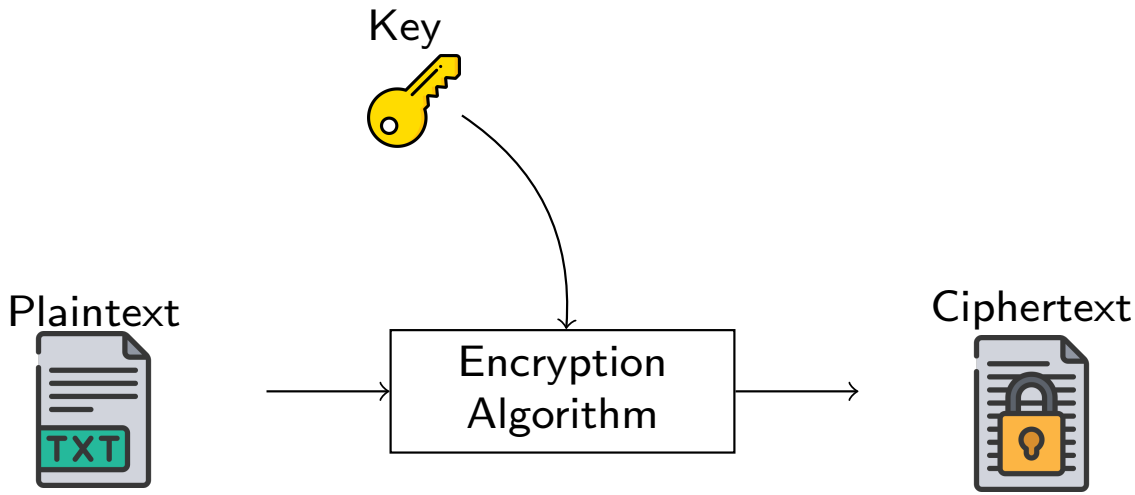
Masked Circuit Compiler in the Cardinal Random Probing Composability Framework

Sonia Belaïd, Victor Normand, Matthieu Rivain

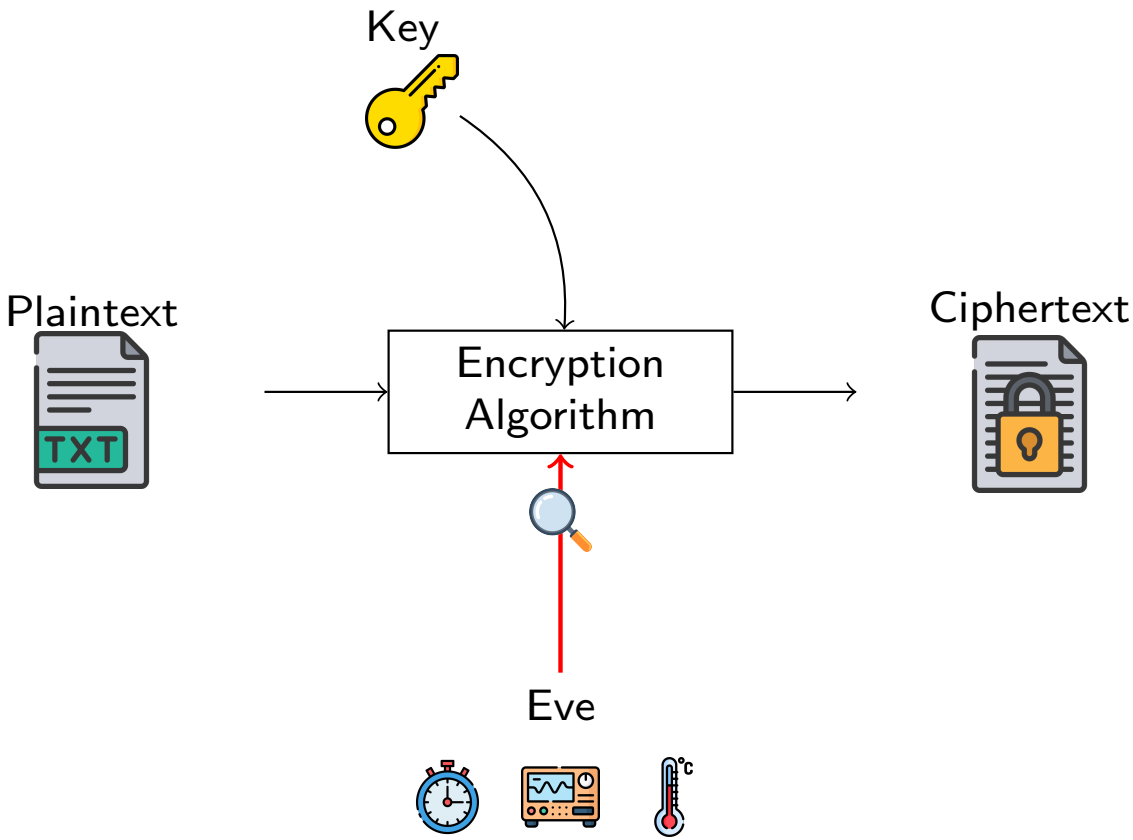
CryptoExperts

December 7, 2025

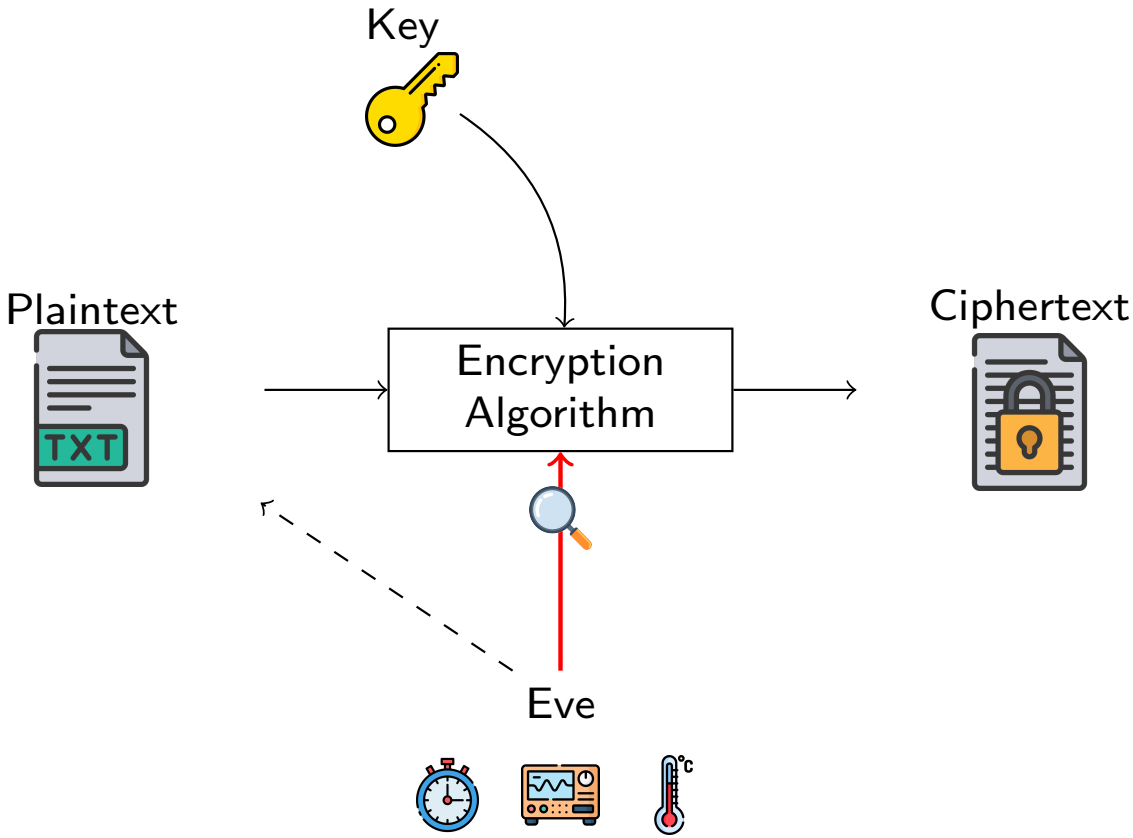
Side Channel Attacks



Side Channel Attacks



Side Channel Attacks



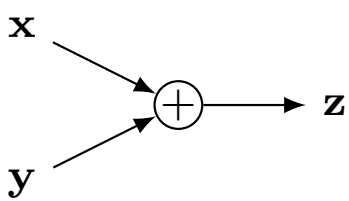
Masking countermeasure

- Replace a sensitive value $\mathbf{x}$ of the Algorithm by a tuple $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.
- Any set of $(n-1)$ shares is independent from $\mathbf{x}$.
- Arithmetic masking :

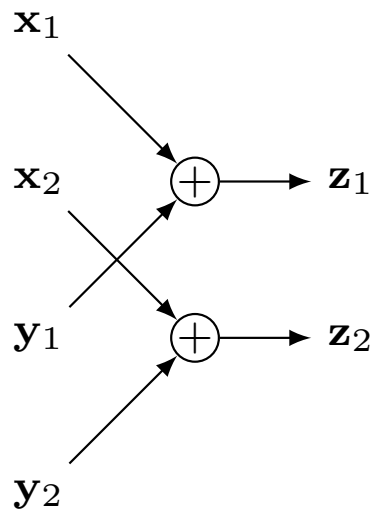
$$\mathbf{x} = \sum_{i=1}^{n} \mathbf{x}_i$$

Gadget

- Operations on sensitives values $\Longrightarrow$ Operations on their respective shares.

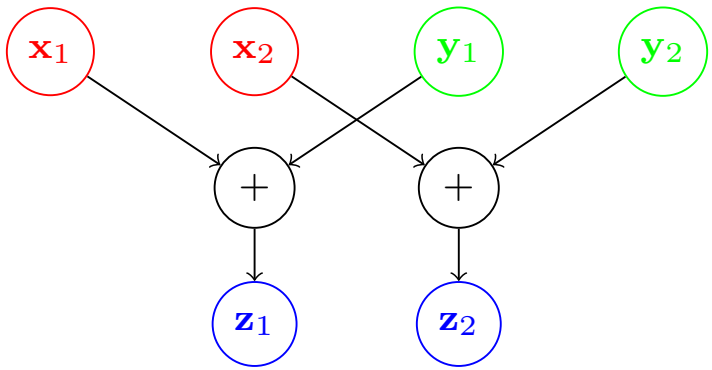


(a) Non masked version



(b) Masked version with 2 shares.

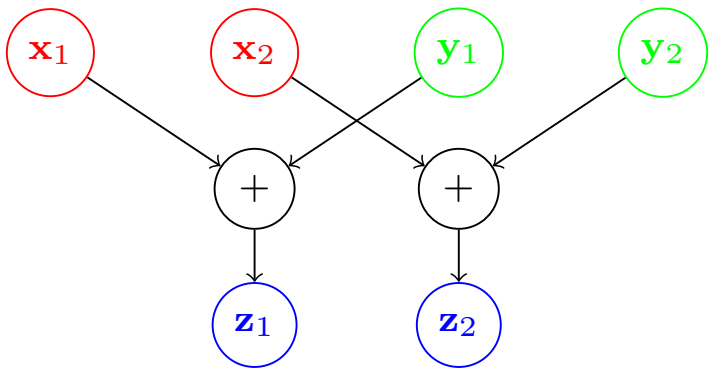
Circuits



Circuits

- Directed acyclic graph.
- Vertex : Arithmetic gate *or* input/output share.
- Edge : Wire of the circuit.

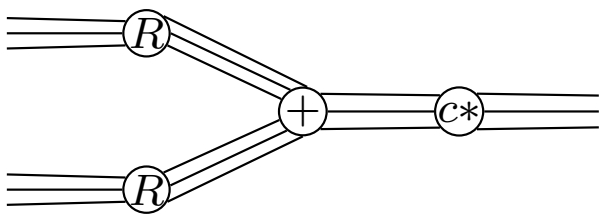
(p,ε)-Random probing model



(p, ε)-Random probing model

- Every wires leak its value with a probability $p \in [0, 1]$.
- Secure if there is a negligible probability ($\leq \varepsilon$) to get information on the secrets.

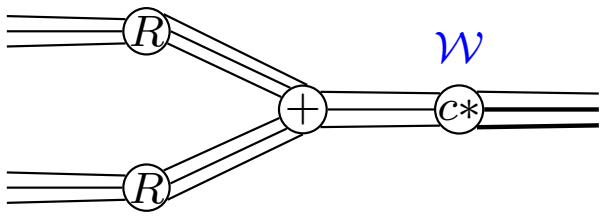
Threshold Random Probing Composability



(t, p, ε)-RPC :

- **Simulate** the internal leakage $\mathcal{W}$ and t output shares with **at most** t input shares of each secret variable.
- Error probability ε.

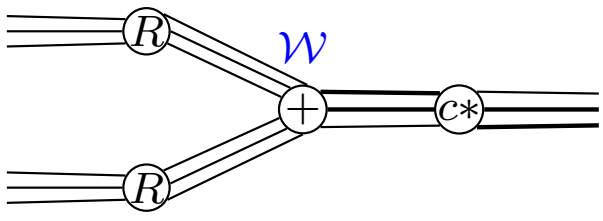
Threshold Random Probing Composability



(t, p, ε)-RPC :

- **Simulate** the internal leakage $\mathcal{W}$ and t output shares with **at most** t input shares of each secret variable.
- Error probability ε.

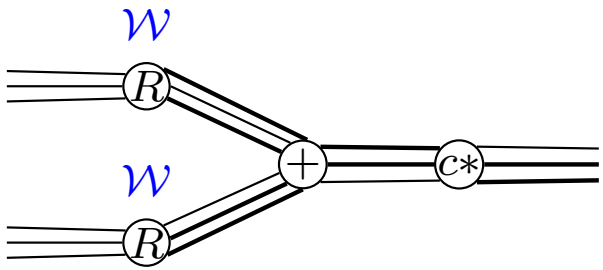
Threshold Random Probing Composability



(t, p, ε)-RPC :

- **Simulate** the internal leakage $\mathcal{W}$ and t output shares with **at most** t input shares of each secret variable.
- Error probability ε.

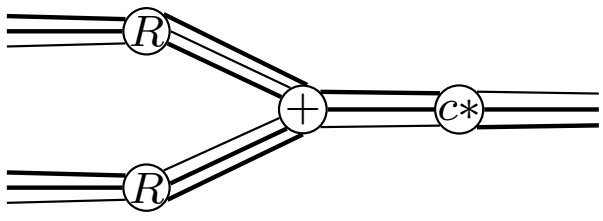
Threshold Random Probing Composability



(t, p, ε)-RPC :

- **Simulate** the internal leakage $\mathcal{W}$ and t output shares with **at most** t input shares of each secret variable.
- Error probability ε.

Threshold Random Probing Composability

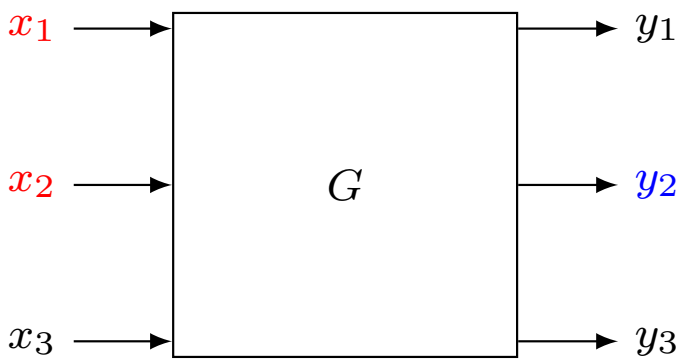


(t, p, ε)-RPC :

- **Simulate** the internal leakage $\mathcal{W}$ and t output shares with **at most** t input shares of each secret variable.
- Error probability ε.

General Random Probing Composability

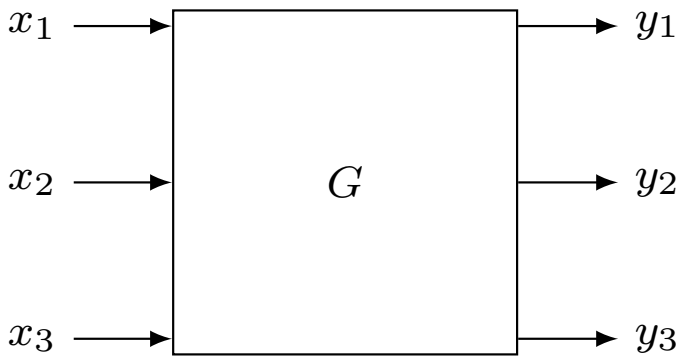
- **Probe Distribution Table :** $\varepsilon_J(I) =$ Upper bound on the probability that we requires the set I of input shares to simulate the internal leakages and the output shares in a set J.
- **Example :**



I \ J	$\emptyset$	$\{y_1\}$	$\{y_2\}$	$\{y_3\}$	$\{y_1, y_2\}$	$\{y_1, y_3\}$	$\{y_2, y_3\}$	$\{y_1, y_2, y_3\}$
$\emptyset$								
$\{x_1\}$								
$\{x_2\}$								
$\{x_3\}$								
$\{x_1, x_2\}$								
$\{x_1, x_3\}$								
$\{x_2, x_3\}$								
$\{x_1, x_2, x_3\}$								

Cardinal Random Probing Composability

- $\varepsilon_{t_{out}}(t_{in})$ = Upper Bound on the probability that we requires t_{in} input shares to simulate the internal leakages and t_{out} output shares.



t_{in} \ t_{out}	0	1	2	3
0				
1				
2				
3				

Tradeoffs between the security notions

- Tighter security Bounds : General RPC $>$ Cardinal RPC $>$ Threshold RPC.
- Verification Complexity : General RPC $<$ Cardinal RPC $<$ Threshold RPC.

Uniform Cardinal Random Probing Composability

- Best of both world : Tight security from General RPC and complexity from Cardinal RPC.
- One Condition :

$$\varepsilon^{g}_{(J_1,\ldots,J_m)}(I_1,\ldots,I_\ell) = \varepsilon^{g}_{(J'_1,\ldots,J'_m)}(I'_1,\ldots,I'_\ell)$$

when

$$(|I_1|,\ldots,|I_\ell|) = (|I'_1|,\ldots,|I'_\ell|)$$
$$(|J_1|,\ldots,|J_m|) = (|J'_1|,\ldots,|J'_m|)$$

Uniform Cardinal Random Probing Composability

- Best of both world : Tight security from General RPC and complexity from Cardinal RPC.
- One Condition :

$$\varepsilon^{g}_{(J_1,\ldots,J_m)}(I_1,\ldots,I_\ell) = \varepsilon^{g}_{(J'_1,\ldots,J'_m)}(I'_1,\ldots,I'_\ell)$$

when

$$(|I_1|,\ldots,|I_\ell|) = (|I'_1|,\ldots,|I'_\ell|)$$
$$(|J_1|,\ldots,|J_m|) = (|J'_1|,\ldots,|J'_m|)$$

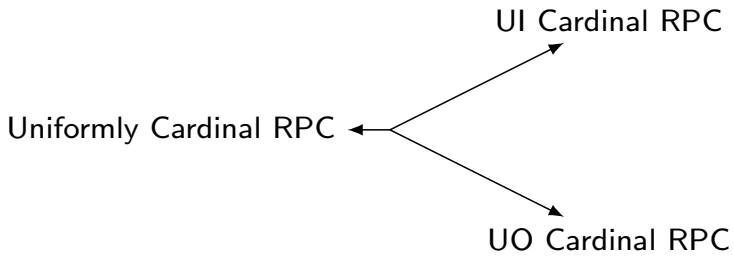
I \ J	$\emptyset$	$\{y_1\}$	$\{y_2\}$	$\{y_3\}$	$\{y_1,y_2\}$	$\{y_1,y_3\}$	$\{y_2,y_3\}$	$\{y_1,y_2,y_3\}$
$\emptyset$								
$\{x_1\}$								
$\{x_2\}$								
$\{x_3\}$								
$\{x_1,x_2\}$								
$\{x_1,x_3\}$								
$\{x_2,x_3\}$								
$\{x_1,x_2,x_3\}$								

I \ J	0	1	2	3
0				
1				
2				
3				

Our contribution

- Revisit the notion of uniformly cardinal RPC.
- Method to transform any gadget satisfying cardinal/general RPC into a uniformly cardinal RPC.
- Design a non-linear multiplication gadget and prove its (uniformly) cardinal RPC security.
- Design a compiler, apply it to the AES circuit, and compare it to state-of-the-art methods in terms of the achievable security-performance trade-off RPC security

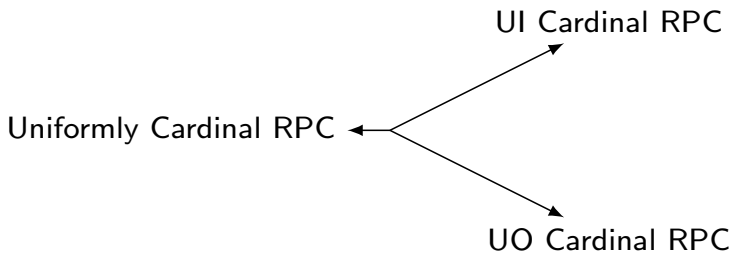
UI/UO cardinal RPC



- UI cardinal RPC, uniformity on the input shares **only**.

$$\varepsilon^g_{(J_1,\ldots,J_m)}(I_1,\ldots,I_\ell) = \varepsilon^g_{(J_1,\ldots,J_m)}(I'_1,\ldots,I'_\ell)$$

UI/UO cardinal RPC



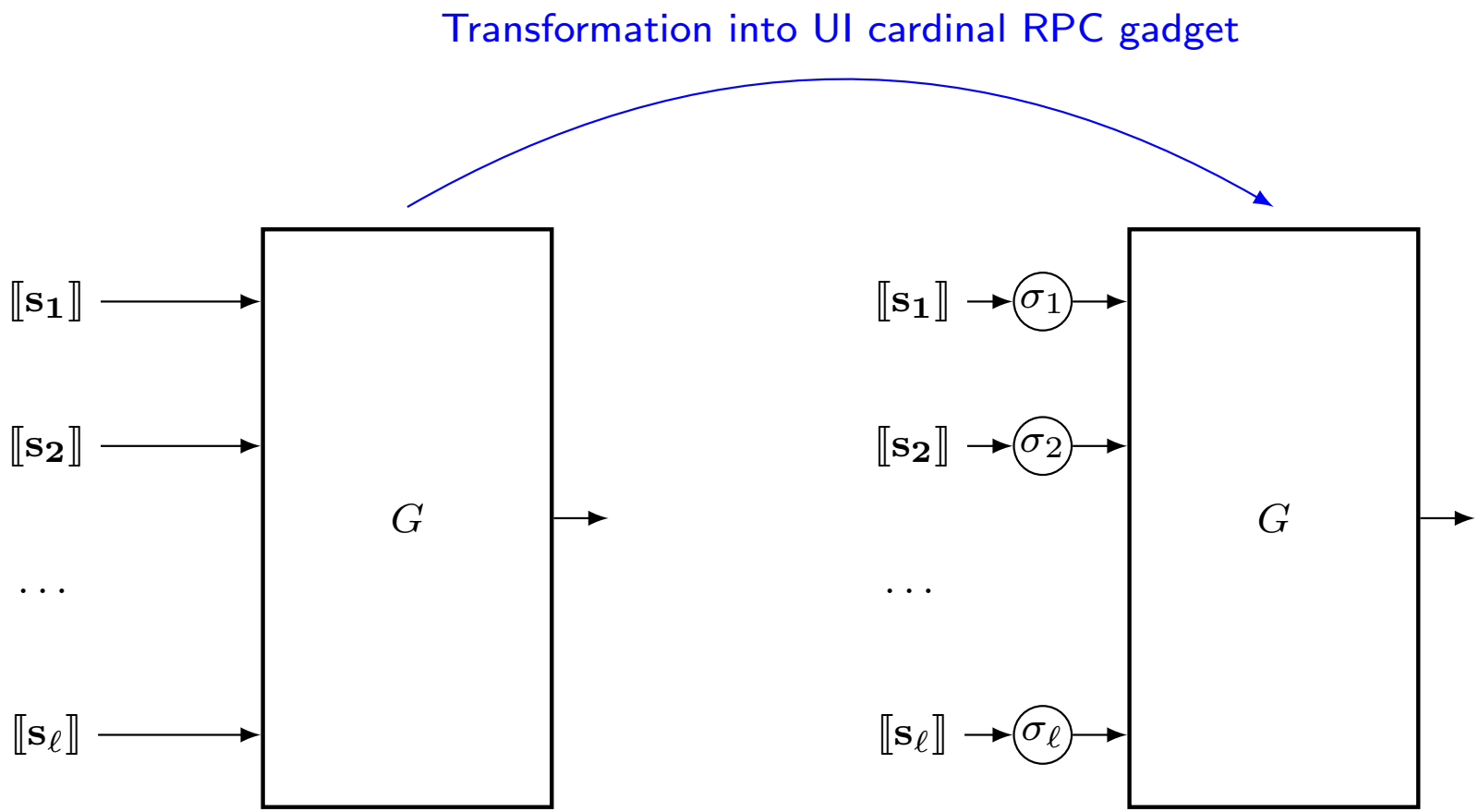
- UI cardinal RPC, uniformity on the input shares **only**.

$$\varepsilon^{g}_{(J_1,\ldots,J_m)}(I_1,\ldots,I_\ell) = \varepsilon^{g}_{(J_1,\ldots,J_m)}(I'_1,\ldots,I'_\ell)$$

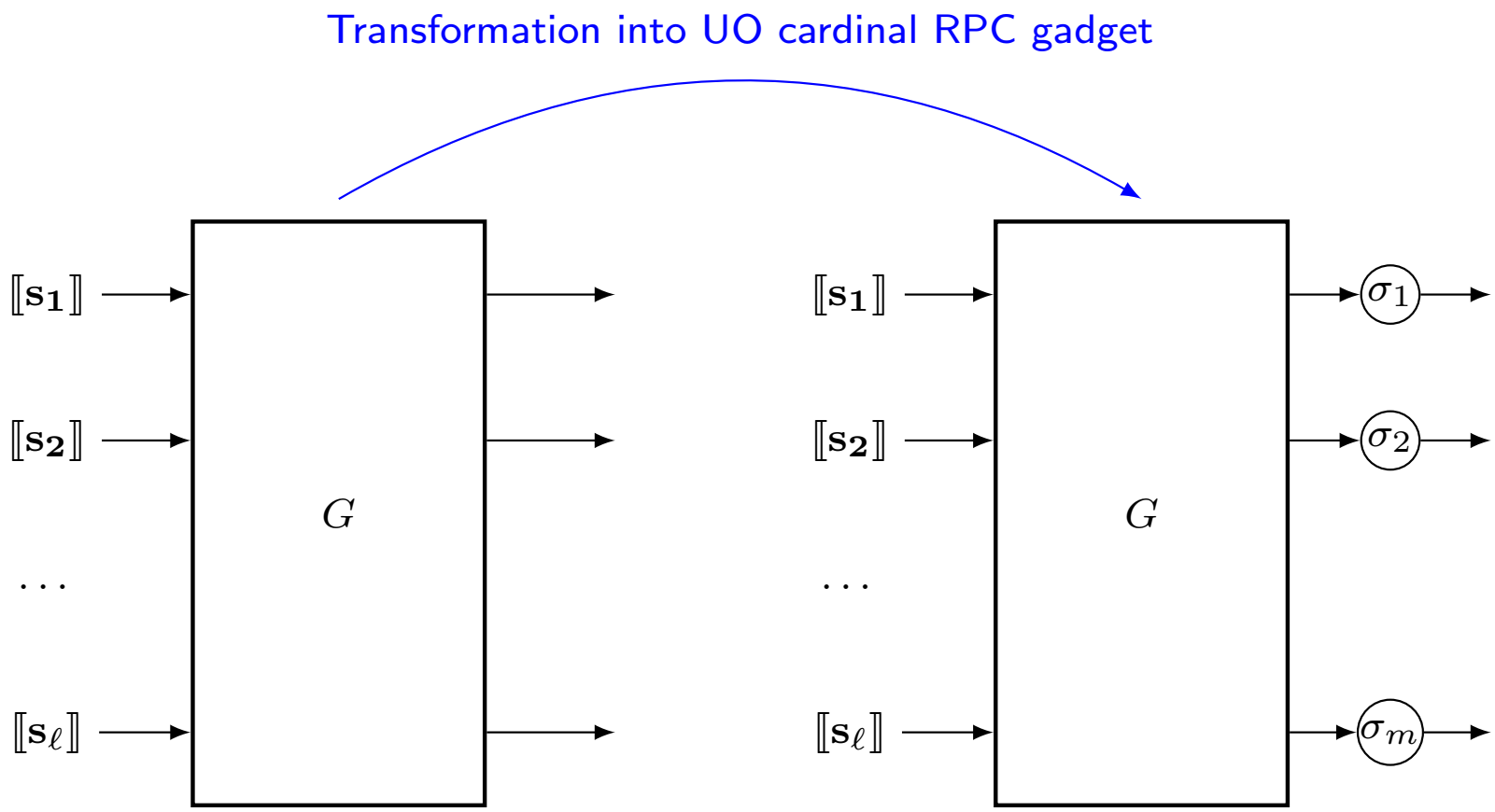
- UO cardinal RPC, uniformity on the output shares **only**.

$$\varepsilon^{g}_{(J_1,\ldots,J_m)}(I_1,\ldots,I_\ell) = \varepsilon^{g}_{(J'_1,\ldots,J'_m)}(I_1,\ldots,I_\ell)$$

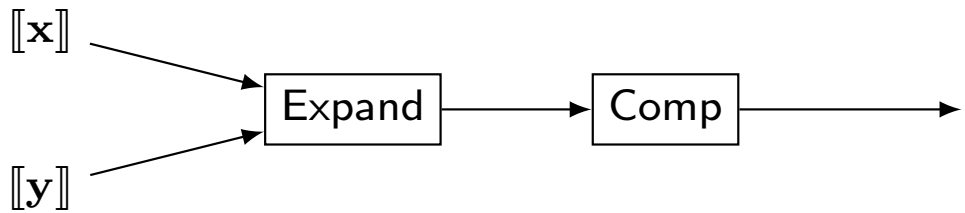
Put a gadget UI cardinal RPC



Put a gadget UO cardinal RPC

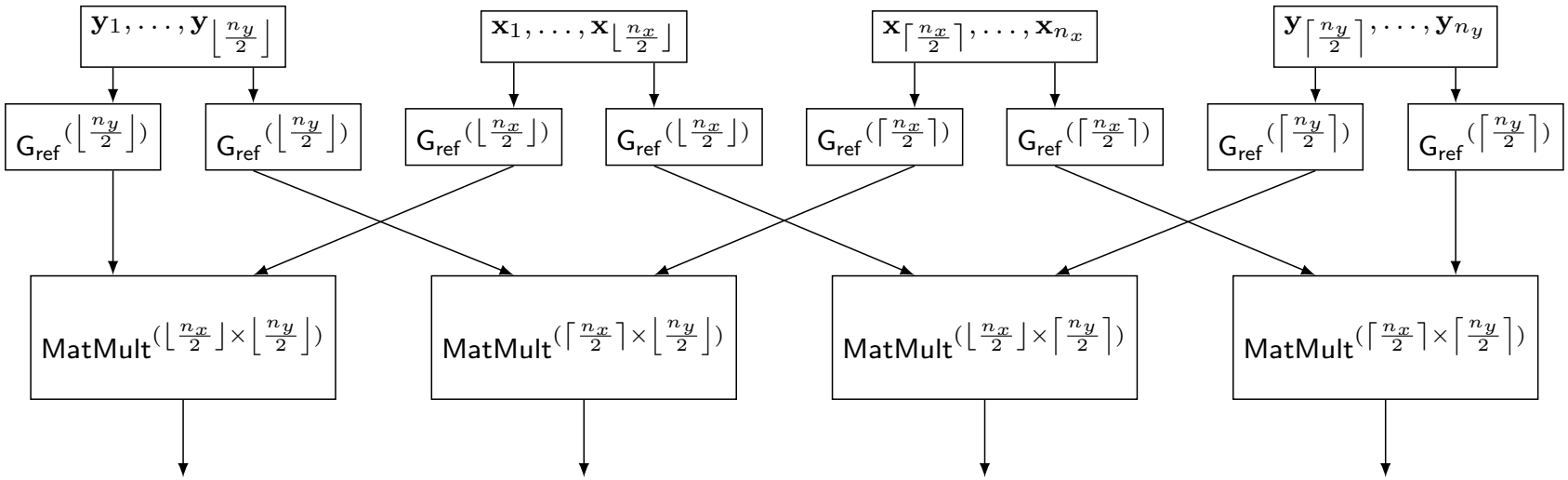


Multiplication gadget

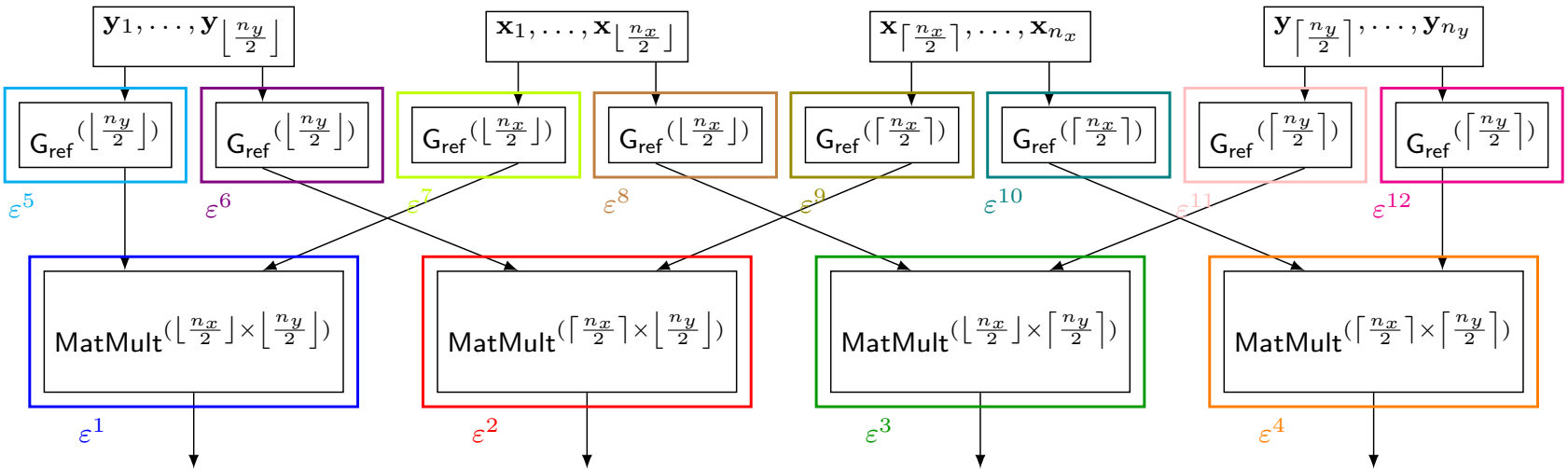


- Expand : From two n-sharing, compute the cross product $x_i \cdot y_j$ to obtain a n^2 sharing.
- Comp : Transform a n^2-sharing into a n-sharing.

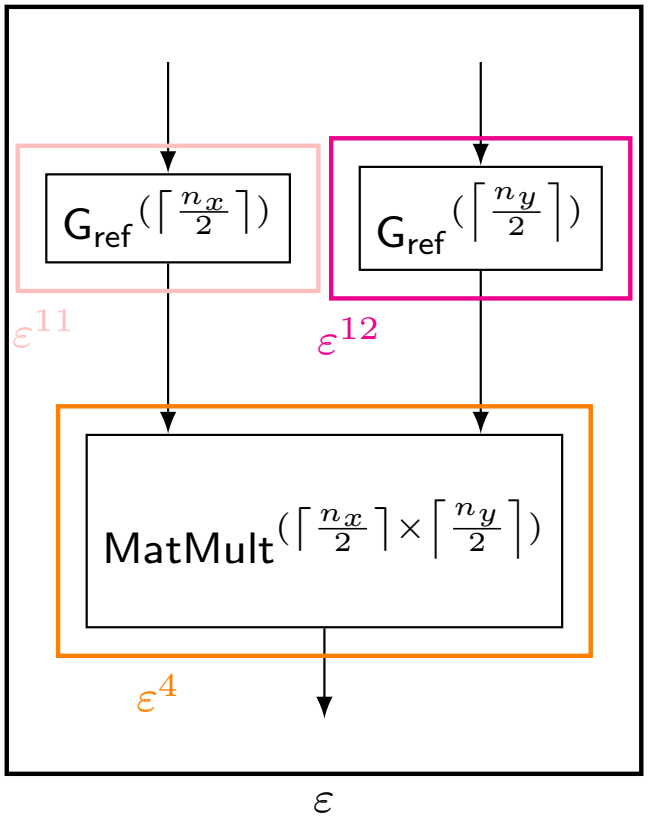
Expand : MatMultSym



Expand : MatMultSym

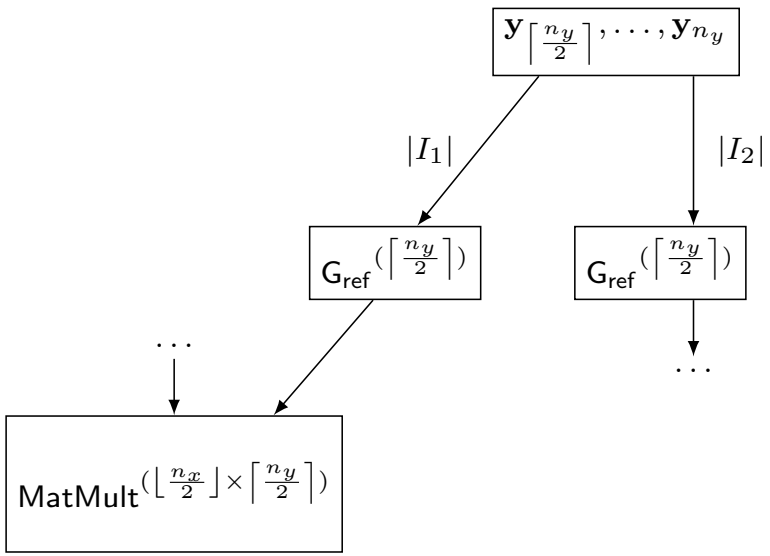


Expand : MatMultSym- Branch



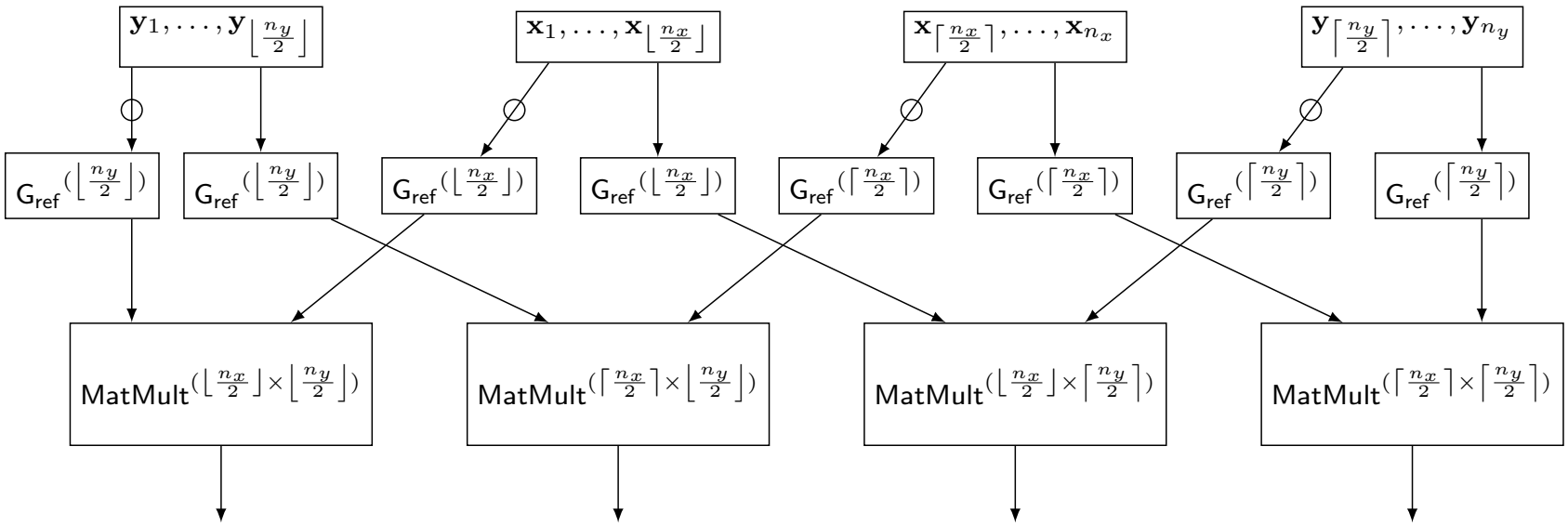
$$\varepsilon_j(i_1, i_2) = \sum_{\ell_1=0}^{n} \sum_{\ell_2=0}^{n} {\varepsilon^{11}}_{\ell_1}(i_1) \cdot {\varepsilon^{12}}_{\ell_2}(i_2) \cdot {\varepsilon^4}_j(\ell_1, \ell_2)$$

Expand : MatMultSym- Drawbacks

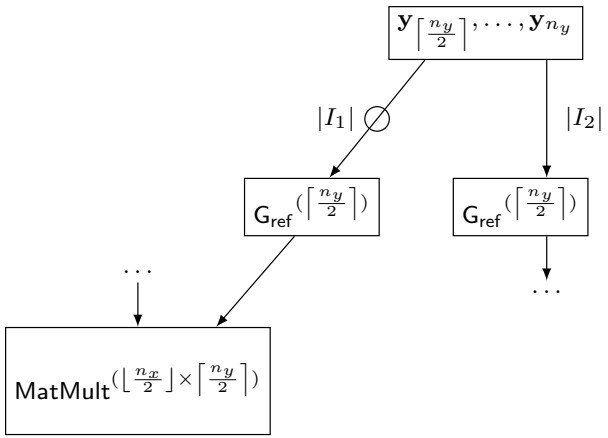


- $|I_1|$ required shares for the simulation of the first branch.
- $|I_2|$ required shares for the simulation of the second branch.
- **No informations** on $|I_1 \cup I_2|$.
- Must consider the worst case: $|I_1 \cup I_2| = \min(n, |I_1| + |I_2|)$.
- **Looser Bounds**

Expand : UnifMatMult



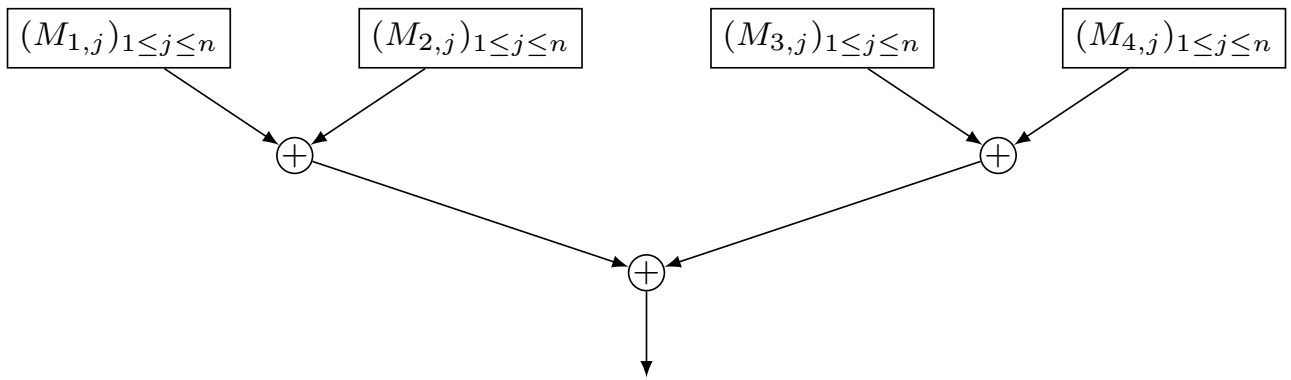
Expand : UnifMatMult



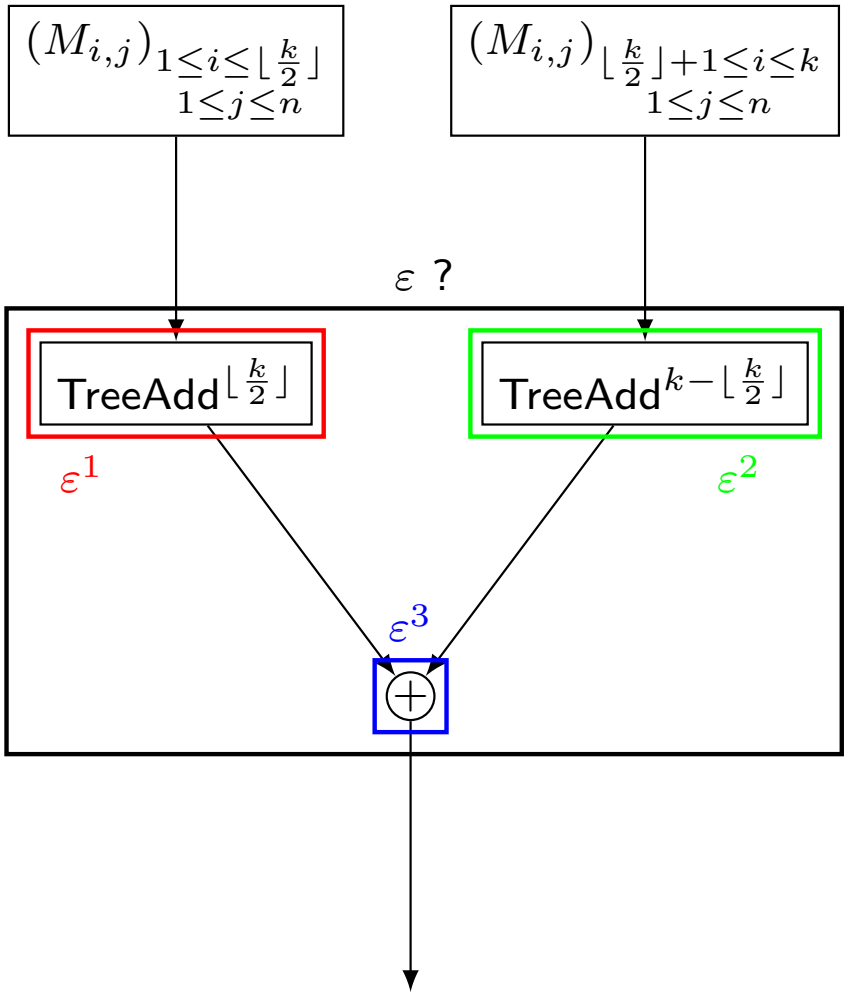
- **Informations** on $|I_1 \cup I_2|$.
- $\mathbb{P}(|I_1 \cup I_2| = \ell, |I_1| = \ell_1, |I_2| = \ell_2) = \frac{\binom{\ell_1}{\ell_1+\ell_2-\ell} \cdot \binom{n-\ell_1}{\ell-\ell_1}}{\binom{n}{\ell_2}}$
- **Tighter Bounds**

Comp : TreeAdd

- Transform a n^2 sharing into a n-sharing
- Tree of n-share gadget addition.
- Example with $n = 4$, and a $(M_{i,j})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}}$ the n^2-sharing :

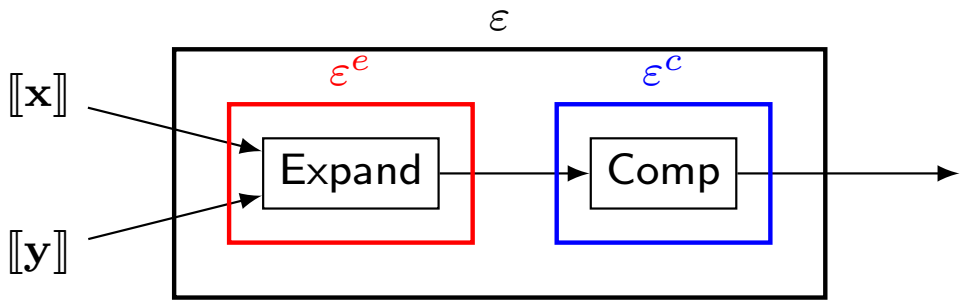


Comp : TreeAdd



$$\varepsilon_j(i) = \sum_{i_1+i_2=i} \sum_{\ell_1=0}^{n} \sum_{\ell_2=0}^{n} {\varepsilon^1}_{\ell_1}(i_1) \cdot {\varepsilon^2}_{\ell_2}(i_2) \cdot {\varepsilon^3}_{j}(\ell_1, \ell_2)$$

Multiplication gadget

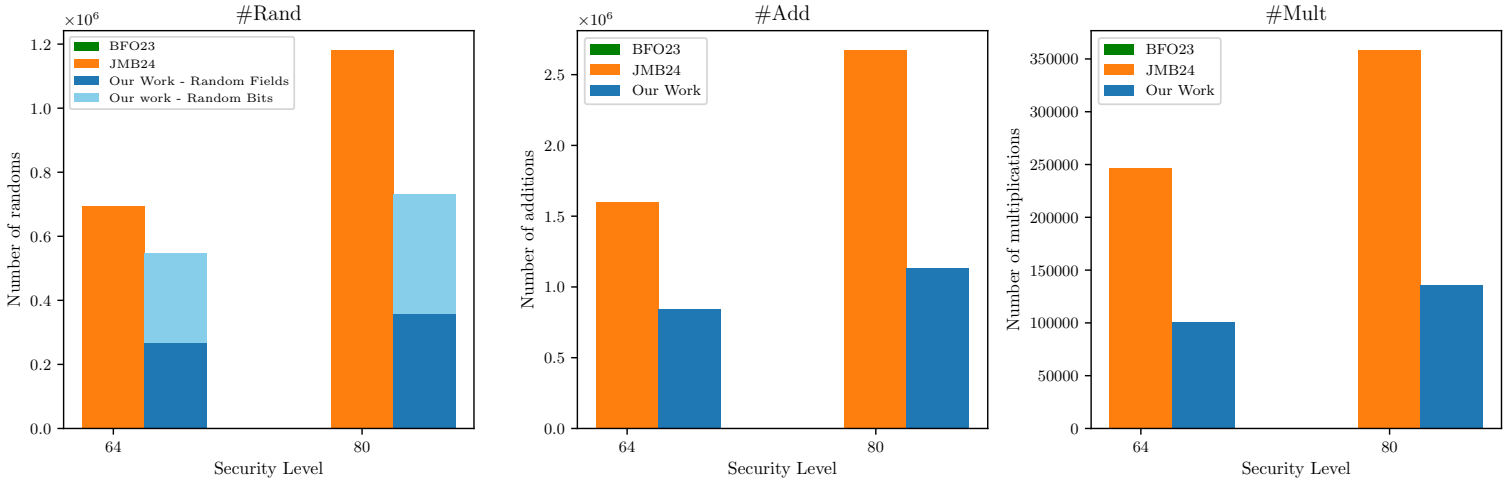


$$\varepsilon_j(i_{\mathbf{x}}, i_{\mathbf{y}}) = \sum_{\ell=0}^{n^2} {\varepsilon^e}_\ell(i_{\mathbf{x}}, i_{\mathbf{y}}) \cdot {\varepsilon^c}_j(\ell)$$

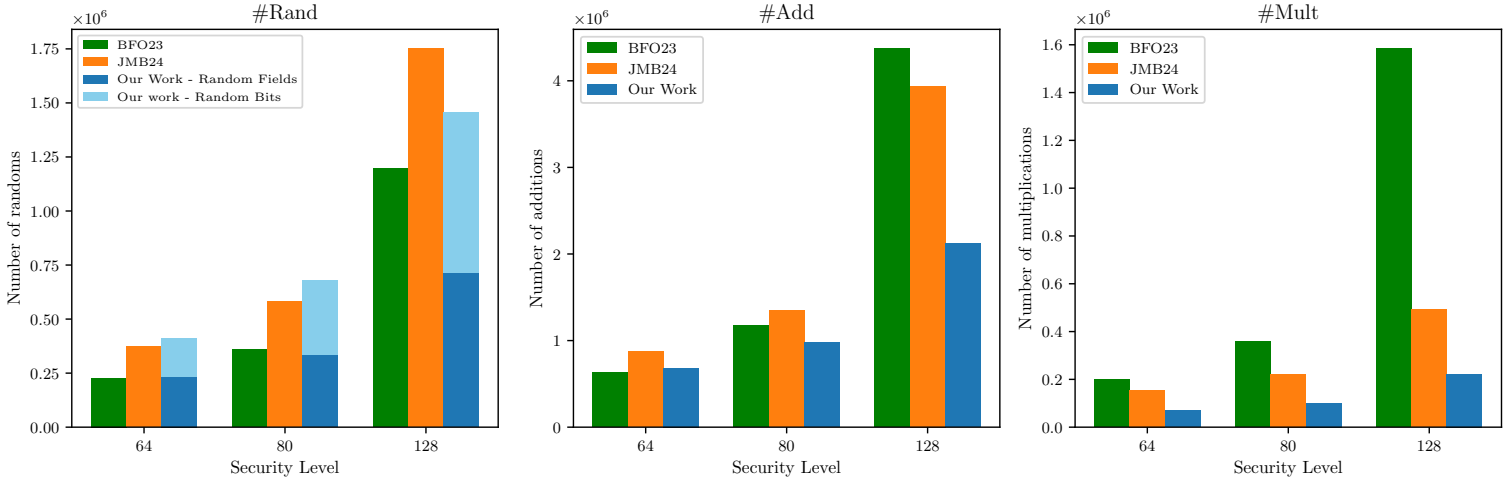
Compiler

- Linear Gadgets : $\mathsf{Add}^\gamma, \mathsf{cMult}^\gamma, \mathsf{cAdd}^\gamma \rightarrow$ Known cardinal RPC enveloppes.
- Squaring Gadget : In case $\mathbb{K} = \mathbb{F}_{2^k}$, $\mathsf{Sq}^\gamma \rightarrow$ Known cardinal RPC security enveloppes.
- Non Linear Gadget : Multiplication gadget used with UnifMatMult $\rightarrow$ Now known cardinal RPC security enveloppes.

AES Results



(a) $p = 2^{-16}$



(b) $p = 2^{-20}$

Thank You for your attention !