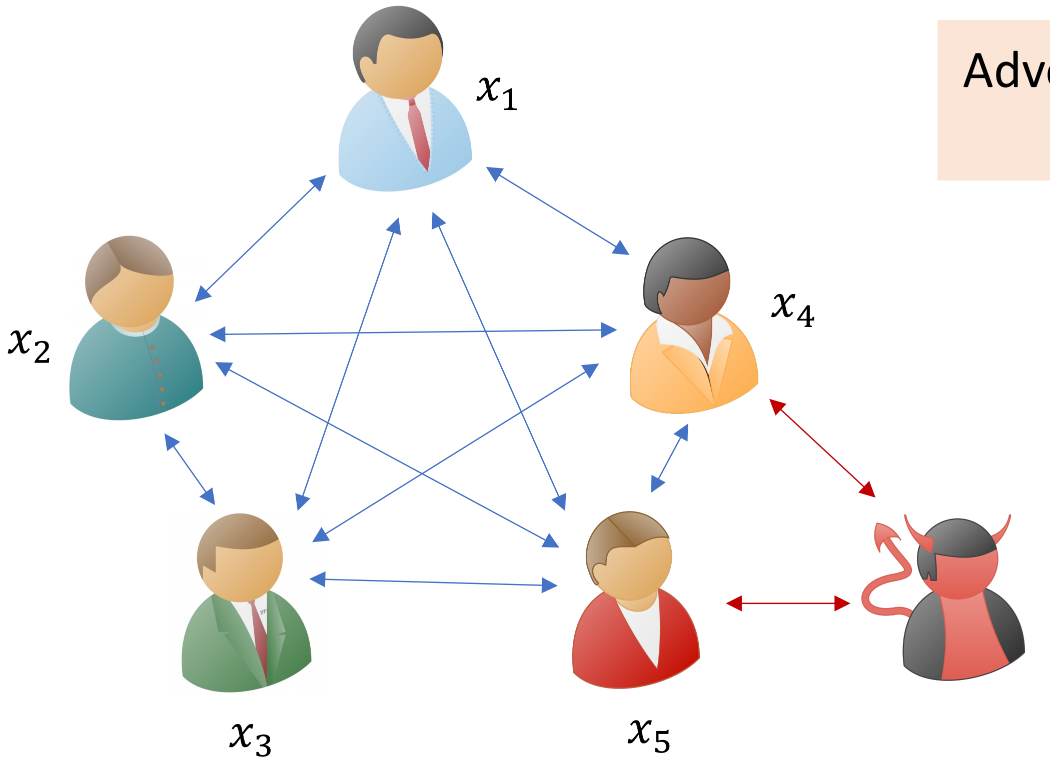
Order-C Secure Multiparty Computation for Highly Repetitive Circuits

Gabrielle Beck Aarushi Goel Abhishek Jain Gabriel Kaptchuk

JOHNS HOPKINS UNIVERSITY BOSTON UNIVERSITY

EUROCRYPT 2021

Secure Multiparty Computation

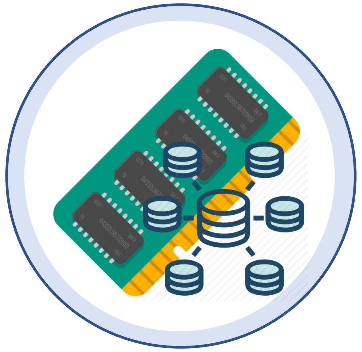


Adversary learns nothing beyond the output of the function, i.e.,
$y = f(x_1, x_2, x_3, x_4, x_5)$

MPC and Emerging Applications

- MPC protocols are becoming increasingly efficient.
- Can be used to compute complex functionalities such as:

Training machine learning algorithms on massive, distributed datasets.



Simulating large RAM programs on distributed datasets

These are large computations on massive distributed datasets

Existing Efficient and Implemented Protocols

[HN06, DN07, LN17, CGHIKLN18, NV18, FL19, GSZ20]

No. of parties

Size of circuit

$O(n|C|)$: Total computation/communication complexity

- Per-party work: $O(|C|)$
- For large computations, parties need to have large computing resources.
- Limits the kind of parties who can participate.

Better than $O(n|C|)$?

$\tilde{O}(|C|)$: Total computation and communication [DIKNS08, DIK10, GIP15]

- $\tilde{O}$ hides polynomial factors in $\log|C|$ and security parameter κ
- Not concretely efficient

$O(|C|)$: Total computation and communication [DIK10, GIP15]

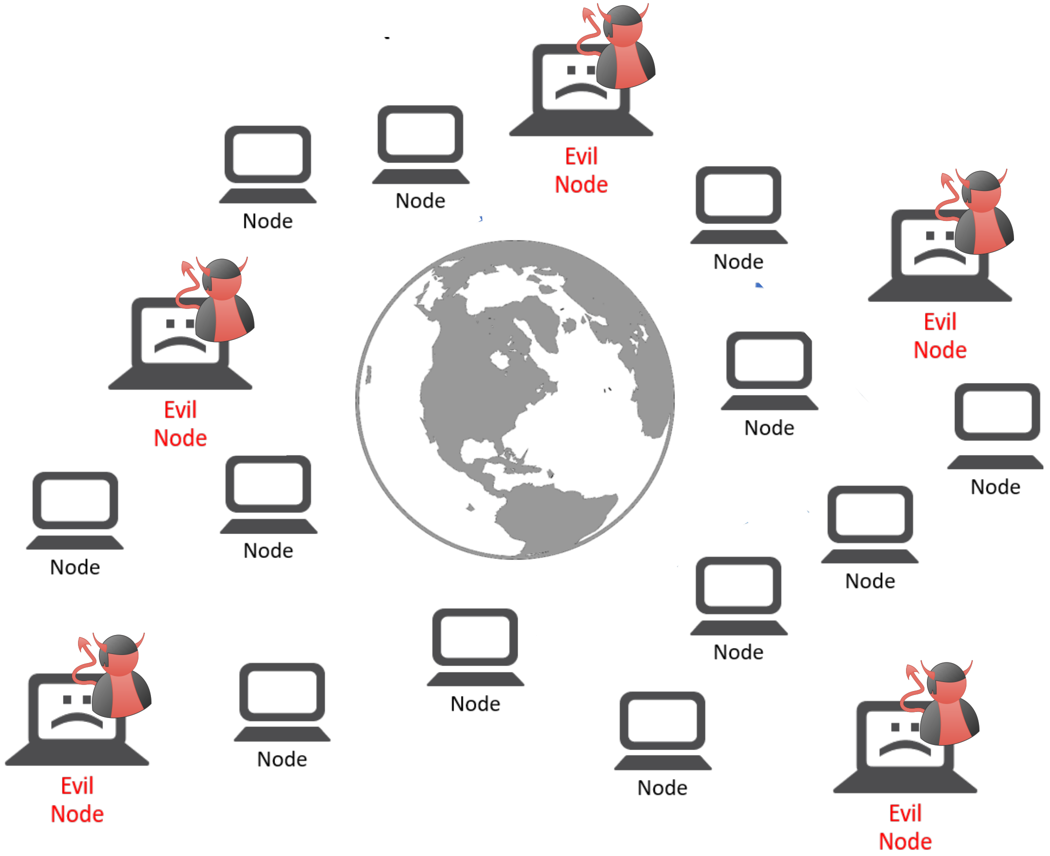
- Only for SIMD circuits
- No known implementations

Main Question:

Can we design an $O(|C|)$ MPC protocol for a larger class of circuits?

Advantages of $O(|C|)$ MPC protocols

- Can be run with many many parties
- Easier to justify honest majority
- Supports division of work
- Can be used with large volunteer networks such are Bitcoin and Tor



Could enable massive, crowd sourced applications such MPC-as-a-service

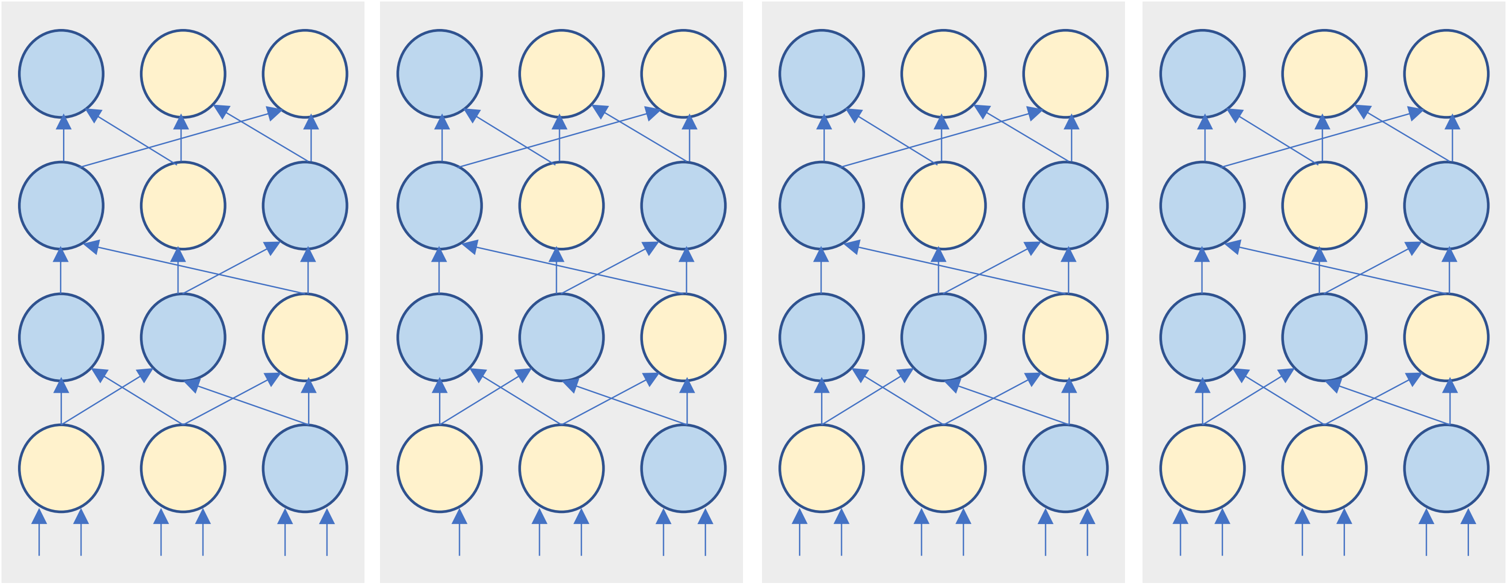
bitcoin

Our Contributions

$O(|C|)$ MPC protocol for Highly Repetitive Circuits

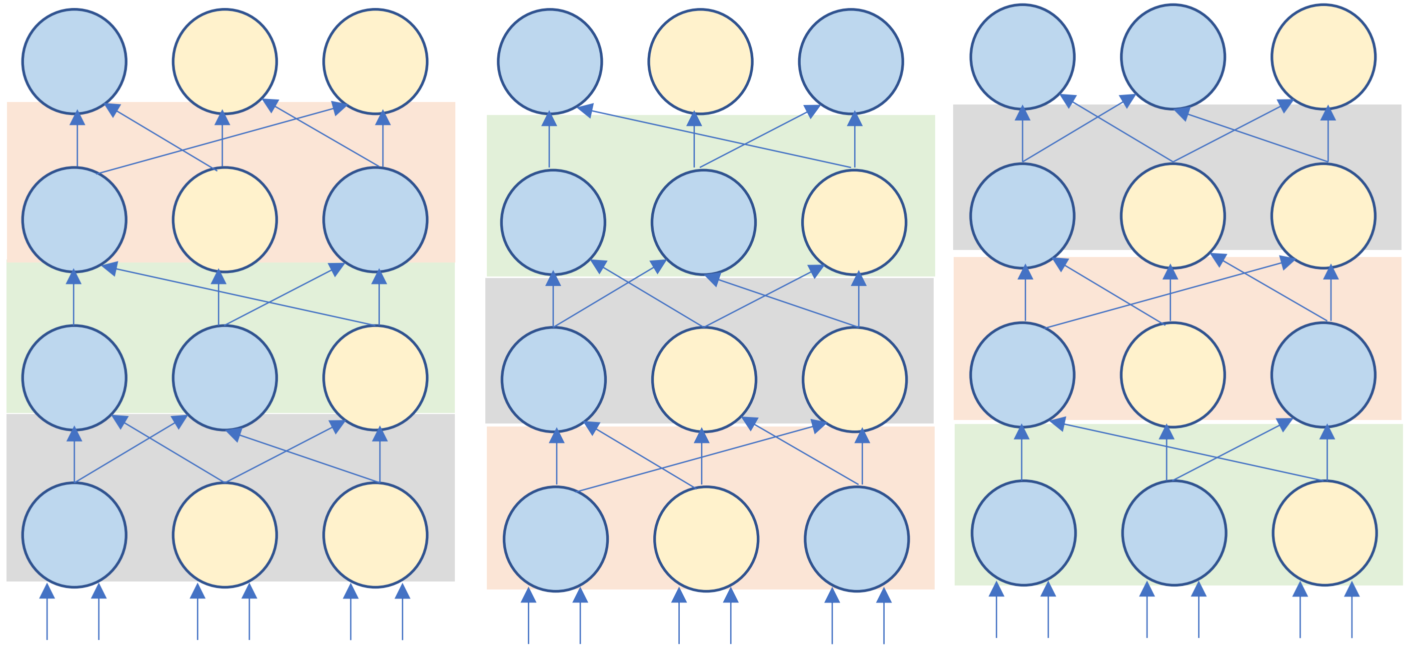
- Semi-honest and maliciously secure protocols
- $t < n\left(\frac{1}{2} - \frac{2}{\varepsilon}\right)$ static corruptions
- Information theoretic
- No setup assumptions
- Security with Abort
- Provide Implementation - first implementation of MPC that uses packed secret sharing

Single Instruction Multiple Data Circuits



Circuits that comprise of multiple parallel copies of the same sub circuit

Highly Repetitive Circuits

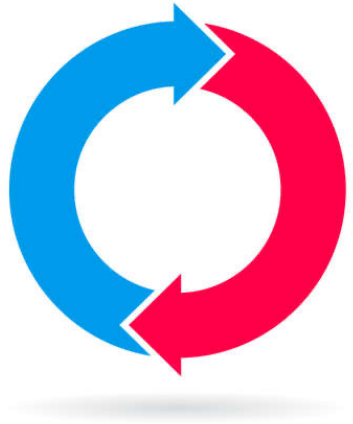


Example of (3,3)-repetitive circuit

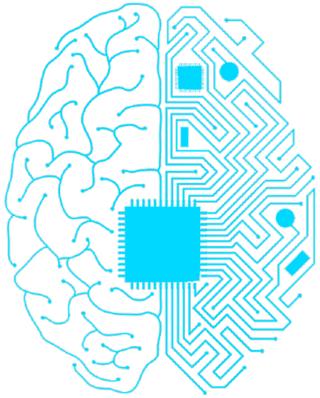
(A, B)-Repetitive Circuits: Composed of an arbitrary number of blocks of width at least A, that recur at least B times.

Highly Repetitive Circuits: (A, B)- repetitive circuit is highly repetitive w.r.t. n parties, if $A \in \Omega(n)$ and $B \in \Omega(n)$.

Examples of Highly Repetitive Circuits

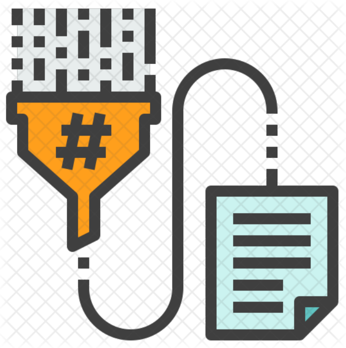


For/While Loops



Machine Learning

Block Ciphers



Cryptographic Hash Functions

Talk Outline

Background

Existing Efficient $O(n|C|)$ Protocols

Packed Secret Sharing

Existing $\tilde{O}(|C|)$ and $O(|C|)$ Protocols

Our Techniques

Existing Efficient $O(n|C|)$ Protocols

[HN06, DN07, LN17, CGHIKLN18, NV18, FL19, GSZ20]

Double sharing of random value

Multiplication (using $[r]_t$, $[r]_{2t}$)

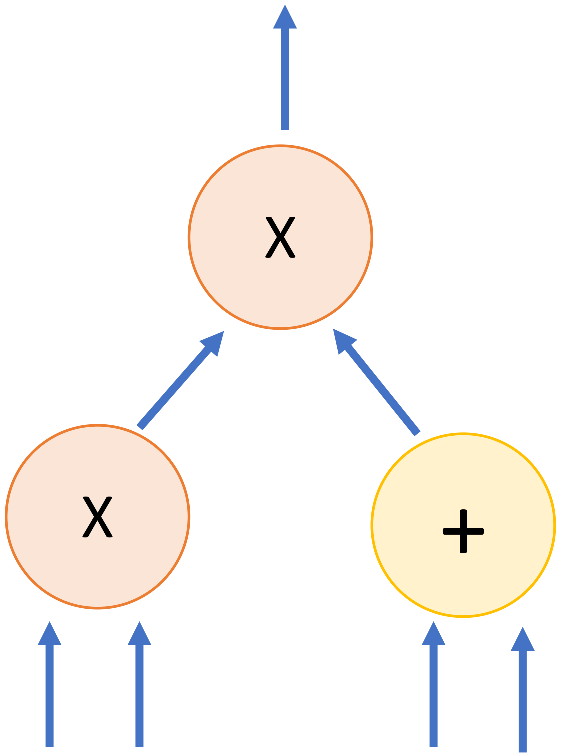
Compute $[e]_{2t} = [a]_t \times [b]_t$

$[e+r]_{2t} \leftarrow [e]_{2t} + [r]_{2t}$

Reconstruct $v = e + r$

Compute $[e]_t \leftarrow [v]_t - [r]_t$

$O(n)$ communication given $[r]_t$, $[r]_{2t}$



$[a]_t$ $[b]_t$ $[c]_t$ $[d]_t$

Input sharing: t-out-of-n shares of inputs

Gate-by-Gate Evaluation of the circuit on secret shared inputs

Addition

Compute $[f]_t = [c]_t + [d]_t$

Non-interactive

Generating $([r]_t, [r]_{2t})$ [HN06, DN07, BTH08]

s_1 s_2 s_3 s_4 s_5

Each party samples a random value and shares it using a degree t-out-of-n and $2t$-out-of-n secret sharing

$$\begin{bmatrix} [s_1]_t & [s_2]_t & [s_3]_t & [s_4]_t & [s_5]_t \\ [s_1]_{2t} & [s_2]_{2t} & [s_3]_{2t} & [s_4]_{2t} & [s_5]_{2t} \end{bmatrix} \times \text{Hyper-invertible or Vandermonde matrix} = \begin{bmatrix} [r_1]_t & [r_2]_t & [r_3]_t \\ [r_1]_{2t} & [r_2]_{2t} & [r_3]_{2t} \end{bmatrix}$$

$O(n^2)$ Communication

$n \times (n-t)$ matrix

$(n-t)$ pairs

Amortized $O(n)$ communication to generate $[r]_t, [r]_{2t}$

Existing Efficient $O(n|C|)$ Protocols

[HN06, DN07, LN17, CGHIKLN18, NV18, FL19, GSZ20]

Double sharing of random value

Multiplication (using $[r]_t$, $[r]_{2t}$)

Compute $[e]_{2t} = [a]_t \times [b]_t$

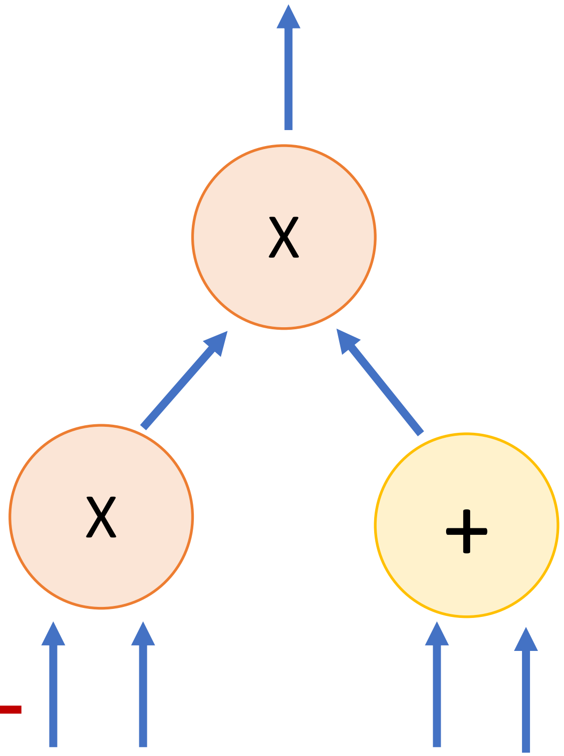
$[e + r]_{2t} \leftarrow [e]_{2t} + [r]_{2t}$

Reconstruct $v = e + r$

Compute $[e]_t \leftarrow [v]_t - [r]_t$

$O(n)$ communication ~~given~~ $\cancel{[r]_t, [r]_{2t}}$

Gate-by-Gate Evaluation of the circuit on secret shared inputs



Addition

Compute $[f]_t = [c]_t + [d]_t$

Non-interactive

$[a]_t$ $[b]_t$ $[c]_t$ $[d]_t$

Input sharing: t-out-of-n shares of inputs

Talk Outline

Background

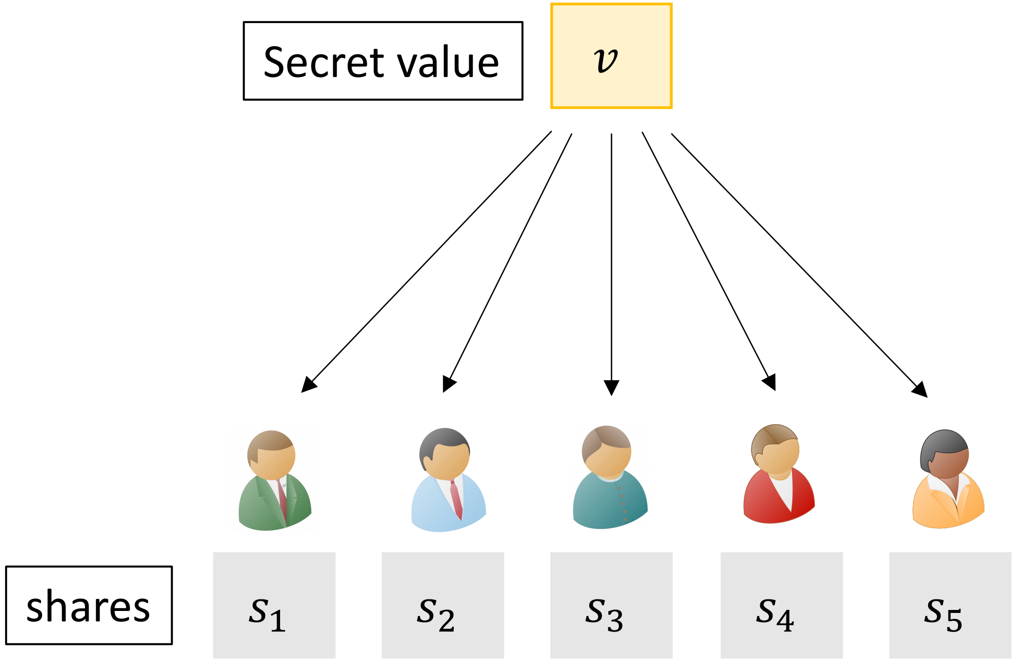
Existing Efficient $O(n|C|)$ Protocols

Packed Secret Sharing

Existing $\tilde{O}(|C|)$ and $O(|C|)$ Protocols

Our Techniques

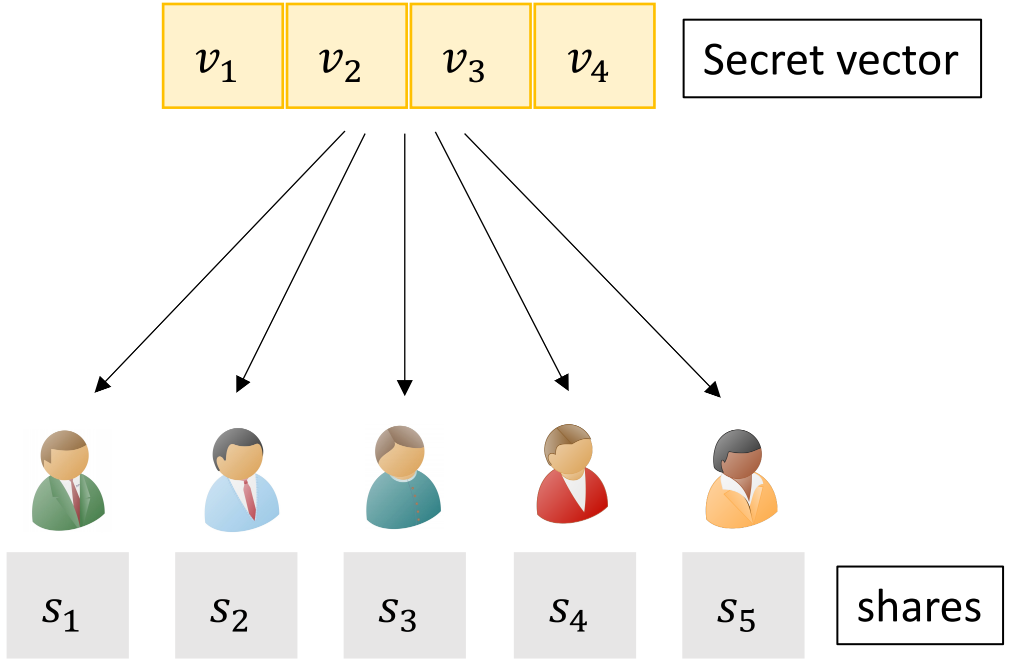
Packed Secret Sharing (PSS) [FY92]



Regular Secret Sharing

1 Value $\rightarrow$ n shares

Corruption threshold: $t < \frac{n}{2}$

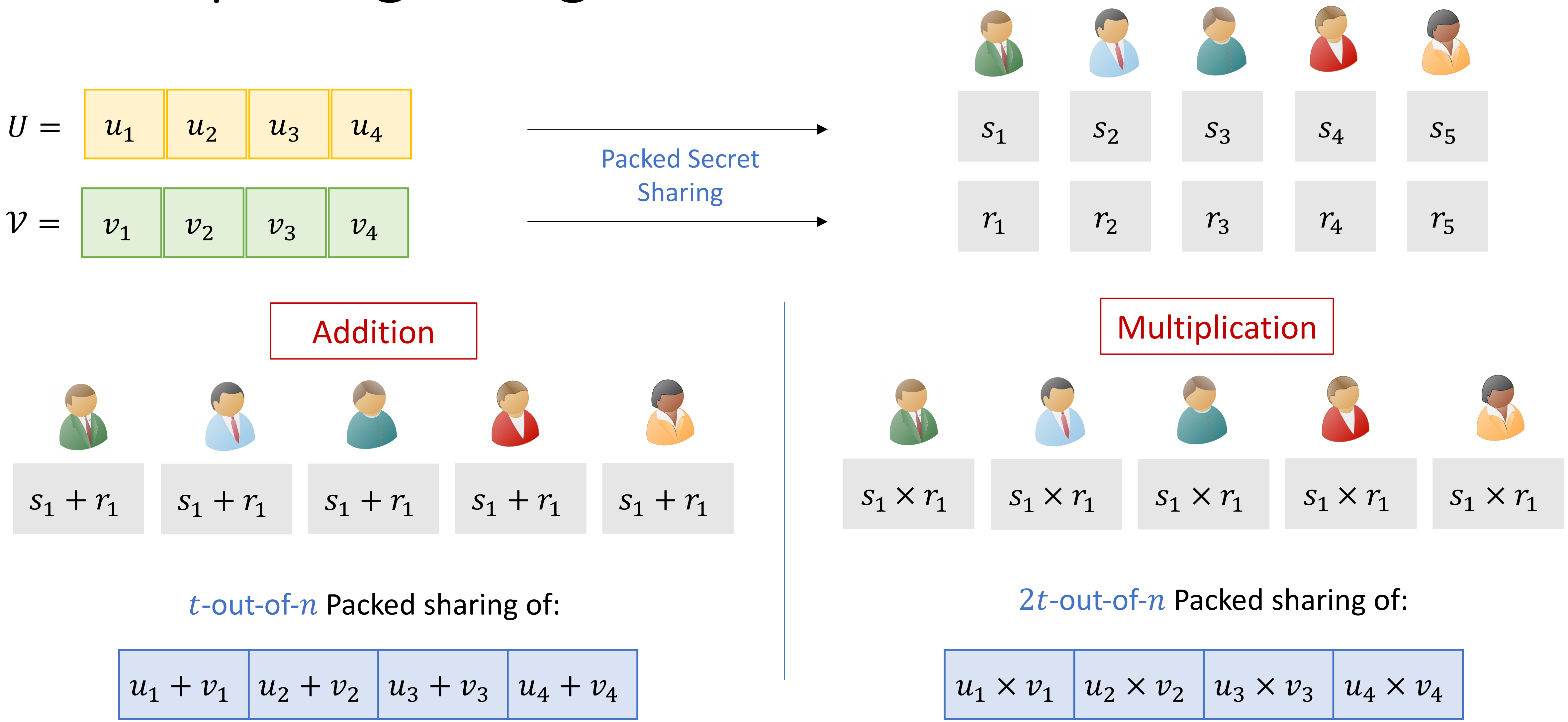


Packed Secret Sharing

$O(n)$ Values $\rightarrow$ n shares

Corruption threshold $t < n(\frac{1}{2} - \frac{1}{\varepsilon})$

Computing using PSS



Talk Outline

Background

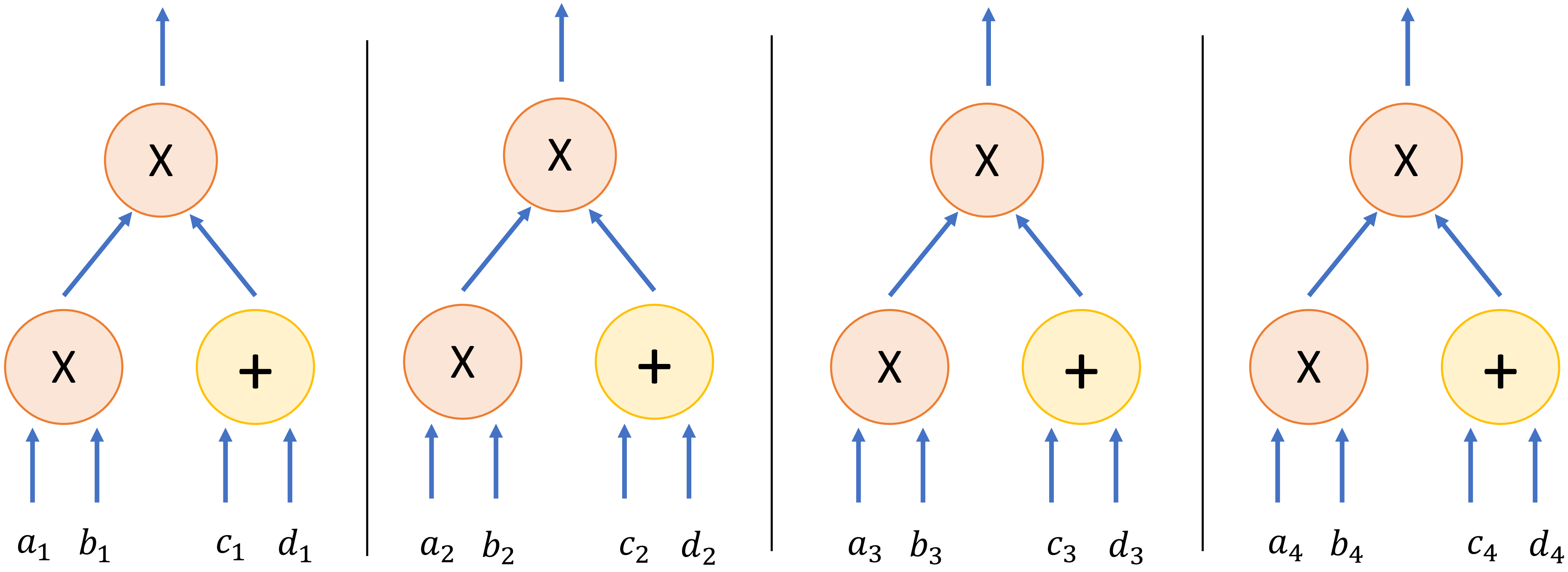
Existing Efficient $O(n|C|)$ Protocols

Packed Secret Sharing

Existing $\tilde{O}(|C|)$ and $O(|C|)$ Protocols

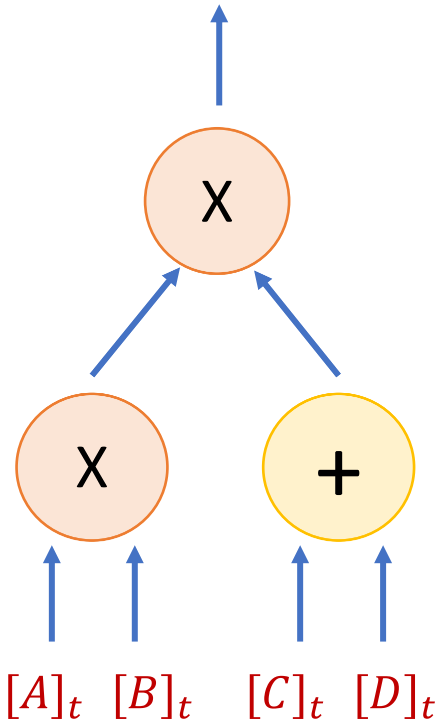
Our Techniques

Evaluating SIMD Circuits [DIK10,GIP15]



$O(n)$ copies of a sub-circuit of size $|C|$
with different inputs

Evaluating SIMD Circuits using PSS [DIK10,GIP15]



Evaluate a single instance of the sub-circuit as before but on packed shares of :

$A =$

a_1	a_2	a_3	a_4

$B =$

c_1	c_2	c_3	c_4

$C =$

b_1	b_2	b_3	b_4

$D =$

d_1	d_2	d_3	d_4

$O(n|C|)$ communication for evaluating $O(n)$ copies of a sub-circuit

Amortized $O(|C|)$ communication to evaluate a single instance of the sub-circuit

Going Beyond SIMD Circuits? [DIK10, GIP15]

Transform any given circuit into a circuit that can be used with packed secret sharing by embedding routing networks

Significantly Increases the size of the circuit to $\tilde{O}(|C|)$

Talk Outline

Background

Existing Efficient $O(n|C|)$ Protocols

+

Packed Secret Sharing

→ Existing $\tilde{O}(|C|)$ and $O(|C|)$ Protocols

Our Techniques

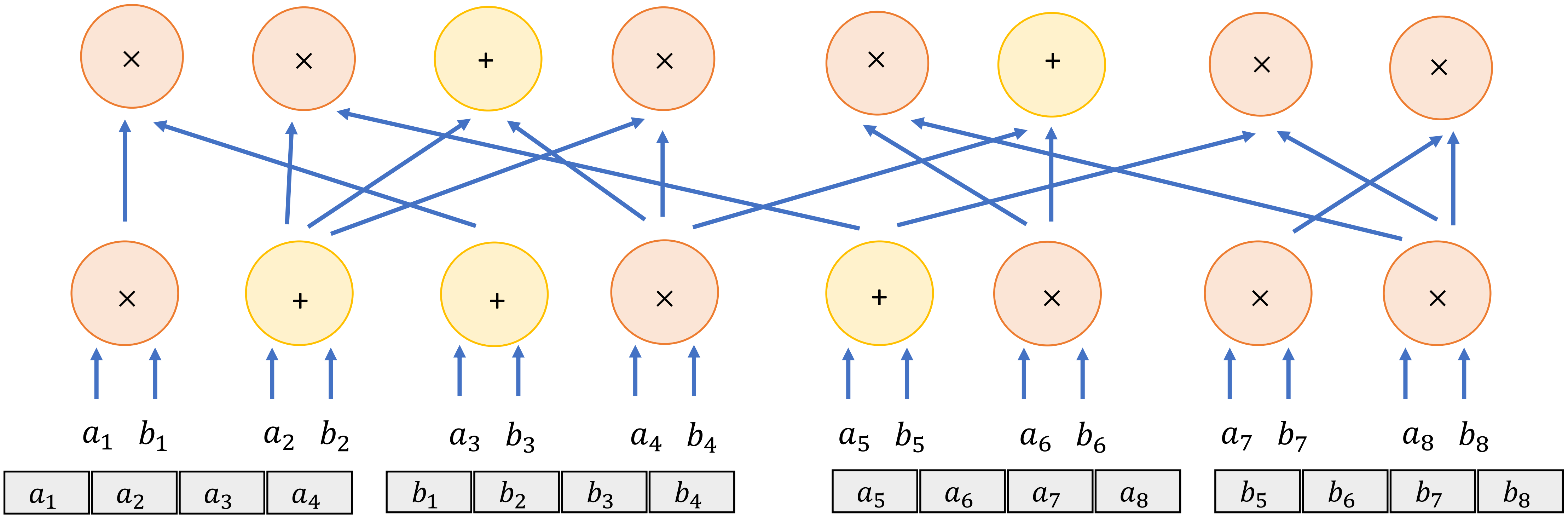
Talk Outline

Background

Our Techniques

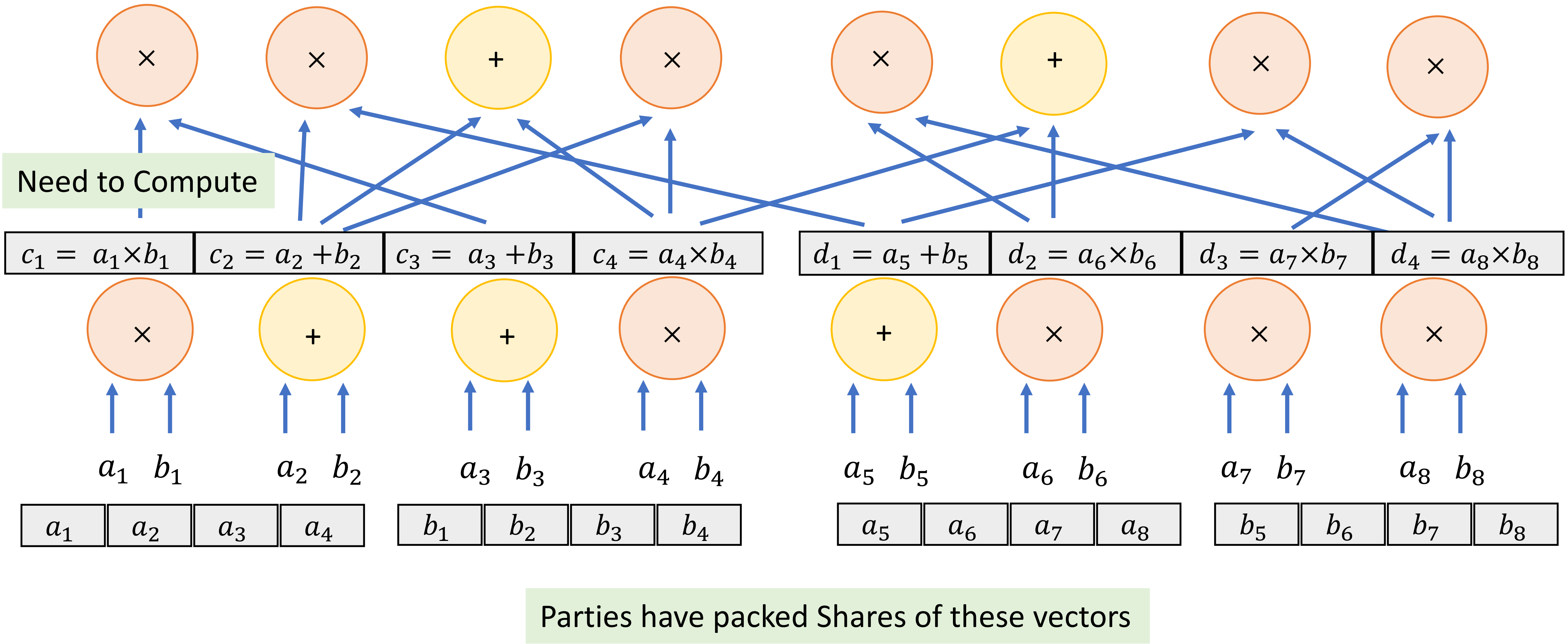
- Main Challenges
- Our Main Ideas
- Leveraging repetition in Highly Repetitive Circuits
- Malicious Security

Going Beyond SIMD without Circuit Transformation?



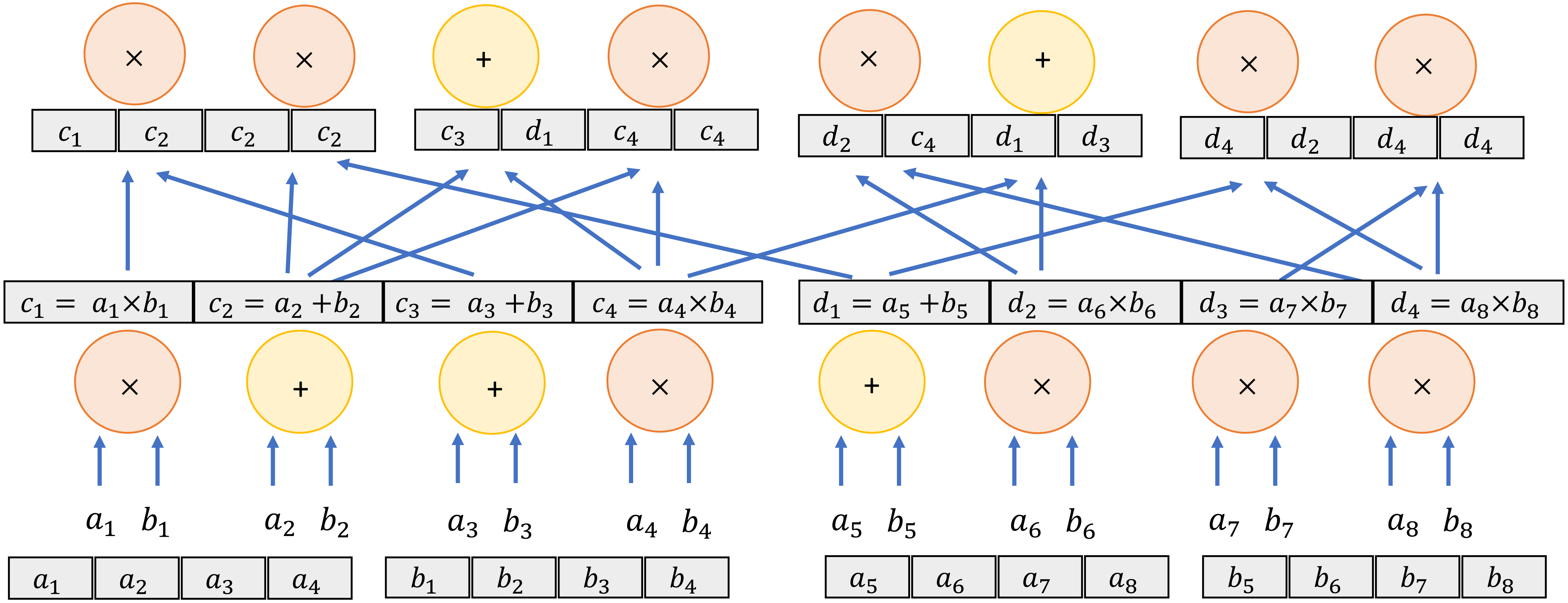
Parties have packed Shares of these vectors

Going Beyond SIMD without Circuit Transformation?



Going Beyond SIMD without Circuit Transformation?

Need these for computing the next layer



Parties have packed Shares of these vectors

Main Challenges

Given packed shares of:

$U = \boxed{u_1}\boxed{u_2}\boxed{u_3}\boxed{u_4}$

$\mathcal{V} = \boxed{v_1}\boxed{v_2}\boxed{v_3}\boxed{v_4}$

Need to compute packed sharing of:

Different Operations: $\boxed{u_1 + v_1}\boxed{u_2 \times v_2}\boxed{u_3 \times v_3}\boxed{u_4 + v_4}$

Re-aligned Vector Values: $\boxed{u_1}\boxed{v_4}\boxed{v_3}\boxed{u_2}$ $\quad$ $\boxed{v_1}\boxed{u_3}\boxed{v_2}\boxed{u_4}$

Talk Outline

- Background
- Our Techniques
  - Main Challenges
  - Our Main Ideas
    - Differing Operation PSS
    - Re-aligning PSS vectors
  - Leveraging repetition in Highly Repetitive Circuits
  - Malicious Security

Differing-operation PSS

Computing packed sharing of:

$u_1 + v_1$	$u_2 \times v_2$	$u_3 \times v_3$	$u_4 + v_4$

What are these masks?

Step 1: Reconstruct both operations

$u_1 + v_1$+ mask	$u_2 + v_2$ + mask	$u_3 + v_3$ + mask	$u_4 + v_4$ + mask
$u_1 \times v_1$+ mask	$u_2 \times v_2$ + mask	$u_3 \times v_3$ + mask	$u_4 \times v_4$ + mask

Step 2: Select and share new vector

$u_1 + v_1$+ mask	$u_2 \times v_2$ + mask	$u_3 \times v_3$ + mask	$u_4 + v_4$ + mask

Step 3: Unmask new vector

$u_1 + v_1$	$u_2 \times v_2$	$u_3 \times v_3$	$u_4 + v_4$

Differing-operation PSS

Computing packed sharing of:

$u_1 + v_1$	$u_2 \times v_2$	$u_3 \times v_3$	$u_4 + v_4$

Parties have shares of these correlated "masking" vectors

$R^{add} = [\, r_1^{add} \mid r_2^{add} \mid r_3^{add} \mid r_4^{add} \,]$

$R^{mult} = [\, r_1^{mult} \mid r_2^{mult} \mid r_3^{mult} \mid r_4^{mult} \,]$

$R = [\, r_1^{add} \mid r_2^{add} \mid r_3^{add} \mid r_4^{add} \,]$

Step 1: Reconstruct both operations

$u_1 + v_1 + r_1^{add}$	$u_2 + v_2 + r_2^{add}$	$u_3 + v_3 + r_3^{add}$	$u_4 + v_4 + r_4^{add}$
$u_1 \times v_1 + r_1^{mult}$	$u_2 \times v_2 + r_2^{mult}$	$u_3 \times v_3 + r_3^{mult}$	$u_4 \times v_4 + r_4^{mult}$

Step 2: Select and share new vector

$u_1 + v_1 + r_1^{add}$	$u_2 \times v_2 + r_2^{mult}$	$u_3 \times v_3 + r_3^{mult}$	$u_4 + v_4 + r_4^{add}$

Step 3: Unmask new vector

$u_1 + v_1$	$u_2 \times v_2$	$u_3 \times v_3$	$u_4 + v_4$

Re-aligning PSS vectors

Computing packed sharing of:

u_1	v_4	v_3	u_2

v_1	u_3	v_2	u_4

Correlated masking vectors can be chosen in a similar way

Step 1: Reconstruct both masked vectors

u_1+ mask	u_2+ mask	u_3+ mask	u_4+ mask
v_1+ mask	v_2 + mask	v_3 + mask	v_4 + mask

Step 2: Select and share new vectors

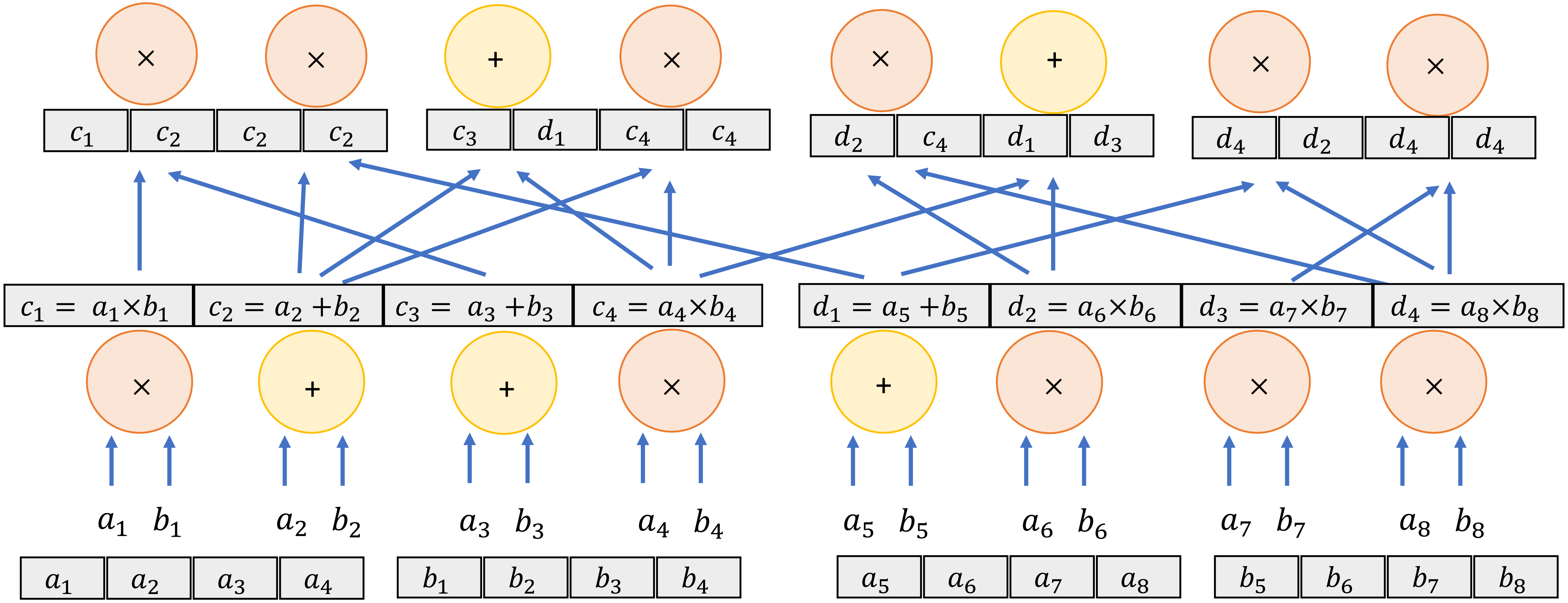
u_1 + mask	v_4 + mask	v_3 + mask	u_2 + mask
v_1+ mask	u_3 + mask	v_2 + mask	u_4+ mask

Step 3: Unmask new vectors

u_1	v_4	v_3	u_2

v_1	u_3	v_2	u_4

Going Beyond SIMD without Circuit Transformation?



Differing Operations + Re-alignment

$C_1 =$ $[\,c_1 \mid c_2 \mid c_2 \mid c_2\,]$

$D_1 =$ $[\,c_3 \mid d_1 \mid c_4 \mid c_4\,]$

$C_2 =$ $[\,d_2 \mid c_4 \mid d_1 \mid d_3\,]$

$D_2 =$ $[\,d_4 \mid d_2 \mid d_4 \mid d_4\,]$

Differing operations PSS and re-alignment process can be combined to compute this in a single step

$A_1 =$ $[\,a_1 \mid a_2 \mid a_3 \mid a_4\,]$ $B_1 =$ $[\,b_1 \mid b_2 \mid b_3 \mid b_4\,]$ $A_2 =$ $[\,a_5 \mid a_6 \mid a_7 \mid a_8\,]$ $B_2 =$ $[\,b_5 \mid b_6 \mid b_7 \mid b_8\,]$

Differing Operations + Re-alignment

$$C_1 = \begin{array}{|c|c|c|c|}\hline a_1 \times b_1 & a_2 + b_2 & a_2 + b_2 & a_2 + b_2 \\ \hline\end{array}$$

$$D_1 = \begin{array}{|c|c|c|c|}\hline a_3 + b_3 & a_5 + b_5 & a_4 \times b_4 & a_4 \times b_4 \\ \hline\end{array}$$

$$C_2 = \begin{array}{|c|c|c|c|}\hline a_6 \times b_6 & a_4 \times b_4 & a_5 + b_5 & a_7 \times b_7 \\ \hline\end{array}$$

$$D_2 = \begin{array}{|c|c|c|c|}\hline a_8 \times b_8 & a_6 \times b_6 & a_8 \times b_8 & a_8 \times b_8 \\ \hline\end{array}$$

Correlated Masking Vectors needed for this computation:

For masking $(A_1 + B_1)$ and $(A_1 \times B_1)$

$$R_1^{add} = \begin{array}{|c|c|c|c|}\hline r_{1,1}^{add} & r_{1,2}^{add} & r_{1,3}^{add} & r_{1,4}^{add} \\ \hline\end{array}$$

$$R_1^{mult} = \begin{array}{|c|c|c|c|}\hline r_{1,1}^{mult} & r_{1,2}^{mult} & r_{1,3}^{mult} & r_{1,4}^{mult} \\ \hline\end{array}$$

$$R_2^{add} = \begin{array}{|c|c|c|c|}\hline r_{2,1}^{add} & r_{2,2}^{add} & r_{2,3}^{add} & r_{2,4}^{add} \\ \hline\end{array}$$

$$R_2^{mult} = \begin{array}{|c|c|c|c|}\hline r_{2,1}^{mult} & r_{2,2}^{mult} & r_{2,3}^{mult} & r_{2,4}^{mult} \\ \hline\end{array}$$

For masking $(A_2 + B_2)$ and $(A_2 \times B_2)$

$$A_1 = \begin{array}{|c|c|c|c|}\hline a_1 & a_2 & a_3 & a_4 \\ \hline\end{array} \quad B_1 = \begin{array}{|c|c|c|c|}\hline b_1 & b_2 & b_3 & b_4 \\ \hline\end{array}$$

$$A_2 = \begin{array}{|c|c|c|c|}\hline a_5 & a_6 & a_7 & a_8 \\ \hline\end{array} \quad B_2 = \begin{array}{|c|c|c|c|}\hline b_5 & b_6 & b_7 & b_8 \\ \hline\end{array}$$

Differing Operations + Re-alignment

$C_1 = [\,a_1 \times b_1 \mid a_2 + b_2 \mid a_2 + b_2 \mid a_2 + b_2\,]$

$D_1 = [\,a_3 + b_3 \mid a_5 + b_5 \mid a_4 \times b_4 \mid a_4 \times b_4\,]$

$C_2 = [\,a_6 \times b_6 \mid a_4 \times b_4 \mid a_5 + b_5 \mid a_7 \times b_7\,]$

$D_2 = [\,a_8 \times b_8 \mid a_6 \times b_6 \mid a_8 \times b_8 \mid a_8 \times b_8\,]$

Correlated Masking Vectors needed for this computation:

For unmasking C_1 and D_1

$R_1^{left} = [\,r_{1,1}^{mult} \mid r_{1,2}^{add} \mid r_{1,3}^{add} \mid r_{1,4}^{add}\,]$

$R_1^{right} = [\,r_{1,3}^{add} \mid r_{2,1}^{add} \mid r_{1,4}^{mult} \mid r_{1,4}^{mult}\,]$

For unmasking C_2 and D_2

$R_2^{left} = [\,r_{2,2}^{mult} \mid r_{1,4}^{mult} \mid r_{2,1}^{add} \mid r_{2,3}^{mult}\,]$

$R_2^{right} = [\,r_{2,4}^{mult} \mid r_{2,2}^{mult} \mid r_{2,4}^{mult} \mid r_{2,4}^{mult}\,]$

$R_1^{add} = [\,r_{1,1}^{add} \mid r_{1,2}^{add} \mid r_{1,3}^{add} \mid r_{1,4}^{add}\,]$

$R_2^{add} = [\,r_{2,1}^{add} \mid r_{2,2}^{add} \mid r_{2,3}^{add} \mid r_{2,4}^{add}\,]$

$R_1^{mult} = [\,r_{1,1}^{mult} \mid r_{1,2}^{mult} \mid r_{1,3}^{mult} \mid r_{1,4}^{mult}\,]$

$R_2^{mult} = [\,r_{2,1}^{mult} \mid r_{2,2}^{mult} \mid r_{2,3}^{mult} \mid r_{2,4}^{mult}\,]$

$A_1 = [\,a_1 \mid a_2 \mid a_3 \mid a_4\,]$ $B_1 = [\,b_1 \mid b_2 \mid b_3 \mid b_4\,]$ $A_2 = [\,a_5 \mid a_6 \mid a_7 \mid a_8\,]$ $B_2 = [\,b_5 \mid b_6 \mid b_7 \mid b_8\,]$

Generating Correlated Masking Vectors

Correlation between masking vectors depends on the topology of individual layers in the circuit

$$R_1^{left} = \boxed{r_{1,1}^{mult}}\boxed{r_{1,2}^{add}}\boxed{r_{1,3}^{add}}\boxed{r_{1,4}^{add}}$$

$$R_1^{right} = \boxed{r_{1,3}^{add}}\boxed{r_{2,1}^{add}}\boxed{r_{1,4}^{mult}}\boxed{r_{1,4}^{mult}}$$

$$R_1^{add} = \boxed{r_{1,1}^{add}}\boxed{r_{1,2}^{add}}\boxed{r_{1,3}^{add}}\boxed{r_{1,4}^{add}}$$

$$R_1^{mult} = \boxed{r_{1,1}^{mult}}\boxed{r_{1,2}^{mult}}\boxed{r_{1,3}^{mult}}\boxed{r_{1,4}^{mult}}$$

$$R_2^{left} = \boxed{r_{2,2}^{mult}}\boxed{r_{1,4}^{mult}}\boxed{r_{2,1}^{add}}\boxed{r_{2,3}^{mult}}$$

$$R_2^{right} = \boxed{r_{2,4}^{mult}}\boxed{r_{2,2}^{mult}}\boxed{r_{2,4}^{mult}}\boxed{r_{2,4}^{mult}}$$

$$R_2^{add} = \boxed{r_{2,1}^{add}}\boxed{r_{2,2}^{add}}\boxed{r_{2,3}^{add}}\boxed{r_{2,4}^{add}}$$

$$R_2^{mult} = \boxed{r_{2,1}^{mult}}\boxed{r_{2,2}^{mult}}\boxed{r_{2,3}^{mult}}\boxed{r_{2,4}^{mult}}$$

Generating Correlated Marking Vectors

Parties sample random vectors and compute shares of correlated vectors based on the topology of a layer.

$O(n^2)$ Communication

Multiply shares of these sets of correlated vectors with a Hyper-invertible or Vandermonde matrix

$n \times (n-t)$ matrix

Get $(n-t)$ sets of correlated random vectors that all have the same correlation

$(n-t)$ sets

$O(n^2)$ communication to generate $(n-t)$ sets of correlated random vectors of length $O(n)$

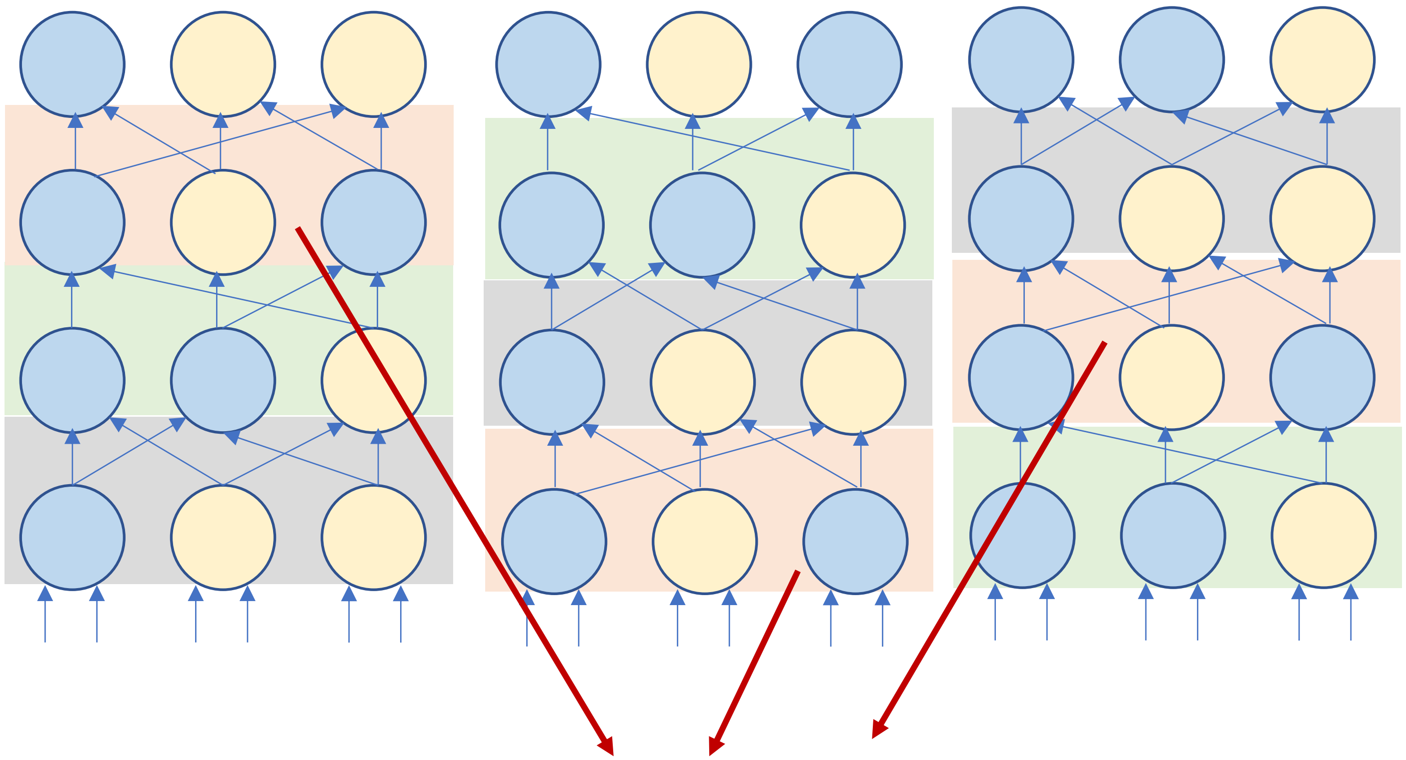
Amortized $O(n)$ communication to generate 1 set of correlated random vectors of length $O(n)$

Each such set is used to evaluate $O(n)$ gates $\Longrightarrow O(1)$ communication per gate

Talk Outline

- Background
- Our Techniques
  - Main Challenges
  - Our Main Ideas
    - Differing Operation PSS
    - Re-aligning PSS vectors
  - Leveraging repetition in Highly Repetitive Circuits
  - Malicious Security

Generating Correlated Masking Vectors for Highly Repetitive Circuits



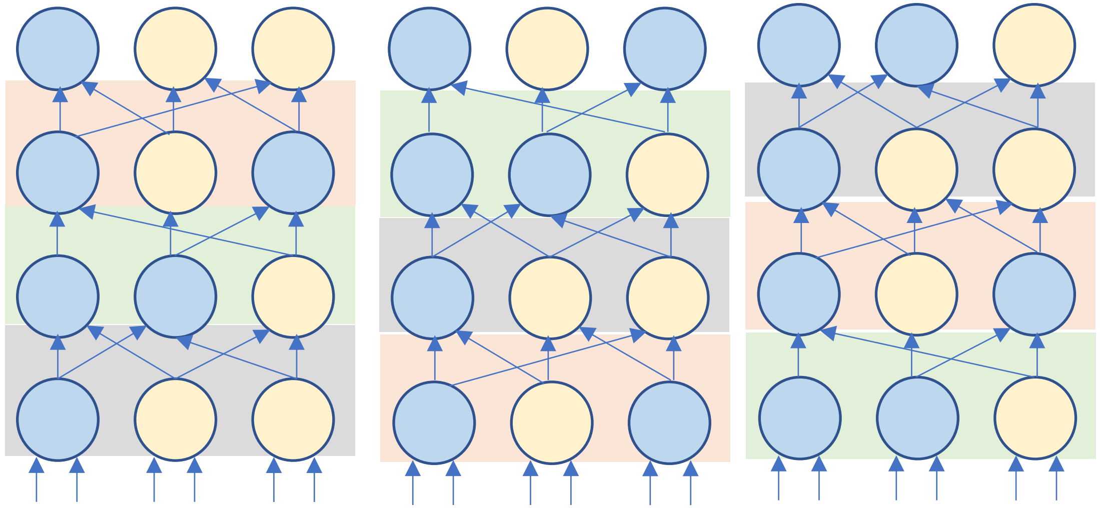
Will use same correlation between masking vectors

To get $O(|C|)$ total communication, each such block must be repeated at least $O(n)$ times and have least $O(n)$ gates

Summary So Far

Generate packed shares of correlated masking vectors for each unique block configuration using batched generation

Use these correlated masking vectors to evaluate blocks of gates using differing operation PSS + re-alignment over packed secret shared vectors



Talk Outline

- Background
- Our Techniques
  - Main Challenges
  - Our Main Ideas
    - Differing Operation PSS
    - Re-aligning PSS vectors
  - Leveraging repetition in Highly Repetitive Circuits
  - Malicious Security

Malicious Security

[GIP15]: Most packed secret sharing based semi-honest protocols are secure against malicious adversaries up to linear attacks.

Existing compilers [GIP15, CGHIKLN18] add malicious security by running multiple instances of the semi-honest protocol and comparing the outputs.

Similar to [CGHIKLN18], our protocol can be made maliciously secure by running two copies of the semi-honest version and comparing the outputs.

Conclusion

$O(|C|)$ MPC protocols for Highly Repetitive Circuits

- Semi-honest and maliciously secure protocols
- $t < n\left(\frac{1}{2} - \frac{2}{\varepsilon}\right)$ static corruptions
- Information theoretic
- No setup assumptions
- Security with Abort
- Provide Implementation - first implementation of MPC that uses packed secret sharing
- Also introduce a new non-interactive share conversion: Regular shares $\longrightarrow$ Packed shares

Thank You!