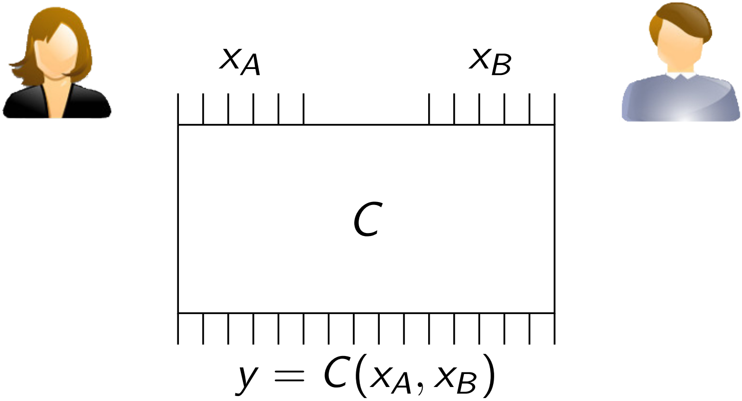
Sublinear-Communication MPC does not require FHE

Elette Boyle[1,2] Geoffroy Couteau[3] Pierre Meyer[1,3]

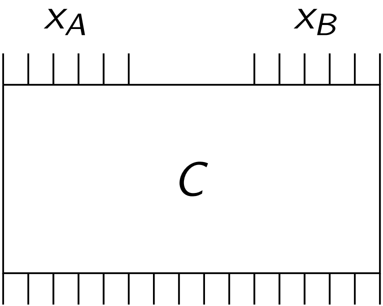
[1]Reichman University
[2]NTT Research
[3]Université Paris Cité, IRIF, CNRS

Eurocrypt 2023

Sublinear-Communication Secure Computation



Sublinear-Communication Secure Computation



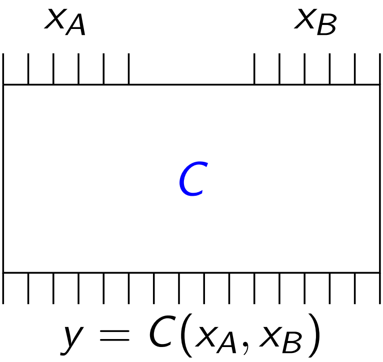
$$y = C(x_A, x_B)$$

Communication

$$\mathcal{O}(|x_A| + |x_B| + |y|) + o(|C|) + \text{poly}(\lambda)$$

Sublinear in the Circuit-Size

Sublinear-Communication Secure Computation



Supported Class?

P/Poly (ideally)

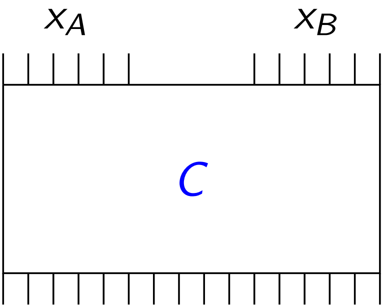
For now:
$\omega(1)$-depth
Layered Circuits

Communication

$$\mathcal{O}(|x_A| + |x_B| + |y|) + o(|C|) + \text{poly}(\lambda)$$

Sublinear in the Circuit-Size

Sublinear-Communication Secure Computation



$$y = C(x_A, x_B)$$

Semi-Honest

2PC or (2+)PC

Static

Dishonest Majority

Supported Class?

P/Poly (ideally)

For now:

$\omega(1)$-depth

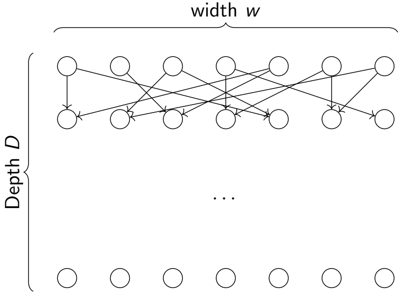
Layered Circuits

Communication

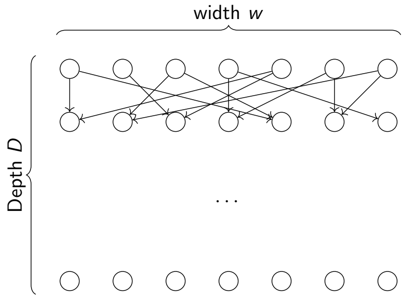
$$\mathcal{O}(|x_A| + |x_B| + |y|) + o(|C|) + \text{poly}(\lambda)$$

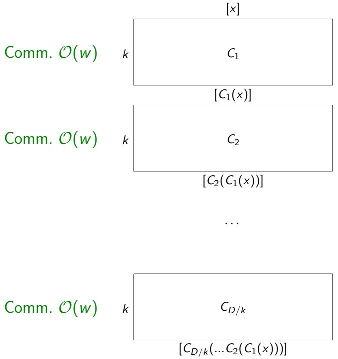
Sublinear in the Circuit-Size

Sublinear Secure Computation of Layered Circuits



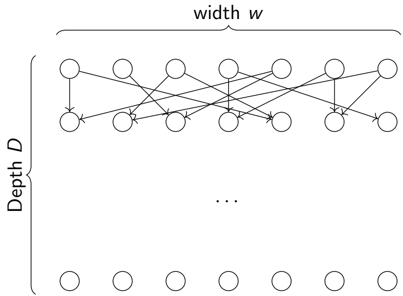
Sublinear Secure Computation of Layered Circuits

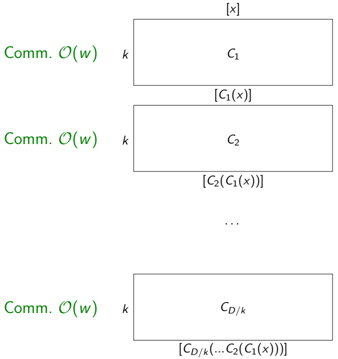




Total Comm. $\frac{D}{k} \cdot \mathcal{O}(w) = \mathcal{O}(s/k)$

Sublinear Secure Computation of Layered Circuits





Total Comm. $\frac{D}{k} \cdot \mathcal{O}(w) = \mathcal{O}(s/k)$

"Communication-Optimal" MPC for depth-k Circuits $\Rightarrow$ (Factor k) Sublinear MPC for Layered Circuits

Correlated Randomness

Fully Homomorphic Encryption

Homomorphic Secret Sharing

(or equivalently its “dual”: **Function Secret Sharing**)

Correlated Randomness

Trusted Setup

[Ishai-Kushilevitz-Meldgaard-Orlandi-Paskin'13]

[Damgård-Nielsen-Nielsen-Ranellucci'17]

[Couteau'19]

DDH

[Boyle-Gilboa-Ishai'16]

poly-modulus LWE

[Boyle-Kohl-Scholl'19]

Fully Homomorphic Encryption

Lattice-Based Assumptions

[Gentry'09]

[Brakerski-Vaikuntanathan'11]

...

Homomorphic Secret Sharing

DCR

[Fazio-Gennaro-Jafarikhah-Skeith'17]

[Orlandi-Scholl-Yacoubov'21]

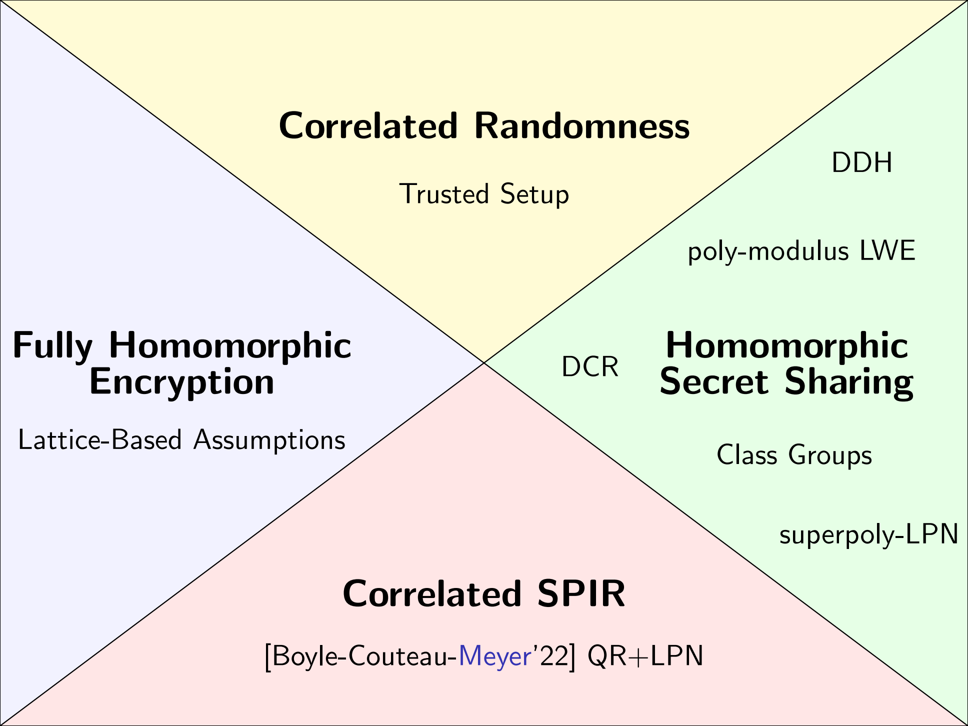
[Roy-Singh'21]

Class Groups

[Abram-Damgård-Orlandi-Scholl.'22]

superpoly-LPN

[Couteau-Meyer'21]

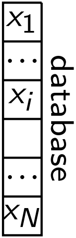

Correlated Randomness
Trusted Setup
DDH
poly-modulus LWE
Fully Homomorphic Encryption
Lattice-Based Assumptions
DCR
Homomorphic Secret Sharing
Class Groups
superpoly-LPN
Correlated SPIR
[Boyle-Couteau-Meyer'22] QR+LPN

(Single-Server) Symmetric Private Information Retrieval

[Chor-Goldreich-Kushilevitz-Sudan'95] and [Kushilevitz-Ostrovsky'97]

index: i

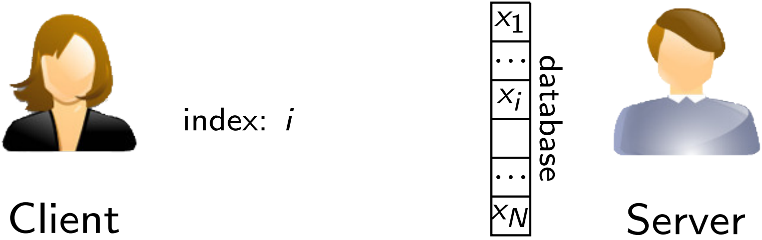
Client



Server

(Single-Server) Symmetric Private Information Retrieval

[Chor-Goldreich-Kushilevitz-Sudan'95] and [Kushilevitz-Ostrovsky'97]

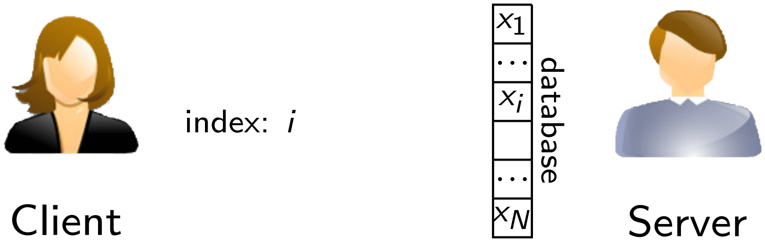


- **Information Retrieval:** The client gets x_i
- **Privacy:** The server does not learn i

- **Communication Requirement:** $o(N)$; ideally $\text{polylog}(N)$

(Single-Server) Symmetric Private Information Retrieval

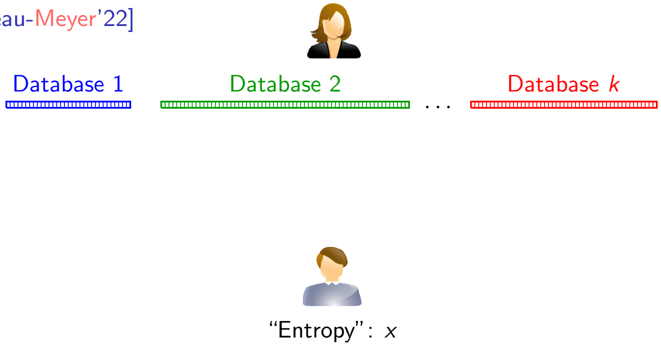
[Chor-Goldreich-Kushilevitz-Sudan'95] and [Kushilevitz-Ostrovsky'97]



- **Information Retrieval:** The client gets x_i
- **Privacy:** The server does not learn i
- **Symmetric Privacy:** The client does not learn x_j if $j \neq i$
- **Communication Requirement:** $o(N)$; ideally $\mathrm{polylog}(N)$

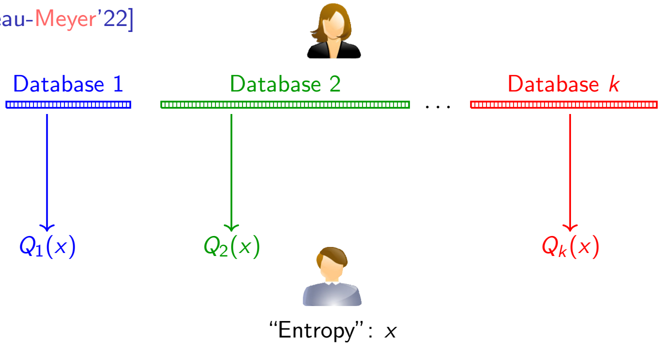
Correlated SPIR

[Boyle-Couteau-Meyer'22]



Correlated SPIR

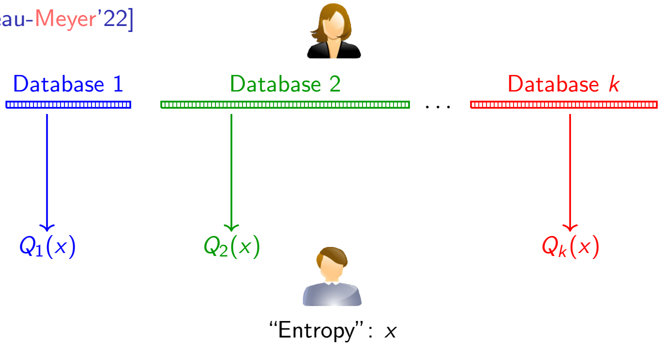
[Boyle-Couteau-Meyer'22]



- Public Correlation: $Q_1(\cdot), \ldots, Q_k(\cdot)$

Correlated SPIR

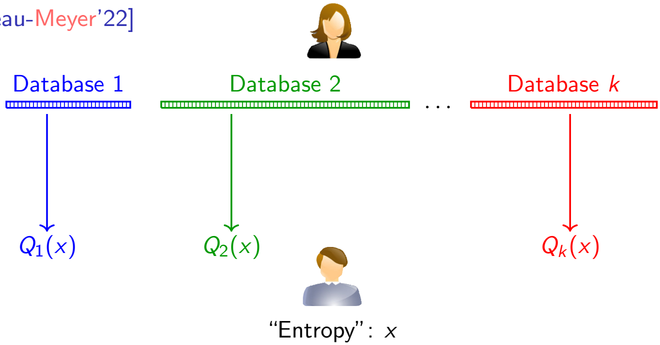
[Boyle-Couteau-Meyer'22]



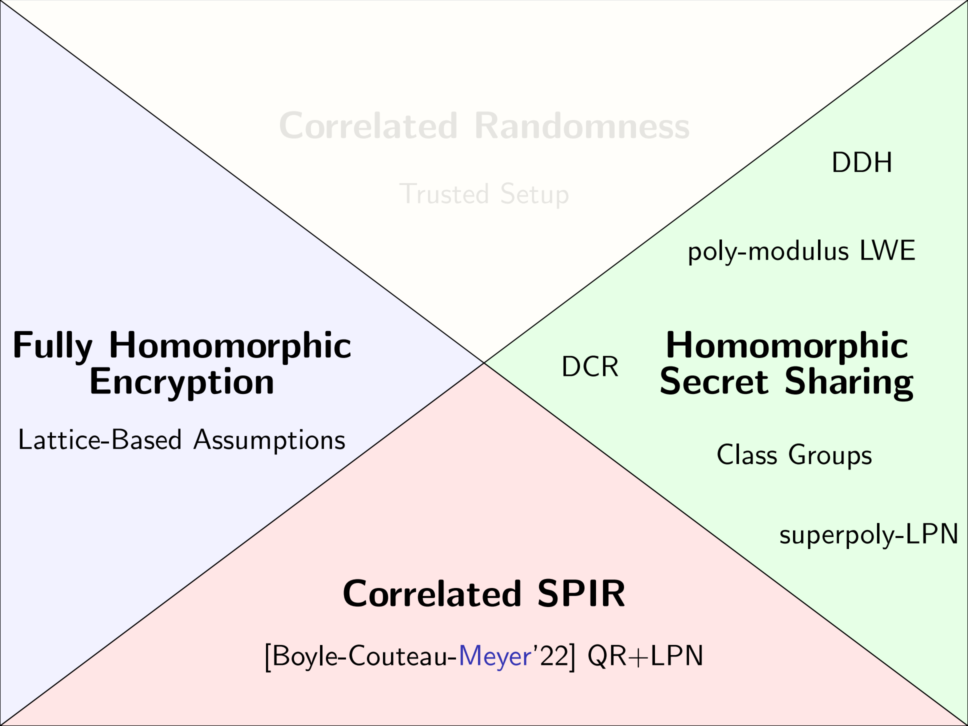
- Public Correlation: $Q_1(\cdot), \ldots, Q_k(\cdot)$
- Upload communication $\mathcal{O}(|x|)$ and Download $\mathcal{O}(k)$

Correlated SPIR

[Boyle-Couteau-Meyer'22]



- Public Correlation: $Q_1(\cdot), \ldots, Q_k(\cdot)$
- Upload communication $\mathcal{O}(|x|)$ and Download $\mathcal{O}(k)$
- [Boyle-Couteau-Meyer'22]:
  Construction based on rate-1 LHE of [Brakerski-Branco-Döttling-Pu'22]
  (LPN + $\{$QR $\vee$ DCR $\vee$ DDH $\vee$ poly-modulus LWE$\}$)


Correlated Randomness
Trusted Setup
DDH
poly-modulus LWE
Fully Homomorphic Encryption
Lattice-Based Assumptions
DCR
Homomorphic Secret Sharing
Class Groups
superpoly-LPN
Correlated SPIR
[Boyle-Couteau-Meyer'22] QR+LPN

Correlated Randomness

Extends to MPC

Fully Homomorphic Encryption

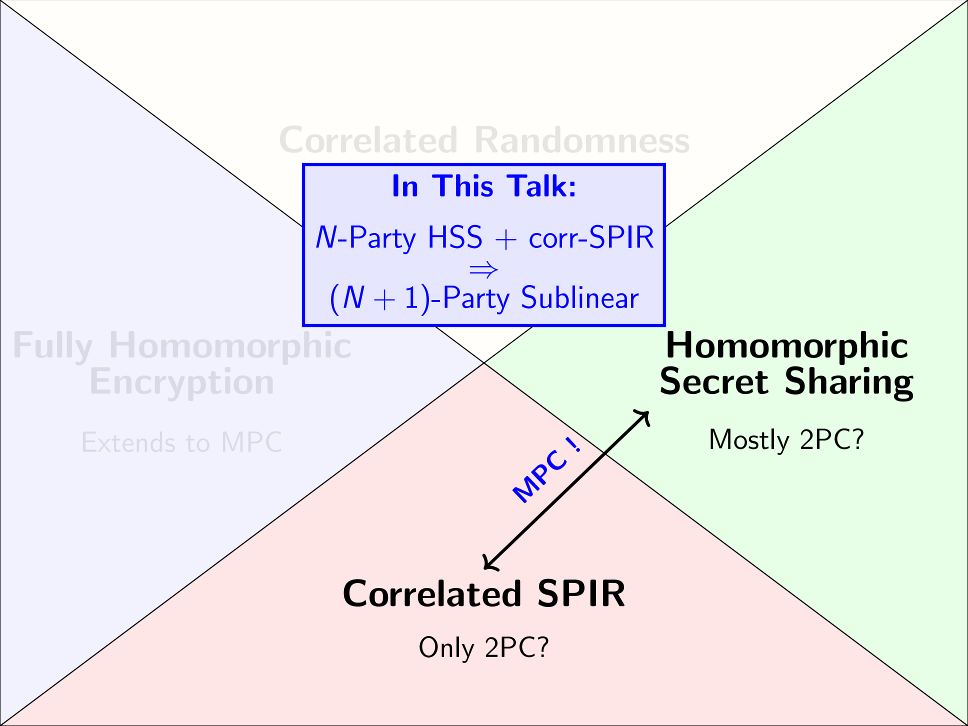
Extends to MPC

Homomorphic Secret Sharing

Mostly 2PC?

Correlated SPIR

Only 2PC?


Correlated Randomness
In This Talk:
N-Party HSS + corr-SPIR
⇒
(N + 1)-Party Sublinear
Fully Homomorphic Encryption
Extends to MPC
Homomorphic Secret Sharing
Mostly 2PC?
MPC !
Correlated SPIR
Only 2PC?

Selected Results for this Talk

1. **A Framework for Sublinear $(N+1)$-PC**

2. **Sublinear $\{$3,4,5$\}$PC without FHE**

Selected Results for this Talk

1. **A Framework for Sublinear $(N+1)$-PC**
   - Tools:
     - N-Party Function Secret Sharing
     - Correlated Symmetric PIR

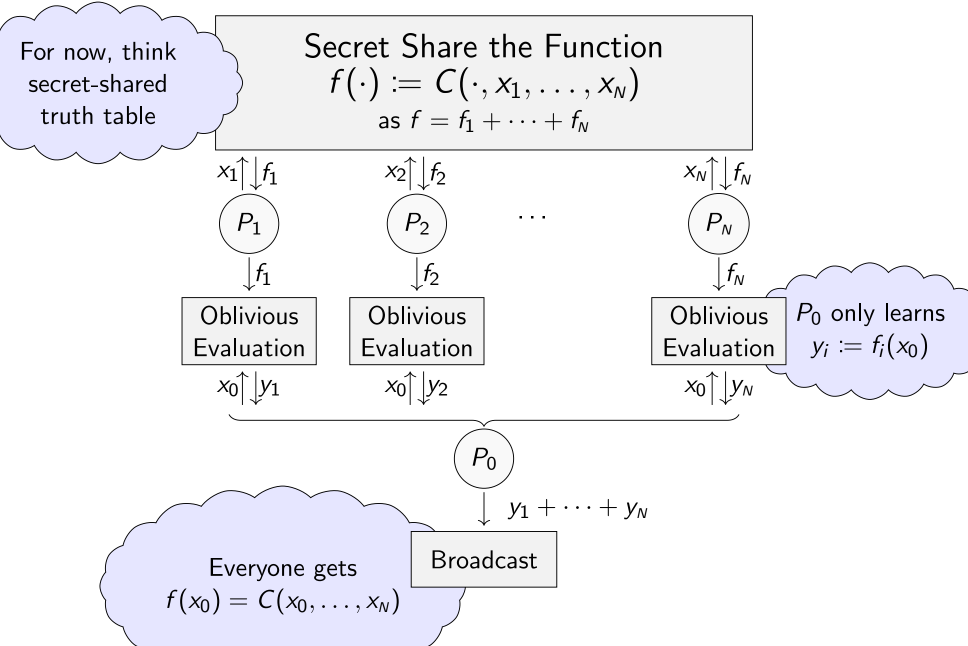
2. **Sublinear $\{$3,4,5$\}$PC without FHE**
   - Assumptions: LPN + $\{$DDH $\vee$ QR $\vee$ DCR $\vee$ poly-modulus LWE$\}$

Selected Results for this Talk

1. **A Framework for Sublinear $(N+1)$-PC**
   - Tools:
     - N-Party Function Secret Sharing
     - Correlated Symmetric PIR
   - Circuit Class:
     - Layered Circuits (Sublinear Communication)
     - LogLog-Depth Circuits ("Very Low" Comm.)

2. **Sublinear $\{3,4,5\}$PC without FHE**
   - Assumptions: LPN + $\{$DDH $\vee$ QR $\vee$ DCR $\vee$ poly-modulus LWE$\}$
   - Circuit Class:
     - Layered Circuits $\mathcal{O}(|\text{in}| + |\text{out}| + \frac{|C|}{\log\log|C|})$
     - LogLog-Depth Circuits $\mathcal{O}(|\text{in}| + |\text{out}| + \sqrt{|C|})$.

Template for $(N+1)$-Party Computation



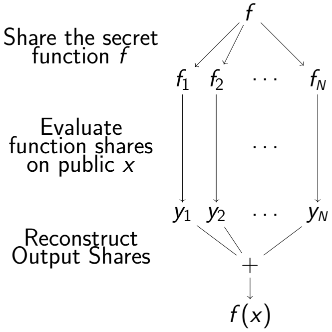
Function Secret Sharing (FSS)

[Boyle-Gilboa-Ishai'15] $\mathsf{FSS} = (\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$

Function Secret Sharing (FSS)

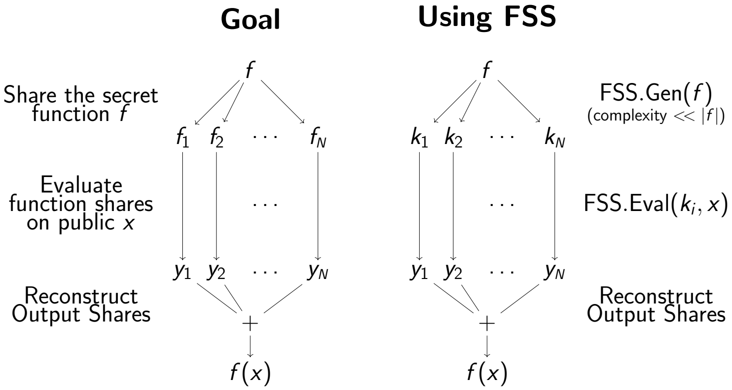
[Boyle-Gilboa-Ishai'15] $\mathsf{FSS} = (\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$

Goal



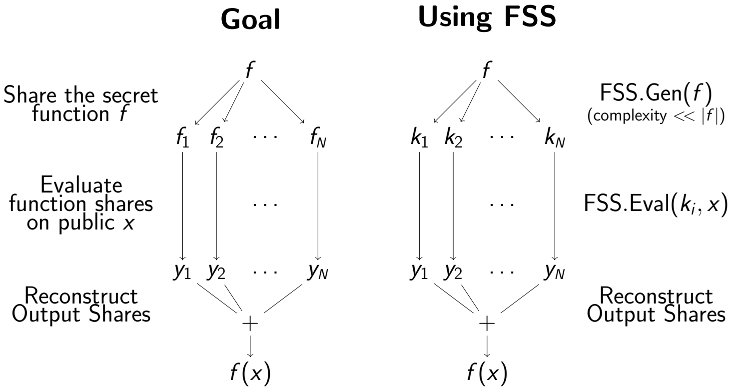
Function Secret Sharing (FSS)

[Boyle-Gilboa-Ishai'15] $\text{FSS} = (\text{FSS.Gen}, \text{FSS.Eval})$



Function Secret Sharing (FSS)

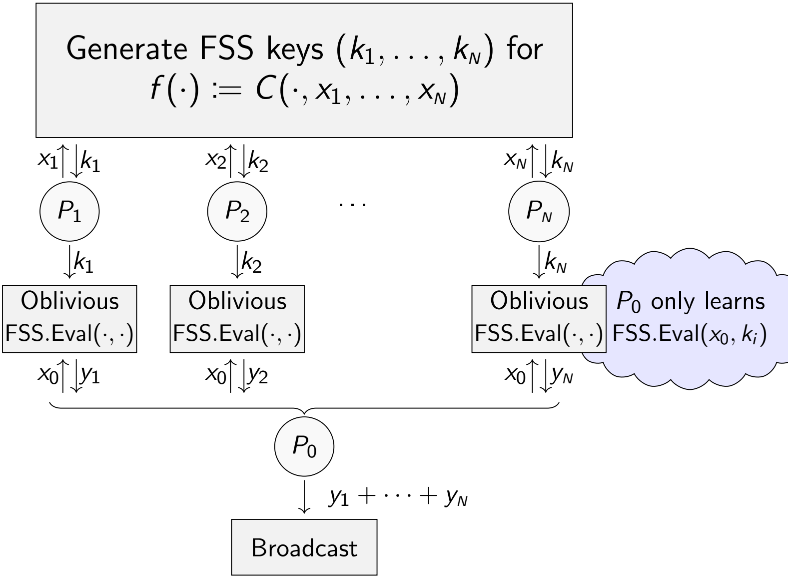
[Boyle-Gilboa-Ishai'15] $\mathsf{FSS} = (\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$



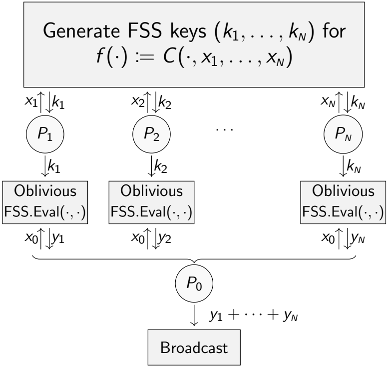
Compressing Function Shares to Short Keys

$$f_i(\cdot) \approx \mathsf{FSS.Eval}(k_i, \cdot)$$

Using FSS to compress function shares down to short keys



General Framework – The FSS Viewpoint



Properties of the N-FSS

1. Efficient Key-Distribution

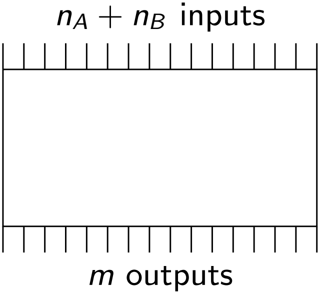
   Comm. $\mathcal{O}(|x_1| + \cdots + |x_N|)$

2. Efficient Oblivious Eval.

   Comm. $\mathcal{O}(|x_0| + |y|)$

Oblivious Evaluation from Correlated SPIR

$g(\cdot, \cdot)$

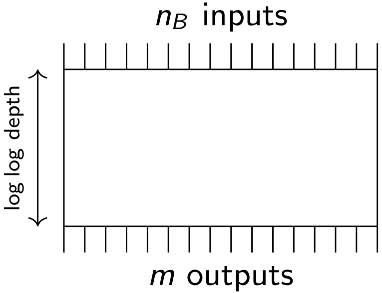


Skipping Ahead:

$g(\cdot, \cdot) = \mathsf{FSS.Eval}(\cdot, \cdot)$

Oblivious Evaluation from Correlated SPIR

$$g(x_A, \cdot)$$



Skipping Ahead:

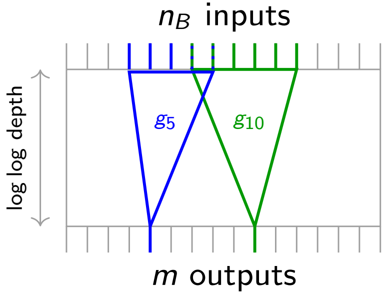
$g(x_A, \cdot) = \mathsf{FSS.Eval}(k_i, \cdot)$

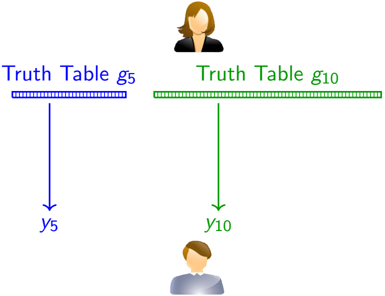
Oblivious Evaluation from Correlated SPIR

Skipping Ahead:

$g(x_A, \cdot) = \mathsf{FSS.Eval}(k_i, \cdot)$

$g(x_A, \cdot)$

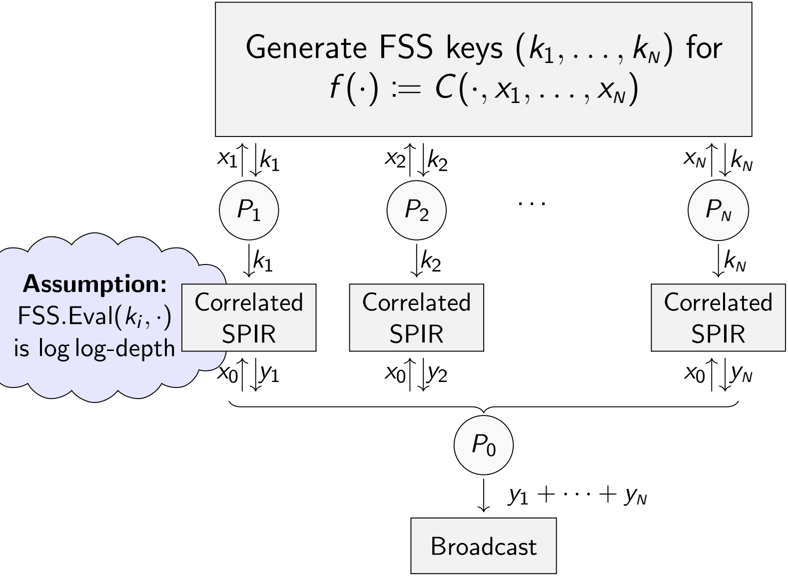




Parallel instances of specialised computation

Total Communication $\mathcal{O}(n_B + m)$ using <u>Correlated</u> SPIR

Using FSS with log log-depth Eval



Less General Framework – FSS with log log-depth Eval

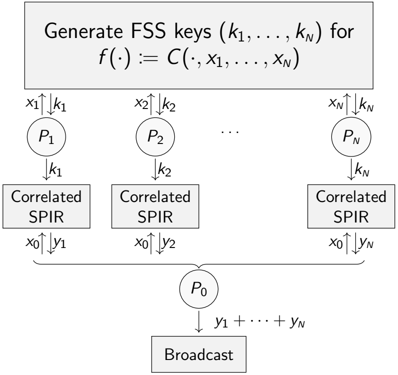
Properties of the N-FSS

1. Efficient Key-Distribution

   Comm. $\mathcal{O}(|x_1| + \cdots + |x_N|)$

2. Low-Depth Evaluation

   log log-depth FSS.Eval$(k_i, \cdot)$



Less General Framework – FSS with log log-depth Eval

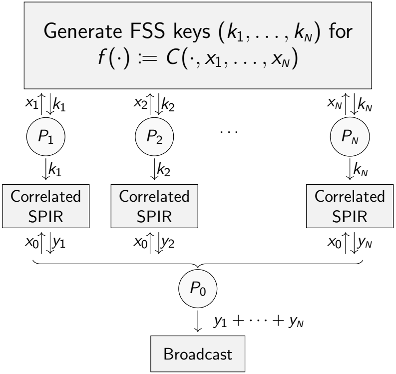
Properties of the N-FSS

1. Efficient Key-Distribution

   Comm. $\mathcal{O}(|x_1| + \cdots + |x_N|)$

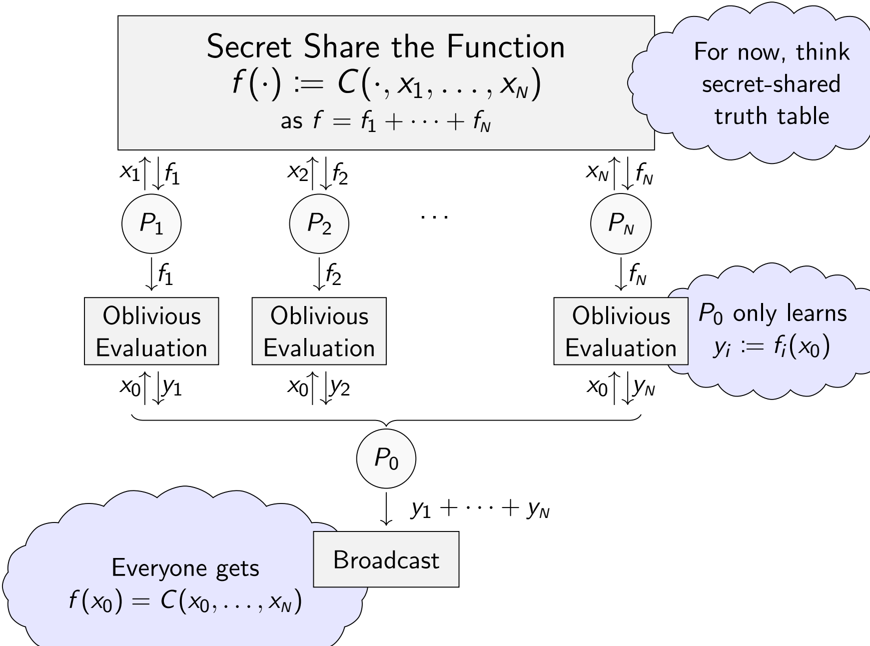
2. Low-Depth Evaluation

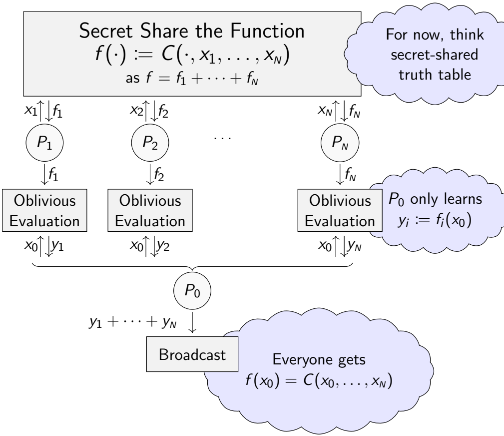
   log log-depth FSS.Eval$(k_i, \cdot)$

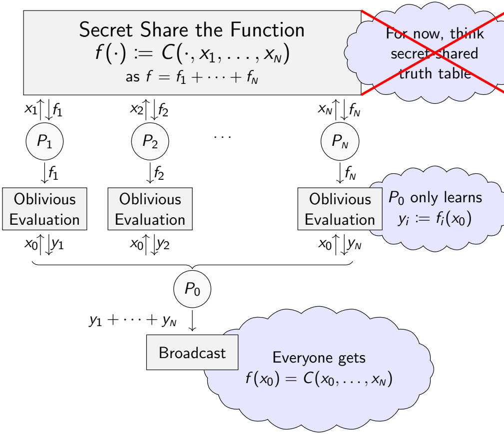


Question: Can we build such an FSS scheme?

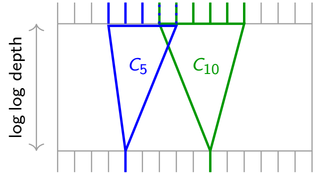
Back to the Template for $(N+1)$-Party Computation

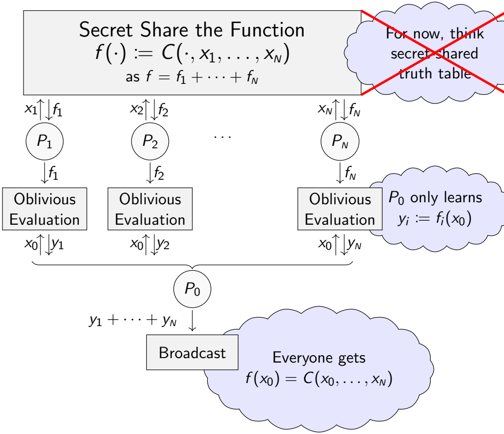



Secret Share the Function
$f(\cdot) := C(\cdot, x_1, \dots, x_N)$
as $f = f_1 + \cdots + f_N$
For now, think secret-shared truth table
x_1 f_1
x_2 f_2
x_N f_N
P_1
P_2
$\cdots$
P_N
f_1
f_2
f_N
Oblivious Evaluation
Oblivious Evaluation
Oblivious Evaluation
P_0 only learns $y_i := f_i(x_0)$
x_0 y_1
x_0 y_2
x_0 y_N
P_0
$y_1 + \cdots + y_N$
Broadcast
Everyone gets $f(x_0) = C(x_0, \dots, x_N)$

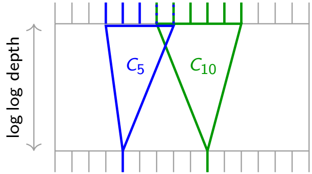

Secret Share the Function
$f(\cdot) := C(\cdot, x_1, \ldots, x_N)$
as $f = f_1 + \cdots + f_N$
For now, think secret shared truth table
x_1 f_1
x_2 f_2
x_N f_N
P_1
P_2
$\cdots$
P_N
f_1
f_2
f_N
Oblivious Evaluation
Oblivious Evaluation
Oblivious Evaluation
P_0 only learns $y_i := f_i(x_0)$
x_0 y_1
x_0 y_2
x_0 y_N
P_0
$y_1 + \cdots + y_N$
Broadcast
Everyone gets $f(x_0) = C(x_0, \ldots, x_N)$

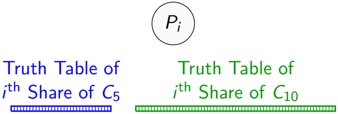
$C(\cdot, x_1, \ldots, x_N)$

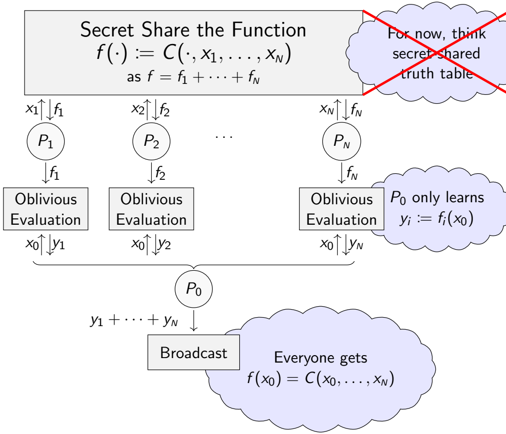


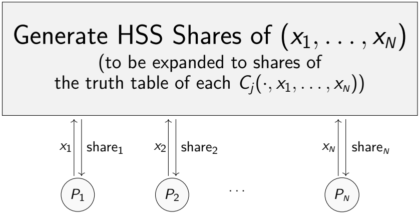


$C(\cdot, x_1, \ldots, x_N)$

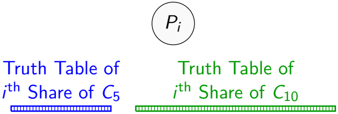


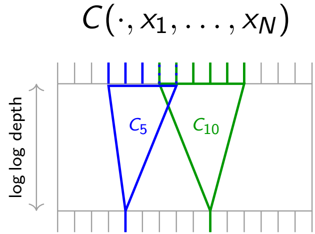


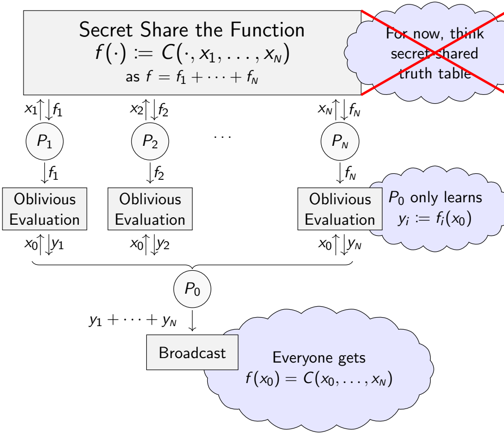


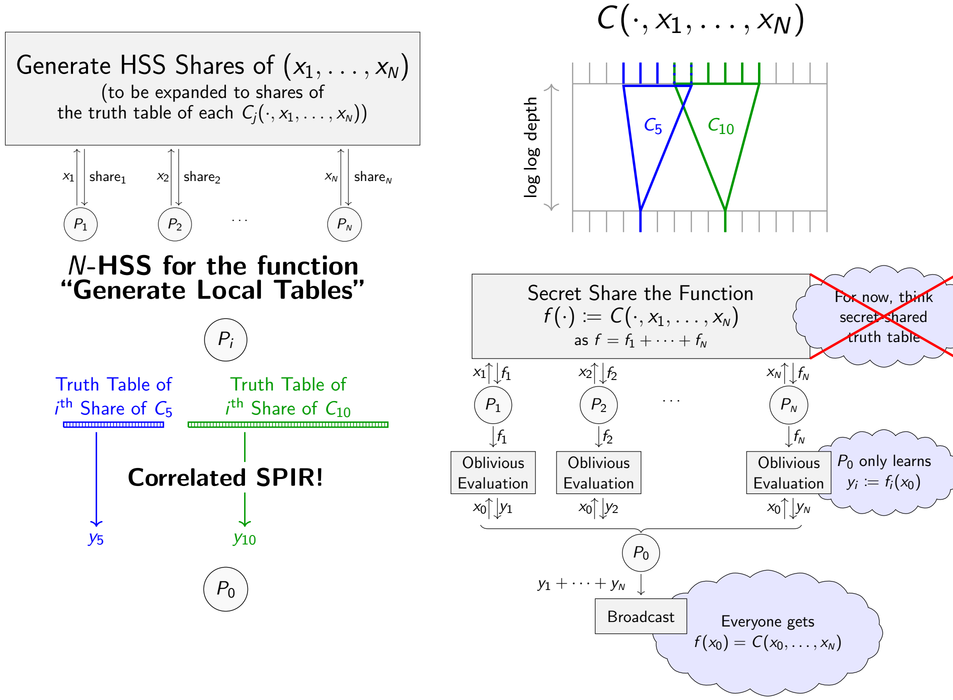


N-HSS for the function "Generate Local Tables"







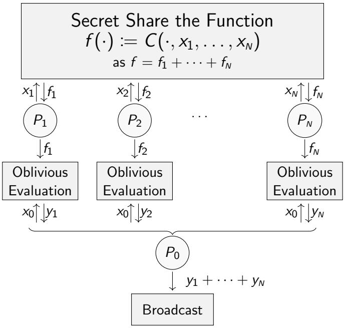


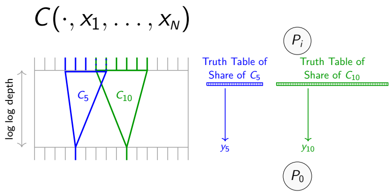
Generate HSS Shares of $(x_1, \ldots, x_N)$
(to be expanded to shares of
the truth table of each $C_j(\cdot, x_1, \ldots, x_N)$)
x_1 share$_1$
x_2 share$_2$
x_N share$_N$
P_1 P_2 $\cdots$ P_N
N-HSS for the function
"Generate Local Tables"
P_i
Truth Table of i^{th} Share of C_5
Truth Table of i^{th} Share of C_{10}
Correlated SPIR!
y_5
y_{10}
P_0
$C(\cdot, x_1, \ldots, x_N)$
log log depth
C_5
C_{10}
Secret Share the Function
$f(\cdot) := C(\cdot, x_1, \ldots, x_N)$
as $f = f_1 + \cdots + f_N$
For now, think secret shared truth table
x_1 f_1
x_2 f_2
x_N f_N
P_1 P_2 $\cdots$ P_N
f_1 f_2 f_N
Oblivious Evaluation
Oblivious Evaluation
Oblivious Evaluation
P_0 only learns $y_i := f_i(x_0)$
x_0 y_1
x_0 y_2
x_0 y_N
P_0
$y_1 + \cdots + y_N$
Broadcast
Everyone gets $f(x_0) = C(x_0, \ldots, x_N)$

1. **General Framework:**

FSS for C s.t.

1. FSS.Gen can be distributed with comm. $o(|C|)$
2. FSS.Eval can be distributed with comm. $o(|C|)$





2. **Framework for LogLog-Depth Circuits:**

1. N-party HSS for Table
   - $N = 4$: DCR
   - $N = 2$: DDH, DCR, Quasipoly-LPN, poly-modulus LWE
2. Correlated SPIR
   - LPN + {QR $\vee$ DDH $\vee$ DCR $\vee$ poly-modulus LWE}

Some Other Results

3. **Instantiating the framework with N-party "Las Vegas" HSS (+ correlated SPIR)**

4. 4-**Party HSS for any** $\log\log$-**depth circuit from DCR**

Some Other Results

3. **Instantiating the framework with N-party "Las Vegas" HSS (+ correlated SPIR)**
   - Core Issue:
     Error Propagation (most elements of each truth table are shared erroneously!)
   - Solution:
     DPF + PIR to oblivious correct errors

4. 4-**Party HSS for any** $\log\log$-**depth circuit from DCR**

Some Other Results

3. **Instantiating the framework with N-party "Las Vegas" HSS (+ correlated SPIR)**
   - Core Issue:

     Error Propagation (most elements of each truth table are shared erroneously!)
   - Solution:

     DPF + PIR to oblivious correct errors

     See the paper for the details!

4. **4-Party HSS for any $\log\log$-depth circuit from DCR**

Some Other Results

3. **Instantiating the framework with N-party "Las Vegas" HSS (+ correlated SPIR)**
   - Core Issue:

     Error Propagation (most elements of each truth table are shared erroneously!)
   - Solution:

     DPF + PIR to oblivious correct errors

     See the paper for the details!

4. **4-Party HSS for any $\log\log$-depth circuit from DCR**

   Tweak of "Scooby" [Chillotti-Orsini-Scholl-Smart-vLeeuwen'22]
   (4-Party constant-depth HSS from DCR)

Open Directions

1. Beyond $\{3,4,5\}$ parties?
   - Bottleneck: N-party HSS, or "succinct function sharing" alternative.

2. Beyond boolean circuits?
   - Bottleneck: Correlated SPIR $\hookrightarrow$ Correlated Oblivious Polynomial Evaluation (OPE)

3. Beyond loglog-depth or layered circuits?
   - Bottleneck: Our "local truth tables" approach

4. Native malicious security?

   (without communication-preserving compilers)

Thank you!