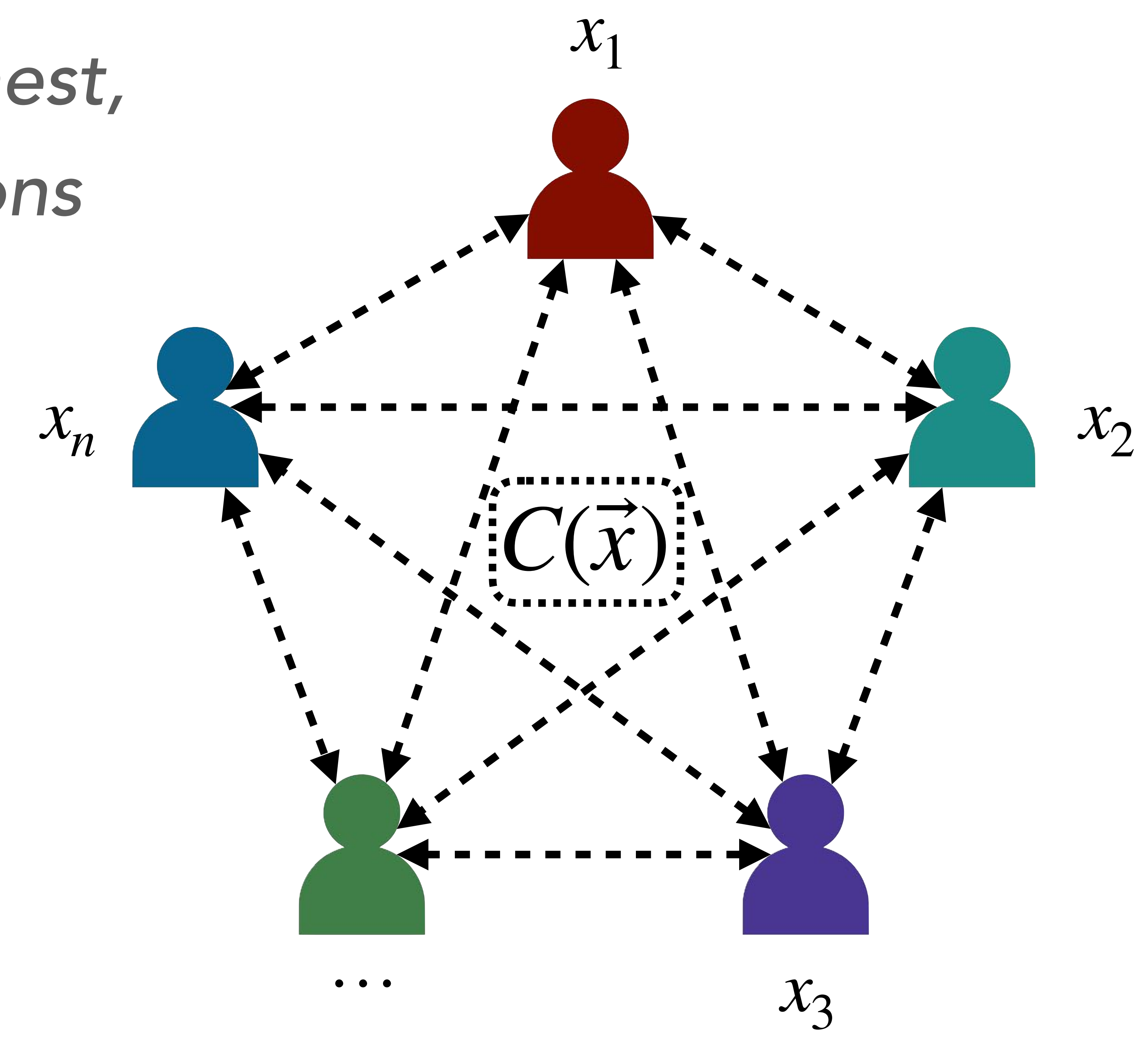
Motivation: "circuit size barrier" in MPC

HSS Application: Sublinear MPC

(secure multi-party computation)

Motivation: "circuit size barrier" in MPC

Setting: *semi-honest,* $(n-1)$ *corruptions*

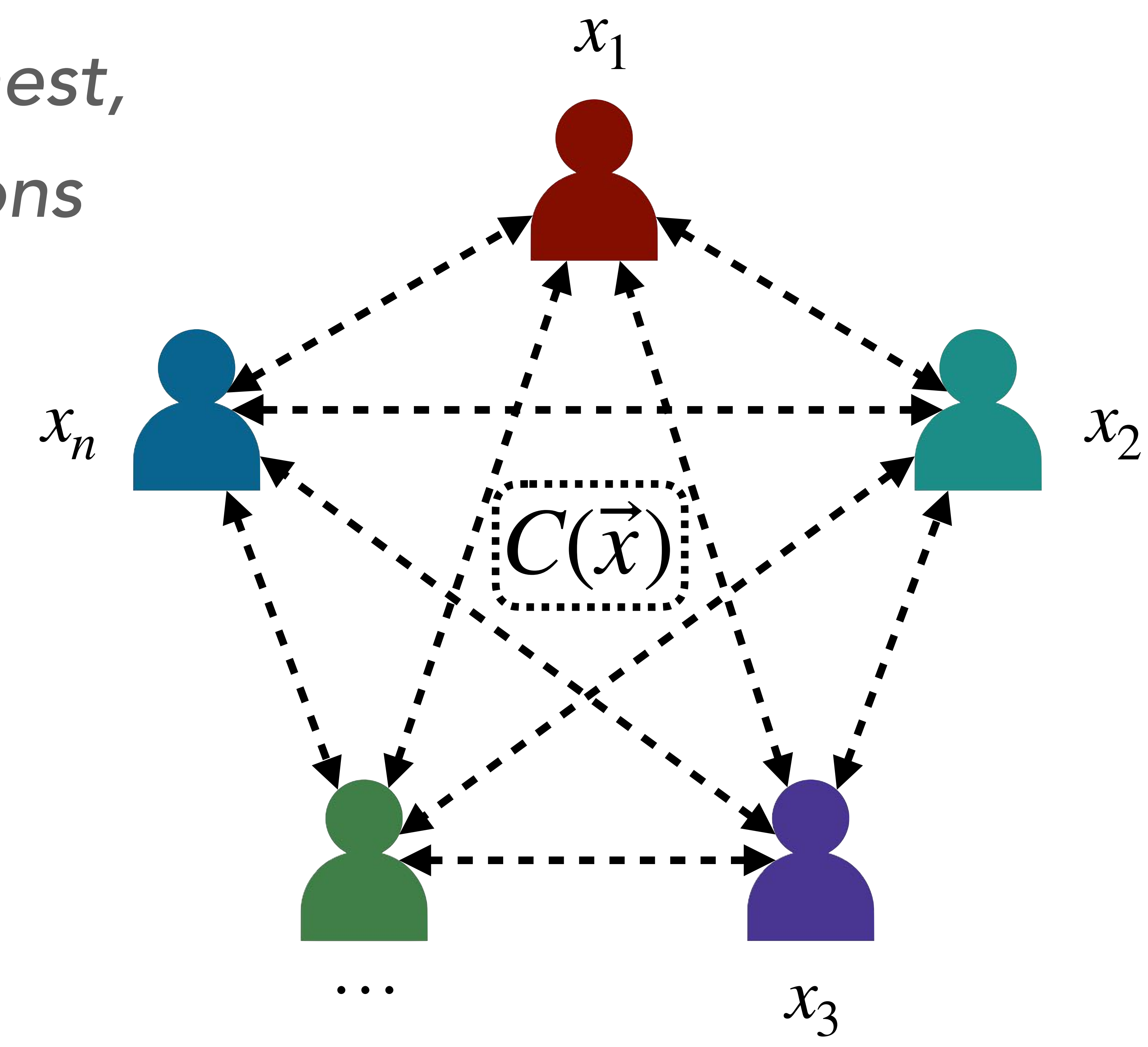


HSS Application: Sublinear MPC

(secure multi-party computation)

Motivation: "circuit size barrier" in MPC

Setting: *semi-honest,*
$(n-1)$ *corruptions*



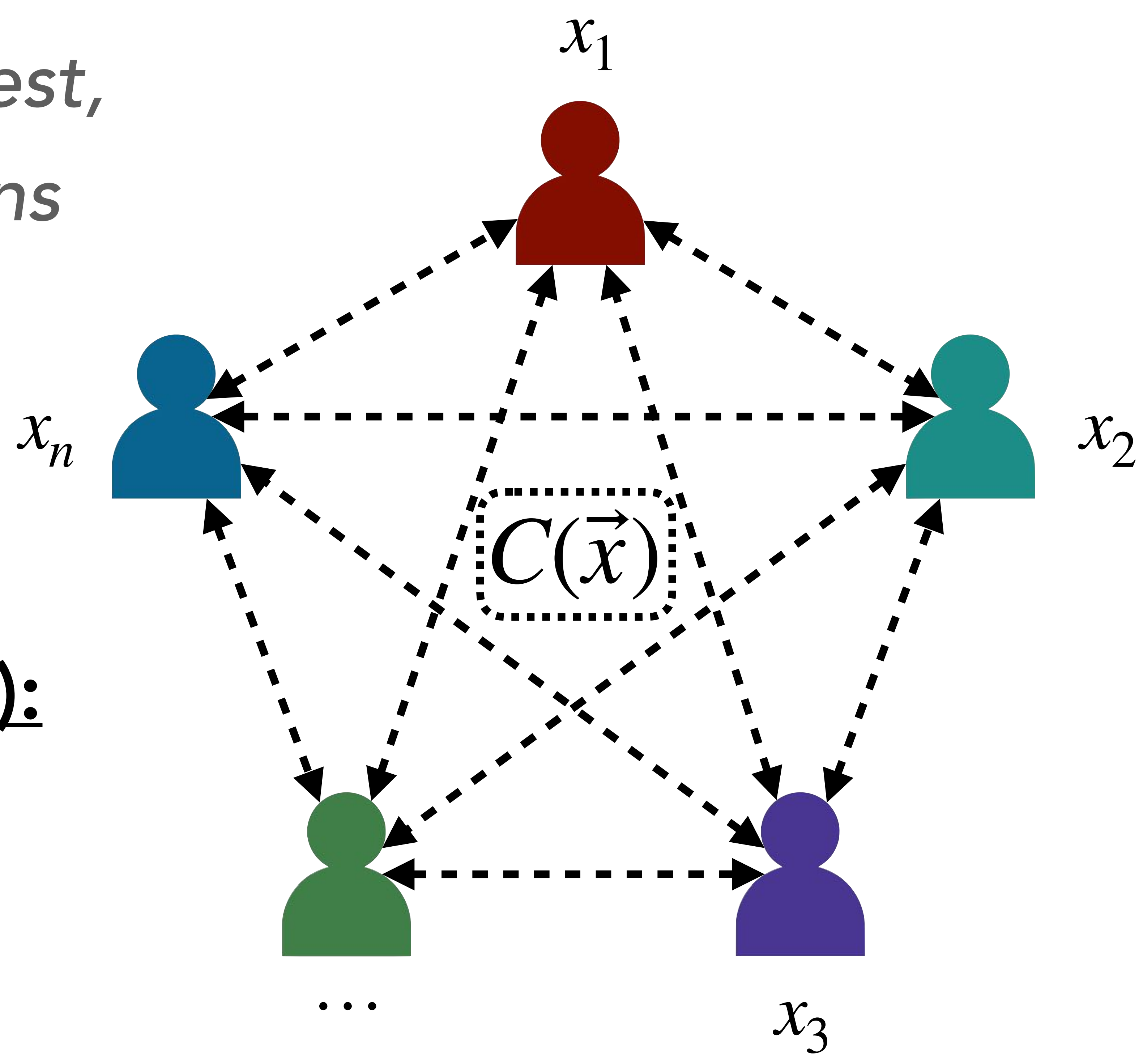
For "classical" protocols [GMW87, BGW88, BMR90]
and their extensions

HSS Application: Sublinear MPC

(secure multi-party computation)

Motivation: "circuit size barrier" in MPC

Setting: *semi-honest,*
$(n-1)$ *corruptions*



Comm (per-party):
$\Omega(|C|)$

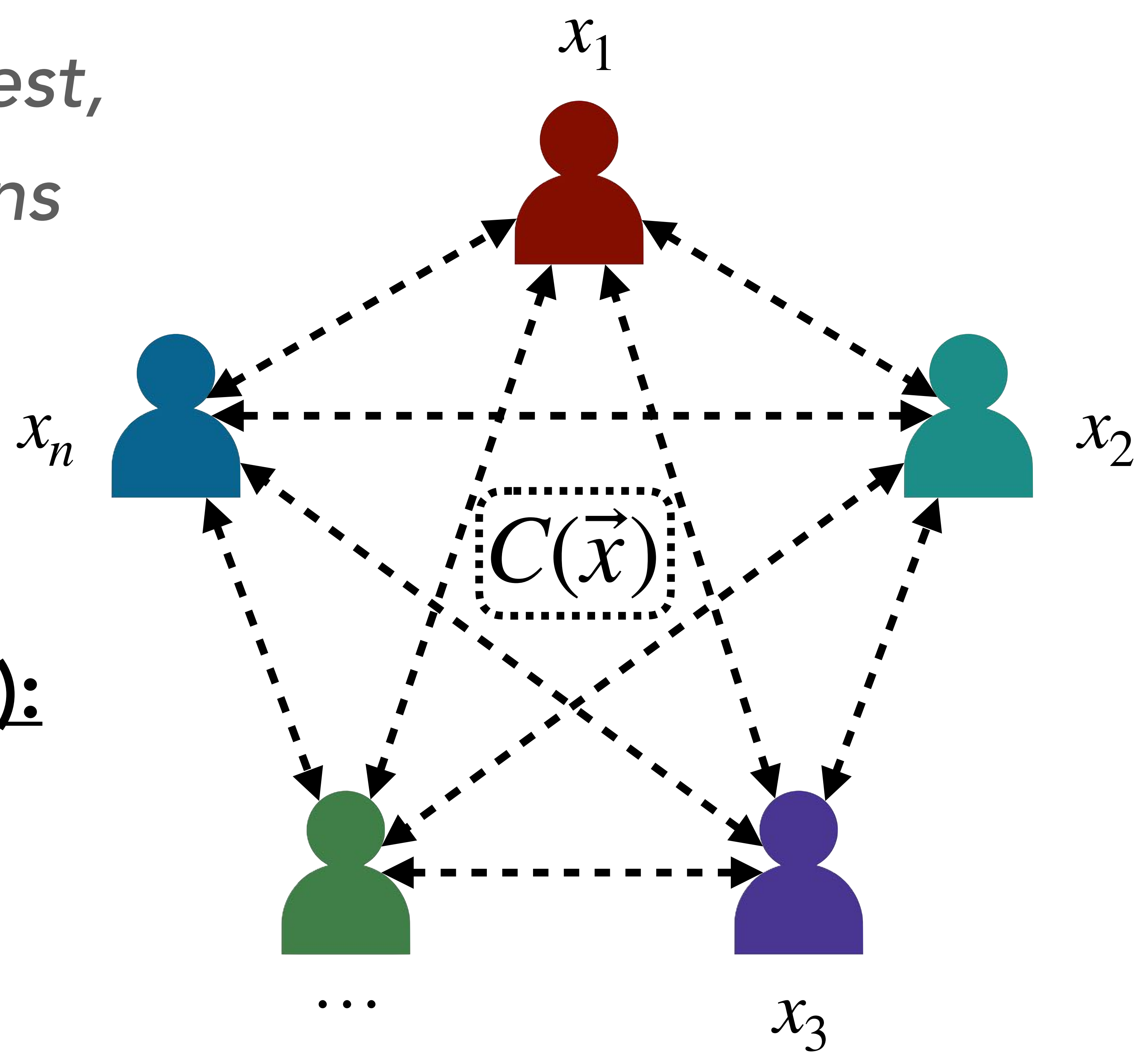
For "classical" protocols [GMW87, BGW88, BMR90]
and their extensions

HSS Application: Sublinear MPC

(secure multi-party computation)

Motivation: "circuit size barrier" in MPC

Setting: *semi-honest,* $(n-1)$ *corruptions*



Comm (per-party):

$\Omega(|C|)$

For "classical" protocols [GMW87, BGW88, BMR90] and their extensions

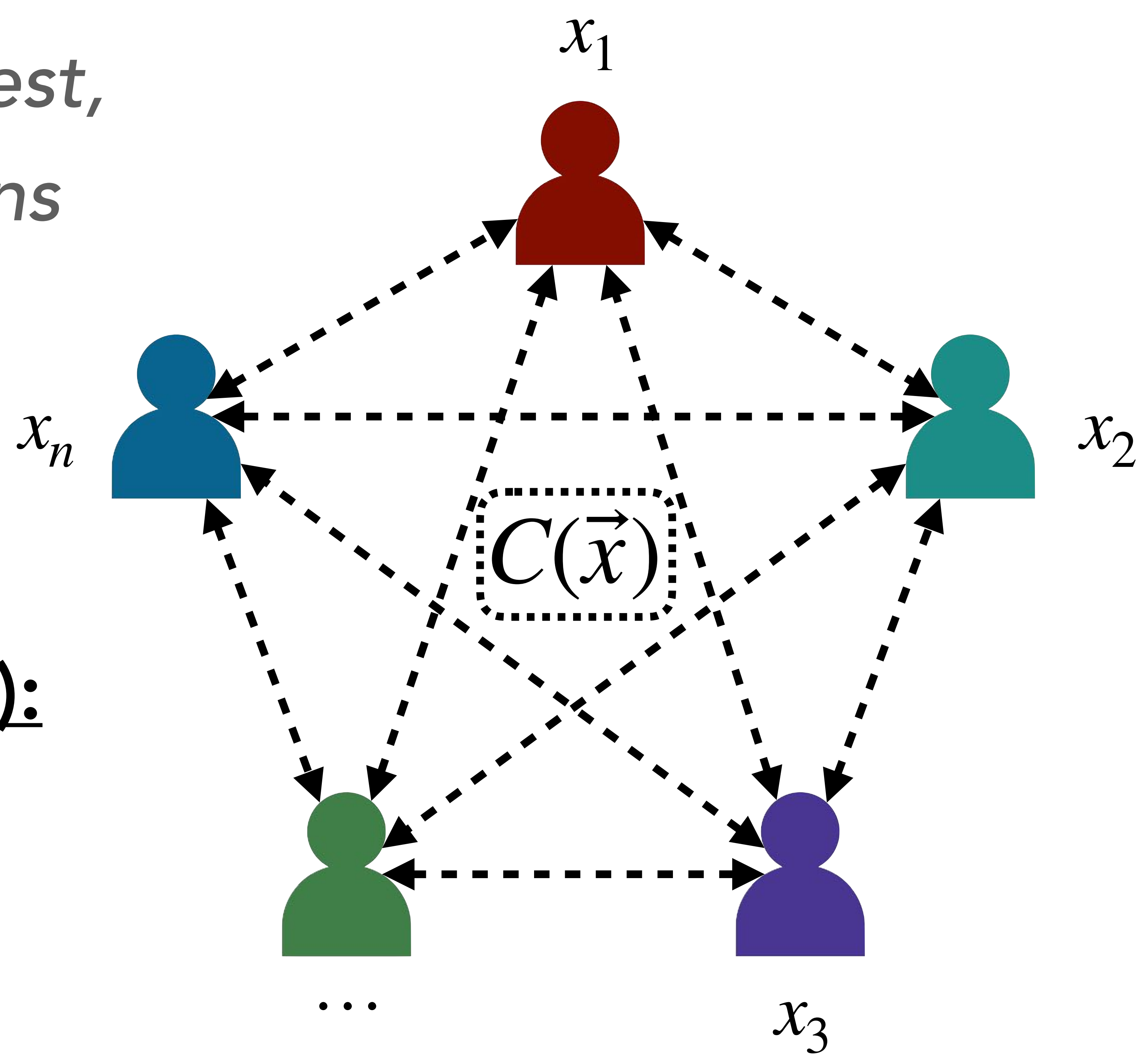
Solution: HSS-based MPC

HSS Application: Sublinear MPC

(secure multi-party computation)

Motivation: "circuit size barrier" in MPC

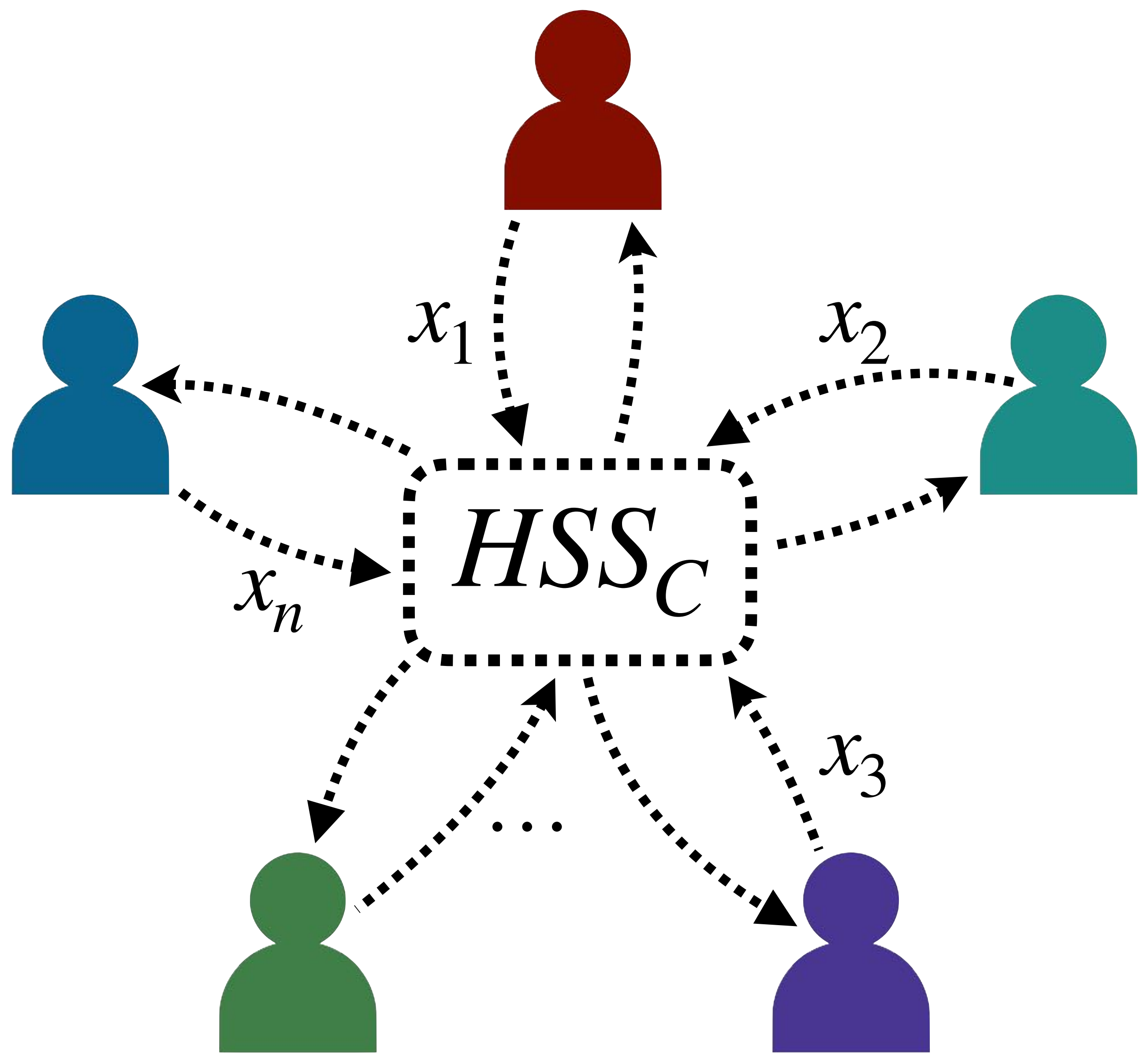
Setting: *semi-honest,*
$(n-1)$ *corruptions*



Comm (per-party):
$\Omega(|C|)$

For "classical" protocols [GMW87, BGW88, BMR90]
and their extensions

Solution: HSS-based MPC

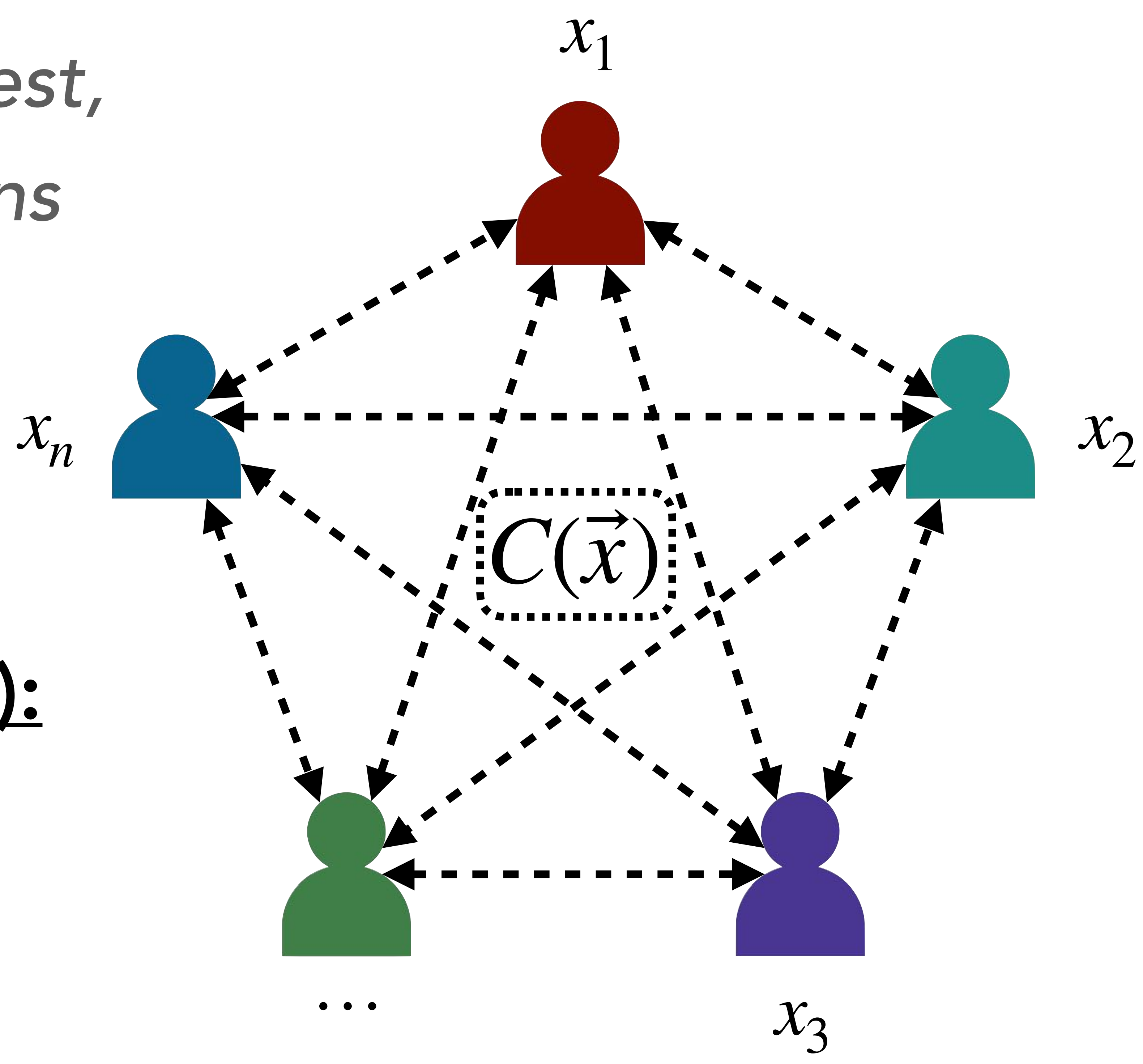


HSS Application: Sublinear MPC

(secure multi-party computation)

Motivation: "circuit size barrier" in MPC

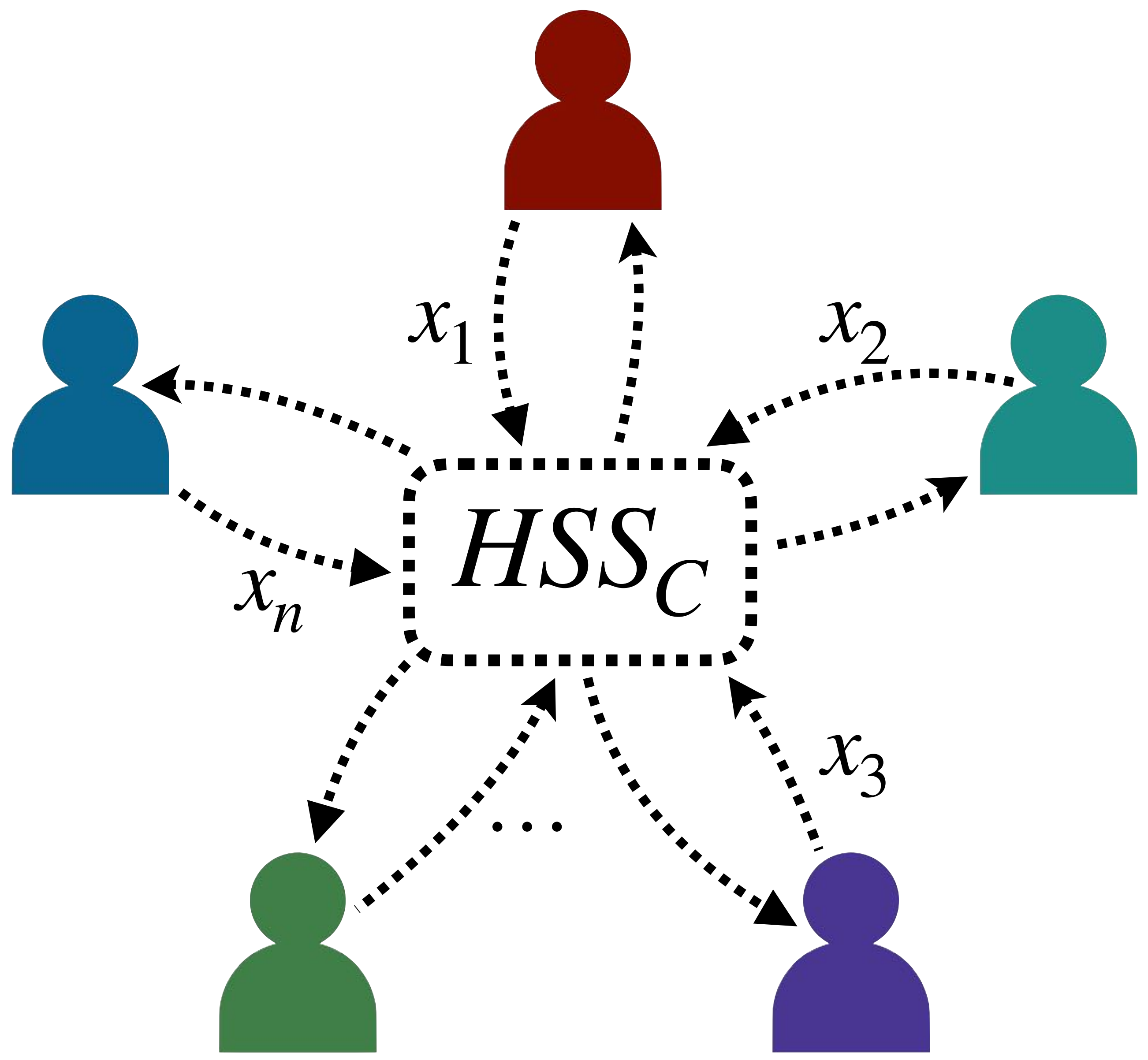
Setting: semi-honest, $(n-1)$ corruptions



Comm (per-party):
$\Omega(|C|)$

For "classical" protocols [GMW87, BGW88, BMR90] and their extensions

Solution: HSS-based MPC



Comm (per-party):
$\Omega\left(|x_i| + |C(\vec{x})|\right)$

Prior Works: The “2-party” Barrier

Prior Works: The "2-party" Barrier

<u>HSS landscape:</u>

Prior Works: The “2-party” Barrier

HSS landscape:

- First 2-party HSS for log-depth circuits from DDH [BGI16].

Prior Works: The "2-party" Barrier

<u>HSS landscape:</u>

- First 2-party HSS for log-depth circuits from DDH [BGI16].
- Later 2-party schemes from LWE [BKS19], DCR [FGJS17, OSY21, RS21], LPN [CM21], and class groups [ADOS22].

Prior Works: The “2-party” Barrier

HSS landscape:

- First 2-party HSS for log-depth circuits from DDH [BGI16].
- Later 2-party schemes from LWE [BKS19], DCR [FGJS17, OSY21, RS21], LPN [CM21], and class groups [ADOS22].
- None supporting >2 parties except those using ***indistinguishability obfuscation*** (iO) [BGI15], or (specific) ***fully homomorphic encryption*** (FHE) schemes [DHRW16, BGI+18].

Prior Works: The "2-party" Barrier

HSS landscape:

- First 2-party HSS for log-depth circuits from DDH [BGI16].
- Later 2-party schemes from LWE [BKS19], DCR [FGJS17, OSY21, RS21], LPN [CM21], and class groups [ADOS22].
- None supporting >2 parties except those using ***indistinguishability obfuscation*** (iO) [BGI15], or (specific) ***fully homomorphic encryption*** (FHE) schemes [DHRW16, BGI+18].

Sublinear MPC landscape:

Prior Works: The "2-party" Barrier

HSS landscape:

- First 2-party HSS for log-depth circuits from DDH [BGI16].
- Later 2-party schemes from LWE [BKS19], DCR [FGJS17, OSY21, RS21], LPN [CM21], and class groups [ADOS22].
- None supporting >2 parties except those using ***indistinguishability obfuscation*** (iO) [BGI15], or (specific) ***fully homomorphic encryption*** (FHE) schemes [DHRW16, BGI+18].

Sublinear MPC landscape:

- 2-party HSS $\implies$ 2-party sublinear MPC for ***layered*** Boolean circuits [BGI16]

Prior Works: The "2-party" Barrier

<u>HSS landscape:</u>

- First 2-party HSS for log-depth circuits from DDH [BGI16].
- Later 2-party schemes from LWE [BKS19], DCR [FGJS17, OSY21, RS21], LPN [CM21], and class groups [ADOS22].
- None supporting >2 parties except those using ***<u>indistinguishability obfuscation</u>*** (iO) [BGI15], or (specific) ***<u>fully homomorphic encryption</u>*** (FHE) schemes [DHRW16, BGI+18].

<u>Sublinear MPC landscape:</u>

- 2-party HSS $\Longrightarrow$ 2-party sublinear MPC for ***<u>layered</u>*** Boolean circuits [BGI16]
- 3-party and 5-party sublinear MPC based on an ***<u>array</u>*** of assumptions [BCM23]

Prior Works: The "2-party" Barrier

HSS landscape:

- First 2-party HSS for log-depth circuits from DDH [BGI16].
- Later 2-party schemes from LWE [BKS19], DCR [FGJS17, OSY21, RS21], LPN [CM21], and class groups [ADOS22].
- None supporting >2 parties except those using ***indistinguishability obfuscation*** (iO) [BGI15], or (specific) ***fully homomorphic encryption*** (FHE) schemes [DHRW16, BGI+18].

Sublinear MPC landscape:

- 2-party HSS $\Longrightarrow$ 2-party sublinear MPC for ***layered*** Boolean circuits [BGI16]
- 3-party and 5-party sublinear MPC based on an ***array*** of assumptions [BCM23]

Can we achieve HSS and sublinear MPC for arbitrary number of parties, without using iO or FHE?

Our Results: Multi-Party HSS & More

Our Results: Multi-Party HSS & More

<u>Theorem 1:</u> Assuming ***<u>Sparse LPN</u>*** (over $\mathbb{F}$), there exists HSS for ***<u>arbitrary</u>*** number of parties, with $1/\text{poly}(\lambda)$ error and linear reconstruction*, for the following function classes:

* *or $negl(\lambda)$ error but <u>non-linear</u> reconstruction*

Our Results: Multi-Party HSS & More

Theorem 1: Assuming ***Sparse LPN*** (over $\mathbb{F}$), there exists HSS for ***arbitrary*** number of parties, with $1/\text{poly}(\lambda)$ error and linear reconstruction*, for the following function classes:

** or $negl(\lambda)$ error but non-linear reconstruction*

1. $O(\log \lambda / \log \log \lambda)-$degree multivariate polynomials over $\mathbb{F}$, consisting of polynomial number of monomials, e.g.

Our Results: Multi-Party HSS & More

Theorem 1: Assuming ***Sparse LPN*** (over $\mathbb{F}$), there exists HSS for ***arbitrary*** number of parties, with $1/\text{poly}(\lambda)$ error and linear reconstruction*, for the following function classes:

** or $negl(\lambda)$ error but non-linear reconstruction*

1. $O(\log \lambda / \log\log \lambda)-$degree multivariate polynomials over $\mathbb{F}$, consisting of polynomial number of monomials, e.g.

$$f(x_1, \ldots, x_m) = \sum_{\text{poly}(\lambda)\ \text{terms}} x_{i_1} \ldots x_{i_s}, \qquad s = O(\log \lambda / \log\log \lambda).$$

Our Results: Multi-Party HSS & More

Theorem 1: Assuming ***Sparse LPN*** (over $\mathbb{F}$), there exists HSS for ***arbitrary*** number of parties, with $1/\mathrm{poly}(\lambda)$ error and linear reconstruction*, for the following function classes:

** or $negl(\lambda)$ error but non-linear reconstruction*

1. $O(\log \lambda / \log \log \lambda)-$degree multivariate polynomials over $\mathbb{F}$, consisting of polynomial number of monomials, e.g.

$$f(x_1, \ldots, x_m) = \sum_{\text{poly}(\lambda)\text{ terms}} x_{i_1} \ldots x_{i_s}, \qquad s = O(\log \lambda / \log \log \lambda).$$

2. $(c \cdot \log \log \lambda)-$depth arithmetic circuits over $\mathbb{F}$. (for any $c < 1$)

Our Results: Multi-Party HSS & More

<u>Theorem 1:</u> Assuming ***<u>Sparse LPN</u>*** (over $\mathbb{F}$), there exists HSS for ***<u>arbitrary</u>*** number of parties, with $1/\text{poly}(\lambda)$ error and linear reconstruction*, for the following function classes:

* *or $negl(\lambda)$ error but <u>non-linear</u> reconstruction*

1. $O(\log\lambda / \log\log\lambda)-$degree multivariate polynomials over $\mathbb{F}$, consisting of polynomial number of monomials, e.g.

$$f(x_1,\ldots,x_m) = \sum_{\text{poly}(\lambda)\ \text{terms}} x_{i_1}\ldots x_{i_s}, \qquad s = O(\log\lambda / \log\log\lambda).$$

2. $(c\cdot\log\log\lambda)-$depth arithmetic circuits over $\mathbb{F}$. (for any $c<1$)

<u>Theorem 2:</u> Assuming ***<u>Sparse LPN</u>*** and OTs*, there exists ***<u>sublinear</u>*** MPC for *<u>layered</u>* Boolean circuits, with per-party communication $\approx \omega(1)\cdot S/\log\log S$ for a layered circuit of size S.

* known from LPN with noise $1/\sqrt{n}$ [Ale03], or a *<u>specific</u>* parameter setting for sparse LPN [ABW10]

Our Assumption: Sparse LPN

Our Assumption: Sparse LPN

Learning Parity with Noise (LPN): for $A \leftarrow \mathbb{F}^{n \times m}$, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

$$Ber(\mathbb{F}, \epsilon) = \begin{cases} 0 \text{ w.p. } 1 - \epsilon \\ x \text{ w.p. } \frac{\epsilon}{|\mathbb{F}| - 1} \\ \forall x \neq 0 \end{cases}$$

Our Assumption: Sparse LPN

Learning Parity with Noise (LPN): for $A \leftarrow \mathbb{F}^{n \times m}$, $s \leftarrow \mathbb{F}^{n}$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^{m}$, $u \leftarrow \mathbb{F}^{m}$,
we have

$$(A, sA + e) \approx_c (A, u)$$

$$Ber(\mathbb{F}, \epsilon) = \begin{cases} 0 \text{ w.p. } 1 - \epsilon \\ x \text{ w.p. } \frac{\epsilon}{|\mathbb{F}| - 1} \\ \forall x \neq 0 \end{cases}$$

Our Assumption: Sparse LPN

Learning Parity with Noise (LPN): for $A \leftarrow \mathbb{F}^{n \times m}$, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$,
we have

$$Ber(\mathbb{F}, \epsilon) = \begin{cases} 0 \text{ w.p. } 1 - \epsilon \\ x \text{ w.p. } \frac{\epsilon}{|\mathbb{F}| - 1} \\ \forall x \neq 0 \end{cases}$$

$$(A, sA + e) \approx_c (A, u)$$

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with k−sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$,
we have

$$(A, sA + e) \approx_c (A, u)$$

Our Assumption: Sparse LPN

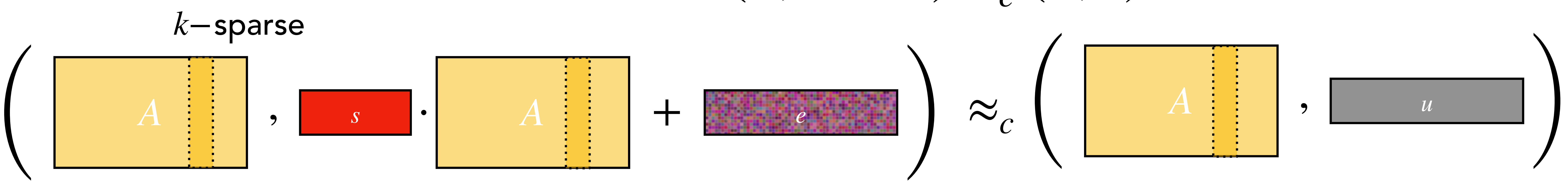
Learning Parity with Noise (LPN): for $A \leftarrow \mathbb{F}^{n \times m}$, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

$$Ber(\mathbb{F}, \epsilon) = \begin{cases} 0 \text{ w.p. } 1 - \epsilon \\ x \text{ w.p. } \frac{\epsilon}{|\mathbb{F}| - 1} \\ \forall x \neq 0 \end{cases}$$

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with $k-$sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$



Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

Our Assumption: Sparse LPN

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with k−sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

Our Assumption: Sparse LPN

<u>Sparse LPN:</u> for $A \leftarrow \mathbb{F}^{n \times m}$ with <u>$k-$sparse</u> columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

<u>Our Setting:</u> $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

<u>History:</u>

Our Assumption: Sparse LPN

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with $\underline{k-\text{sparse}}$ columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

History:

- When $|\mathbb{F}| = 2$, this problem (and close variants) have been studied extensively in works on average-case complexity [Gol00, CM01, Fei02, MST03, FKO06, AOW15, AL16, KMOW17].

Our Assumption: Sparse LPN

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with $k-$sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

History:

- When $|\mathbb{F}| = 2$, this problem (and close variants) have been studied extensively in works on average-case complexity [Gol00, CM01, Fei02, MST03, FKO06, AOW15, AL16, KMOW17].
- **Prior Applications:** hardness of approximation [Ale03], linear-stretch PRGs with constant locality [AIK06], constant-overhead commitments [IKOS08], PKE and semi-honest OT [ABW10], pseudorandom correlation generators (PCGs) [BCG+18, BCG+19], and constant-rate VOLEs [ADI+17, AK23]

Our Assumption: Sparse LPN

<u>**Sparse LPN:**</u> for $A \leftarrow \mathbb{F}^{n\times m}$ with <u>$k-$sparse</u> columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

<u>**Our Setting:**</u> $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

Our Assumption: Sparse LPN

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with $k-$sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

Hardness:

Our Assumption: Sparse LPN

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with $k-$sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

Hardness:

- Matrix A has probability $O(n^{-\text{poly} \log n})$ of being “bad”, i.e. having a ***sparse*** linear dependency.

Our Assumption: Sparse LPN

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with $k-$sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

Hardness:

- Matrix A has probability $O(n^{-\text{poly} \log n})$ of being "bad", i.e. having a ***sparse*** linear dependency.
- Outside of this "bad" choice, the best attacks (ISD-based) takes time $2^{\widetilde{O}(n^{1-\delta})}$.

Our Assumption: Sparse LPN

Sparse LPN: for $A \leftarrow \mathbb{F}^{n \times m}$ with $k-$sparse columns, $s \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)^m$, $u \leftarrow \mathbb{F}^m$, we have

$$(A, sA + e) \approx_c (A, u)$$

Our Setting: $k = \text{poly}(\log n)$ and $\epsilon = O(n^{-\delta})$ for any $\delta \in (0,1)$.

Hardness:

- Matrix A has probability $O(n^{-\text{poly} \log n})$ of being "bad", i.e. having a ***sparse*** linear dependency.
- Outside of this "bad" choice, the best attacks (ISD-based) takes time $2^{\widetilde{O}(n^{1-\delta})}$.
- This parameter regime is not known to imply PKE [ABW10] $\Longrightarrow$ multi-party HSS* potentially ***weaker*** than PKE.

* we consider secret-key HSS in this work, public-key HSS necessarily implies PKE

HSS Construction: Motivation

HSS Construction: Motivation

[BGI16] Template:

HSS Construction: Motivation

<u>[BGI16] Template:</u>

- Each share of an input $x \in \mathbb{F}$ include:

HSS Construction: Motivation

[BGI16] Template:

- Each share of an input $x \in \mathbb{F}$ include: $\left\{\right.$
  - (linear) secret share $[x]$,

HSS Construction: Motivation

[BGI16] Template:

- Each share of an input $x \in \mathbb{F}$ include:
  - (linear) secret share $[x]$,
  - (linearly homomorphic) encryption $Enc_s(x)$ under key s,

HSS Construction: Motivation

[BGI16] Template:

- Each share of an input $x \in \mathbb{F}$ include:
  - (linear) secret share $[x]$,
  - (linearly homomorphic) encryption $Enc_s(x)$ under key s,
  - secret shares $[x \cdot s]$ and encryptions $Enc_s(x \cdot s)$.

HSS Construction: Motivation

[BGI16] Template:

- Each share of an input $x \in \mathbb{F}$ include:
  - (linear) secret share $[x]$,
  - (linearly homomorphic) encryption $Enc_s(x)$ under key s,
  - secret shares $[x \cdot s]$ and encryptions $Enc_s(x \cdot s)$.

- **Invariant:** any intermediate value y is stored as ***noisy*** shares $\left[y + e_y\right], \left[y \cdot s + e_{y \cdot s}\right]$.

HSS Construction: Motivation

[BGI16] Template:

- Each share of an input $x \in \mathbb{F}$ include:
  - (linear) secret share $[x]$,
  - (linearly homomorphic) encryption $Enc_s(x)$ under key s,
  - secret shares $[x \cdot s]$ and encryptions $Enc_s(x \cdot s)$.

- **Invariant:** any intermediate value y is stored as ***noisy*** shares $\left[y + e_y\right], \left[y \cdot s + e_{y \cdot s}\right]$.

- **Multiplication:** given $\left[y + e_y\right], \left[y \cdot s + e_{y \cdot s}\right]$ and $Enc_s(x), Enc_s(x \cdot s)$, compute

HSS Construction: Motivation

[BGI16] Template:

- Each share of an input $x \in \mathbb{F}$ include:
  - (linear) secret share $[x]$,
  - (linearly homomorphic) encryption $Enc_s(x)$ under key s,
  - secret shares $[x \cdot s]$ and encryptions $Enc_s(x \cdot s)$.

- **Invariant:** any intermediate value y is stored as ***noisy*** shares $\left[y + e_y\right], \left[y \cdot s + e_{y \cdot s}\right]$.

- **Multiplication:** given $\left[y + e_y\right], \left[y \cdot s + e_{y \cdot s}\right]$ and $Enc_s(x), Enc_s(x \cdot s)$, compute

$$\begin{cases} \left[Enc_s(xy + e_{xy})\right] := \left(\left[y + e_y\right], -\left[y \cdot s + e_{y \cdot s}\right]\right) \cdot Enc_s(x) \\ \left[Enc_s(xy \cdot s + e_{xy \cdot s})\right] := \left(\left[y + e_y\right], -\left[y \cdot s + e_{y \cdot s}\right]\right) \cdot Enc_s(x \cdot s) \end{cases} \xRightarrow{\text{"rounding"}} \begin{cases} \left[xy + e_{xy}\right] \\ \left[xy \cdot s + e_{xy \cdot s}\right] \end{cases}$$

HSS Construction: Motivation

[BGI16] Template:

- Each share of an input $x \in \mathbb{F}$ include:
  - (linear) secret share $[x]$,
  - (linearly homomorphic) encryption $Enc_s(x)$ under key s,
  - secret shares $[x \cdot s]$ and encryptions $Enc_s(x \cdot s)$.

- **Invariant:** any intermediate value y is stored as ***noisy*** shares $\left[y + e_y\right], \left[y \cdot s + e_{y \cdot s}\right]$.

- **Multiplication:** given $\left[y + e_y\right], \left[y \cdot s + e_{y \cdot s}\right]$ and $Enc_s(x), Enc_s(x \cdot s)$, compute

$$\begin{cases} \left[Enc_s(xy + e_{xy})\right] := \left(\left[y + e_y\right], -\left[y \cdot s + e_{y \cdot s}\right]\right) \cdot Enc_s(x) \\ \left[Enc_s(xy \cdot s + e_{xy \cdot s})\right] := \left(\left[y + e_y\right], -\left[y \cdot s + e_{y \cdot s}\right]\right) \cdot Enc_s(x \cdot s) \end{cases} \overset{\text{"rounding"}}{\Longrightarrow} \begin{cases} \left[xy + e_{xy}\right] \\ \left[xy \cdot s + e_{xy \cdot s}\right] \end{cases}$$

Limitation: ***Distributed rounding*** procedure only works for 2 parties.

HSS Construction: Motivation

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

$\implies$ ciphertext over same field as plaintext, no rounding needed!

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \big(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\big)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \big(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\big)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

$\implies$ ciphertext over same field as plaintext, no rounding needed!

Multiplication: given $\big[y + e_y\big], \Big(\big[y \cdot s_i + e_{y \cdot s_i}\big]\Big)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\big(Enc_{\vec{s}}(x \cdot s_i)\big)_{i=1}^{n}$, compute

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

$\Longrightarrow$ ciphertext over same field as plaintext, no rounding needed!

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + e + x\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_i \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_{i,j} \end{cases}$$

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

$\Longrightarrow$ ciphertext over same field as plaintext, no rounding needed!

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + e + x\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_i \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_{i,j} \end{cases}$$

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

$\Longrightarrow$ ciphertext over same field as plaintext, no rounding needed!

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + e + x\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_i \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_{i,j} \end{cases}$$

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

$\Longrightarrow$ ciphertext over same field as plaintext, no rounding needed!

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + e + x\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_i \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right) - \sum_{j=1}^{n} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_{i,j} \end{cases}$$

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right), \text{ where } \vec{a} \leftarrow \mathbb{F}^n, e \leftarrow Ber(\mathbb{F}, \epsilon),$$

$$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right), \text{ where } \vec{a}_i \leftarrow \mathbb{F}^n, e_i \leftarrow Ber(\mathbb{F}, \epsilon) \text{ for all } i \in [n].$$

$\Longrightarrow$ ciphertext over same field as plaintext, no rounding needed!

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + \boxed{e} + x\right) - \boxed{\sum_{j=1}^{n}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_i} \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + \boxed{e_i} + x \cdot s_i\right) - \boxed{\sum_{j=1}^{n}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_{i,j}} \end{cases}$$

HSS Construction: Motivation

Insight 1: Use LPN-based encryption

$$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right), \text{ where } \vec{a} \leftarrow \mathbb{F}^n,\ e \leftarrow Ber(\mathbb{F}, \epsilon),$$

$$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right), \text{ where } \vec{a}_i \leftarrow \mathbb{F}^n,\ e_i \leftarrow Ber(\mathbb{F}, \epsilon) \text{ for all } i \in [n].$$

$\Longrightarrow$ ciphertext over same field as plaintext, no rounding needed!

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + \boxed{e} + x\right) - \boxed{\sum_{j=1}^{n}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_i} \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + \boxed{e_i} + x \cdot s_i\right) - \boxed{\sum_{j=1}^{n}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_{i,j}} \end{cases}$$

Problem: Noise grows by factor of $O(n) \Longrightarrow$ too fast!

HSS Construction: Motivation

HSS Construction: Motivation

Insight 2: Use *Sparse LPN*-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$ is k-sparse, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$ is k-sparse, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

HSS Construction: Motivation

Insight 2: Use *Sparse LPN*-based encryption

$Enc_{\vec{s}}(x) := \big(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\big)$, where $\vec{a} \leftarrow \mathbb{F}^n$ is *k-sparse*, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \big(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\big)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$ is *k-sparse*, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + e + x\right) - \sum_{a_i \neq 0} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_i \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + e_y\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right) - \sum_{a_{i,j} \neq 0} \left[y \cdot s_j + e_{y \cdot s_j}\right] \cdot a_{i,j} \end{cases}$$

HSS Construction: Motivation

Insight 2: Use ***Sparse LPN***-based encryption

$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right)$, where $\vec{a} \leftarrow \mathbb{F}^n$ is k-sparse, $e \leftarrow Ber(\mathbb{F}, \epsilon)$,

$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right)$, where $\vec{a}_i \leftarrow \mathbb{F}^n$ is k-sparse, $e_i \leftarrow Ber(\mathbb{F}, \epsilon)$ for all $i \in [n]$.

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + \boxed{e} + x\right) - \boxed{\sum_{a_i \neq 0}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_i} \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + \boxed{e_i} + x \cdot s_i\right) - \boxed{\sum_{a_{i,j} \neq 0}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_{i,j}} \end{cases}$$

HSS Construction: Motivation

Insight 2: Use *Sparse LPN*-based encryption

$$Enc_{\vec{s}}(x) := \left(\vec{a}, \langle \vec{s}, \vec{a} \rangle + e + x\right), \text{ where } \vec{a} \leftarrow \mathbb{F}^n \text{ is } \underline{k\text{-sparse}},\ e \leftarrow Ber(\mathbb{F}, \epsilon),$$

$$Enc_{\vec{s}}(x \cdot s_i) := \left(\vec{a}_i, \langle \vec{s}, \vec{a}_i \rangle + e_i + x \cdot s_i\right), \text{ where } \vec{a}_i \leftarrow \mathbb{F}^n \text{ is } \underline{k\text{-sparse}},\ e_i \leftarrow Ber(\mathbb{F}, \epsilon) \text{ for all } i \in [n].$$

Multiplication: given $\left[y + e_y\right], \left(\left[y \cdot s_i + e_{y \cdot s_i}\right]\right)_{i=1}^{n}$ and $Enc_{\vec{s}}(x)$, $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$, compute

$$\begin{cases} \left[xy + e_{xy}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a} \rangle + \boxed{e} + x\right) - \boxed{\sum_{a_i \neq 0}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_i} \\ \left[xy \cdot s_i + e_{xy \cdot s_i}\right] := \left[y + \boxed{e_y}\right] \cdot \left(\langle \vec{s}, \vec{a}_i \rangle + \boxed{e_i} + x \cdot s_i\right) - \boxed{\sum_{a_{i,j} \neq 0}} \left[y \cdot s_j + \boxed{e_{y \cdot s_j}}\right] \cdot \boxed{a_{i,j}} \end{cases}$$

Noise growth: only $O(k)$ each time $\Longrightarrow$ for degree-d monomials, noise grows by $k^{O(d)}$.

HSS Construction: Parameters

HSS Construction: Parameters

Parameter Setting: To achieve a desired $1/\text{poly}(\lambda)$ correctness error for degree-d polynomials with M terms, we need:

HSS Construction: Parameters

Parameter Setting: To achieve a desired $1/\text{poly}(\lambda)$ correctness error for degree-d polynomials with M terms, we need:

$$\frac{1}{n^{-\delta}} \cdot k^{O(d)} \cdot M < \frac{1}{\text{poly}(\lambda)}, \qquad \text{where } \epsilon = \frac{1}{n^{-\delta}} \text{ is the initial noise rate.}$$

HSS Construction: Parameters

<u>Parameter Setting:</u> To achieve a desired $1/\text{poly}(\lambda)$ correctness error for degree-d polynomials with M terms, we need:

$$\frac{1}{n^{-\delta}} \cdot k^{O(d)} \cdot M < \frac{1}{\text{poly}(\lambda)}, \quad \text{where } \epsilon = \frac{1}{n^{-\delta}} \text{ is the initial noise rate.}$$

For any $d = O(\log \lambda / \log \log \lambda)$, $M = \text{poly}(\lambda)$, and $k = \text{poly} \log n$, it suffices to set $n = O\left(\lambda^C\right)$ for a large enough exponent C.

HSS Construction: Parameters

Parameter Setting: To achieve a desired $1/\text{poly}(\lambda)$ correctness error for degree-d polynomials with M terms, we need:

$$\frac{1}{n^{-\delta}} \cdot k^{O(d)} \cdot M < \frac{1}{\text{poly}(\lambda)}, \quad \text{where } \epsilon = \frac{1}{n^{-\delta}} \text{ is the initial noise rate.}$$

For any $d = O(\log \lambda / \log \log \lambda)$, $M = \text{poly}(\lambda)$, and $k = \text{poly} \log n$, it suffices to set $n = O\left(\lambda^C\right)$ for a large enough exponent C.

Advantages:

HSS Construction: Parameters

Parameter Setting: To achieve a desired $1/\text{poly}(\lambda)$ correctness error for degree-d polynomials with M terms, we need:

$$\frac{1}{n^{-\delta}} \cdot k^{O(d)} \cdot M < \frac{1}{\text{poly}(\lambda)}, \quad \text{where } \epsilon = \frac{1}{n^{-\delta}} \text{ is the initial noise rate.}$$

For any $d = O(\log \lambda / \log \log \lambda)$, $M = \text{poly}(\lambda)$, and $k = \text{poly} \log n$, it suffices to set $n = O\left(\lambda^C\right)$ for a large enough exponent C.

Advantages:

- Our HSS can be used with ***any*** linear secret sharing scheme.

HSS Construction: Parameters

Parameter Setting: To achieve a desired $1/\text{poly}(\lambda)$ correctness error for degree-d polynomials with M terms, we need:

$$\frac{1}{n^{-\delta}} \cdot k^{O(d)} \cdot M < \frac{1}{\text{poly}(\lambda)}, \quad \text{where } \epsilon = \frac{1}{n^{-\delta}} \text{ is the initial noise rate.}$$

For any $d = O(\log \lambda / \log \log \lambda)$, $M = \text{poly}(\lambda)$, and $k = \text{poly} \log n$, it suffices to set $n = O\left(\lambda^C\right)$ for a large enough exponent C.

Advantages:

- Our HSS can be used with ***any*** linear secret sharing scheme.
- Small computation overhead $O(k) = \text{poly} \log n$ for each multiplication.

HSS Construction: Security

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

$$\left\{ [x]_\ell, \left([x \cdot s_i]_\ell\right)_{i=1}^n \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x), \left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^n \right\} \approx_c \left\{ [x']_\ell, \left([x' \cdot s_i]_\ell\right)_{i=1}^n \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x'), \left(Enc_{\vec{s}}(x' \cdot s_i)\right)_{i=1}^n \right\}$$

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

$$\left\{ [x]_\ell, \left([x \cdot s_i]_\ell\right)_{i=1}^{n} \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x), \left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n} \right\} \approx_c \left\{ [x']_\ell, \left([x' \cdot s_i]_\ell\right)_{i=1}^{n} \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x'), \left(Enc_{\vec{s}}(x' \cdot s_i)\right)_{i=1}^{n} \right\}$$

- Secret shares are indistinguishable due to t-privacy of $[\,\cdot\,]$.

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

$$\left\{ [x]_\ell, \left([x \cdot s_i]_\ell\right)_{i=1}^{n} \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x), \left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n} \right\} \approx_c \left\{ [x']_\ell, \left([x' \cdot s_i]_\ell\right)_{i=1}^{n} \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x'), \left(Enc_{\vec{s}}(x' \cdot s_i)\right)_{i=1}^{n} \right\}$$

- Secret shares are indistinguishable due to t-privacy of $[\,\cdot\,]$.
- Encryptions $Enc_{\vec{s}}(x) \approx_c Enc_{\vec{s}}(x')$ due to semantic security.

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

$$\boxed{\left\{ [x]_\ell, \left([x \cdot s_i]_\ell\right)_{i=1}^{n} \right\}_{\ell \in T}} \cup \left\{ \boxed{Enc_{\vec{s}}(x)}, \left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n} \right\} \approx_c \boxed{\left\{ [x']_\ell, \left([x' \cdot s_i]_\ell\right)_{i=1}^{n} \right\}_{\ell \in T}} \cup \left\{ \boxed{Enc_{\vec{s}}(x')}, \left(Enc_{\vec{s}}(x' \cdot s_i)\right)_{i=1}^{n} \right\}$$

- Secret shares are indistinguishable due to t-privacy of $[\,\cdot\,]$.
- Encryptions $Enc_{\vec{s}}(x) \approx_c Enc_{\vec{s}}(x')$ due to semantic security.

KDM Security: Need to show *key-dependent message* (KDM) security of $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$.

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

$$\left\{[x]_\ell, \left([x \cdot s_i]_\ell\right)_{i=1}^{n}\right\}_{\ell \in T} \cup \left\{Enc_{\vec{s}}(x), \left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}\right\} \approx_c \left\{[x']_\ell, \left([x' \cdot s_i]_\ell\right)_{i=1}^{n}\right\}_{\ell \in T} \cup \left\{Enc_{\vec{s}}(x'), \left(Enc_{\vec{s}}(x' \cdot s_i)\right)_{i=1}^{n}\right\}$$

- Secret shares are indistinguishable due to t-privacy of $[\cdot]$.
- Encryptions $Enc_{\vec{s}}(x) \approx_c Enc_{\vec{s}}(x')$ due to semantic security.

KDM Security: Need to show *key-dependent message* (KDM) security of $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^{n}$.

- Existing KDM proofs for LWE/LPN ***do not apply***!

  $\Longrightarrow$ Problem is distribution of sparse matrices not uniform.

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

$$\boxed{\left\{ [x]_\ell, \left([x \cdot s_i]_\ell\right)_{i=1}^n \right\}_{\ell \in T}} \cup \left\{ \boxed{Enc_{\vec{s}}(x)}, \left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^n \right\} \approx_c \boxed{\left\{ [x']_\ell, \left([x' \cdot s_i]_\ell\right)_{i=1}^n \right\}_{\ell \in T}} \cup \left\{ \boxed{Enc_{\vec{s}}(x')}, \left(Enc_{\vec{s}}(x' \cdot s_i)\right)_{i=1}^n \right\}$$

- Secret shares are indistinguishable due to t-privacy of $[\,\cdot\,]$.
- Encryptions $Enc_{\vec{s}}(x) \approx_c Enc_{\vec{s}}(x')$ due to semantic security.

KDM Security: Need to show *key-dependent message* (KDM) security of $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^n$.

- Existing KDM proofs for LWE/LPN ***do not apply***!

  $\Longrightarrow$ Problem is distribution of sparse matrices not uniform.

- **Our Idea:** use security for k-sparse to argue KDM security for $(2k-1)$-sparse

HSS Construction: Security

HSS Security: For any subset of parties T of size $\leq t$, and any $x, x' \in \mathbb{F}$, we need to show

$$\left\{ [x]_\ell, \left([x \cdot s_i]_\ell\right)_{i=1}^n \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x), \left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^n \right\} \approx_c \left\{ [x']_\ell, \left([x' \cdot s_i]_\ell\right)_{i=1}^n \right\}_{\ell \in T} \cup \left\{ Enc_{\vec{s}}(x'), \left(Enc_{\vec{s}}(x' \cdot s_i)\right)_{i=1}^n \right\}$$

- Secret shares are indistinguishable due to t-privacy of $[\,\cdot\,]$.
- Encryptions $Enc_{\vec{s}}(x) \approx_c Enc_{\vec{s}}(x')$ due to semantic security.

KDM Security: Need to show *key-dependent message* (KDM) security of $\left(Enc_{\vec{s}}(x \cdot s_i)\right)_{i=1}^n$.

- Existing KDM proofs for LWE/LPN ***do not apply***!

  $\Longrightarrow$ Problem is distribution of sparse matrices not uniform.

- **Our Idea:** use security for k-sparse to argue KDM security for $(2k-1)$-sparse
- **Technical Issue:** our proof only works for $|\mathbb{F}| > 2$! $\Longrightarrow$ HSS for $\mathbb{F}_2$ can be done in $\mathbb{F}_4$

Summary

Summary

<u>Our Result:</u> Assuming ***<u>Sparse LPN</u>***, there exists HSS for $O(\log\log)$-depth arithmetic circuits, and sublinear MPC for *<u>layered</u>* Boolean circuits, both supporting ***<u>arbitrary</u>*** number of parties.

Summary

Our Result: Assuming ***Sparse LPN***, there exists HSS for $O(\log\log)$-depth arithmetic circuits, and sublinear MPC for *layered* Boolean circuits, both supporting ***arbitrary*** number of parties.

Future Directions:

Summary

Our Result: Assuming ***Sparse LPN***, there exists HSS for $O(\log \log)$-depth arithmetic circuits, and sublinear MPC for *layered* Boolean circuits, both supporting ***arbitrary*** number of parties.

Future Directions:

- Public-key multi-party HSS? (e.g. computations can be done on inputs from ***different*** clients)

Summary

Our Result: Assuming ***Sparse LPN***, there exists HSS for $O(\log\log)$-depth arithmetic circuits, and sublinear MPC for *layered* Boolean circuits, both supporting ***arbitrary*** number of parties.

Future Directions:

- Public-key multi-party HSS? (e.g. computations can be done on inputs from ***different*** clients)
- Concrete hardness of Sparse LPN?

Summary

<u>Our Result:</u> Assuming ***<u>Sparse LPN</u>***, there exists HSS for $O(\log\log)$-depth arithmetic circuits, and sublinear MPC for *<u>layered</u>* Boolean circuits, both supporting ***<u>arbitrary</u>*** number of parties.

<u>Future Directions:</u>

- Public-key multi-party HSS? (e.g. computations can be done on inputs from ***<u>different</u>*** clients)
- Concrete hardness of Sparse LPN?
- Improved efficiency for practical applications?

Summary

Our Result: Assuming ***Sparse LPN***, there exists HSS for $O(\log \log)$-depth arithmetic circuits, and sublinear MPC for *layered* Boolean circuits, both supporting ***arbitrary*** number of parties.

Future Directions:

- Public-key multi-party HSS? (e.g. computations can be done on inputs from ***different*** clients)
- Concrete hardness of Sparse LPN?
- Improved efficiency for practical applications?

Thank you! Questions?