Probing Security through Input-Output Separation and Revisited Quasilinear Masking

Dahmun Goudarzi, Thomas Prest,
Matthieu Rivain and Damien Vergnaud

— CHES 2021 —

Introduction

- What is this about ?
  - Security against side-channel attacks
  - Masking schemes
  - Formal proofs through probing security
- Our contributions
  - New masking composition approach:

    IOS refresh gadget + probing-secure gadgets
    $\Rightarrow$ region probing security of the composition

  - Quasilinear IOS refresh gadget (variant of [BPCZ, CHES'16])
  - Quasilinear masking scheme (improved version of [GJR, AC'18])

Introduction

- What is this about ?
  - Security against side-channel attacks
  - Masking schemes
  - Formal proofs through probing security
- Our contributions
  - New masking composition approach:

    IOS refresh gadget + probing-secure gadgets
    ⇒ region probing security of the composition

    new simple property
  - Quasilinear IOS refresh gadget (variant of [BPCZ, CHES'16])
  - Quasilinear masking scheme (improved version of [GJR, AC'18])

Introduction

- What is this about ?
  - Security against side-channel attacks
  - Masking schemes
  - Formal proofs through probing security
- Our contributions
  - New masking composition approach:

    IOS refresh gadget + probing-secure gadgets
    ⇒ region probing security of the composition

    new simple property (IOS)

    weaker than composition properties (probing-secure)

  - Quasilinear IOS refresh gadget (variant of [BPCZ, CHES'16])
  - Quasilinear masking scheme (improved version of [GJR, AC'18])

Introduction

- What is this about ?
  - Security against side-channel attacks
  - Masking schemes
  - Formal proofs through probing security
- Our contributions
  - New masking composition approach:

    IOS refresh gadget + probing-secure gadgets
    ⇒ region probing security of the composition

    new simple property

    weaker than composition properties

    stronger than probing security
  - Quasilinear IOS refresh gadget (variant of [BPCZ, CHES'16])
  - Quasilinear masking scheme (improved version of [GJR, AC'18])

Masking

$$x \xrightarrow{\text{Encode}} \overrightarrow{x} = (x_1, \ldots, x_n)$$

$$x = x_1 + \cdots + x_n$$ (on a field $\mathbb{K}$)

Masking

sharing

the shares

$$x \xrightarrow{\text{Encode}} \vec{x} = (x_1, \ldots, x_n)$$

$$x = x_1 + \cdots + x_n \quad \text{(on a field } \mathbb{K}\text{)}$$

n : number of shares

Masking

sharing — *the shares*

$$x \xrightarrow{\text{Encode}} \vec{x} = (x_1, \ldots, x_n)$$

In this work:

$$x = v_1 \cdot x_1 + \cdots + v_n \cdot x_n$$

$$= \langle \vec{v}, \vec{x} \rangle \qquad \text{(on a field } \mathbb{K}\text{)}$$

Masking

sharing

the shares

$$x \xrightarrow{\text{Encode}} \vec{x} = (x_1, \ldots, x_n)$$

In this work:

$$x = v_1 \cdot x_1 + \cdots + v_n \cdot x_n$$

$$= \langle \vec{v}, \vec{x} \rangle \qquad \text{(on a field } \mathbb{K}\text{)}$$

constant coefficients

sharing

Masking

sharing

the shares

$$x \xrightarrow{\text{Encode}} \vec{x} = (x_1, \ldots, x_n)$$

In this work:

$$x = v_1 \cdot x_1 + \cdots + v_n \cdot x_n$$

$$= \langle \vec{v}, \vec{x} \rangle \qquad \text{(on a field } \mathbb{K}\text{)}$$

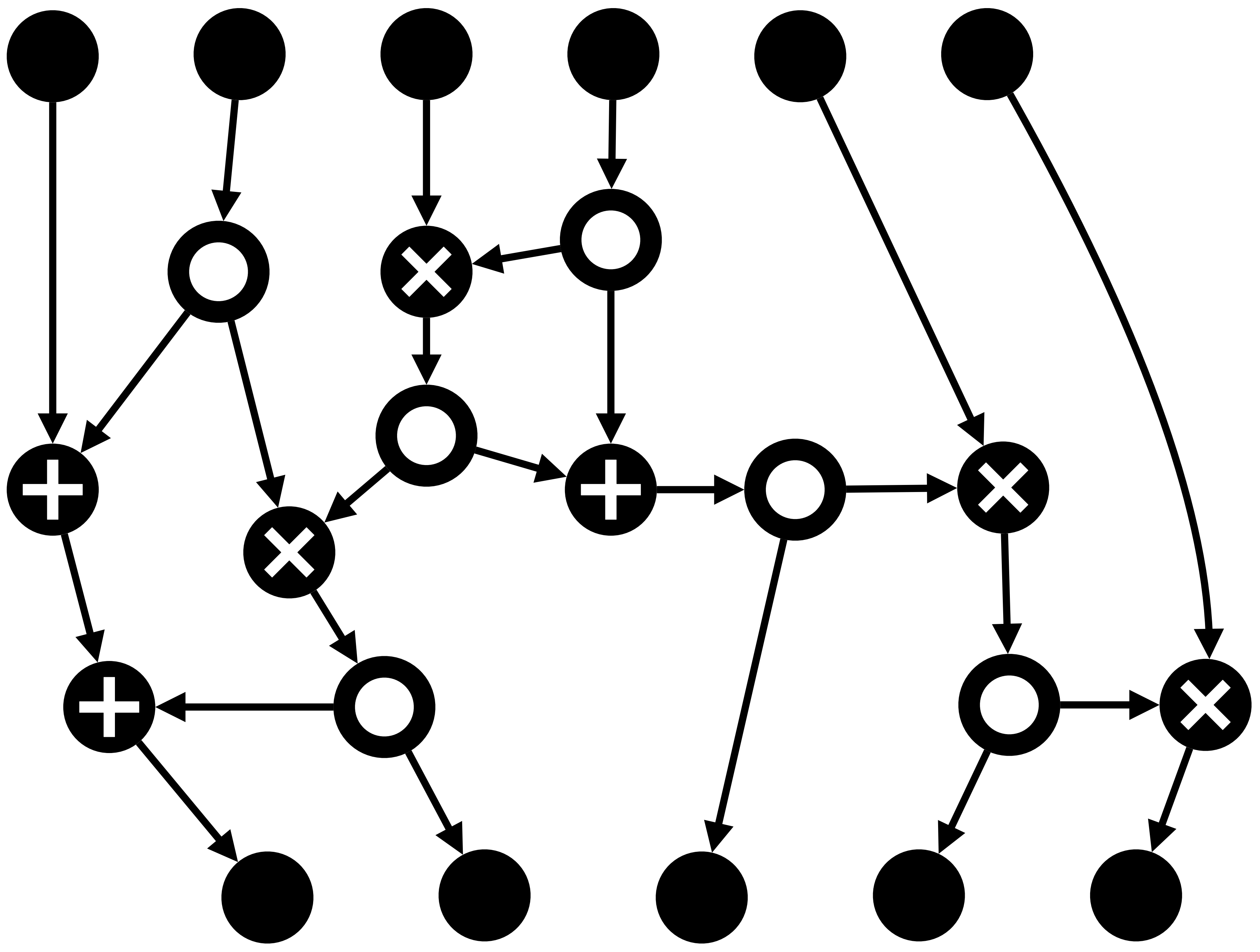
constant coefficients

sharing

$\vec{v}$-sharing of x

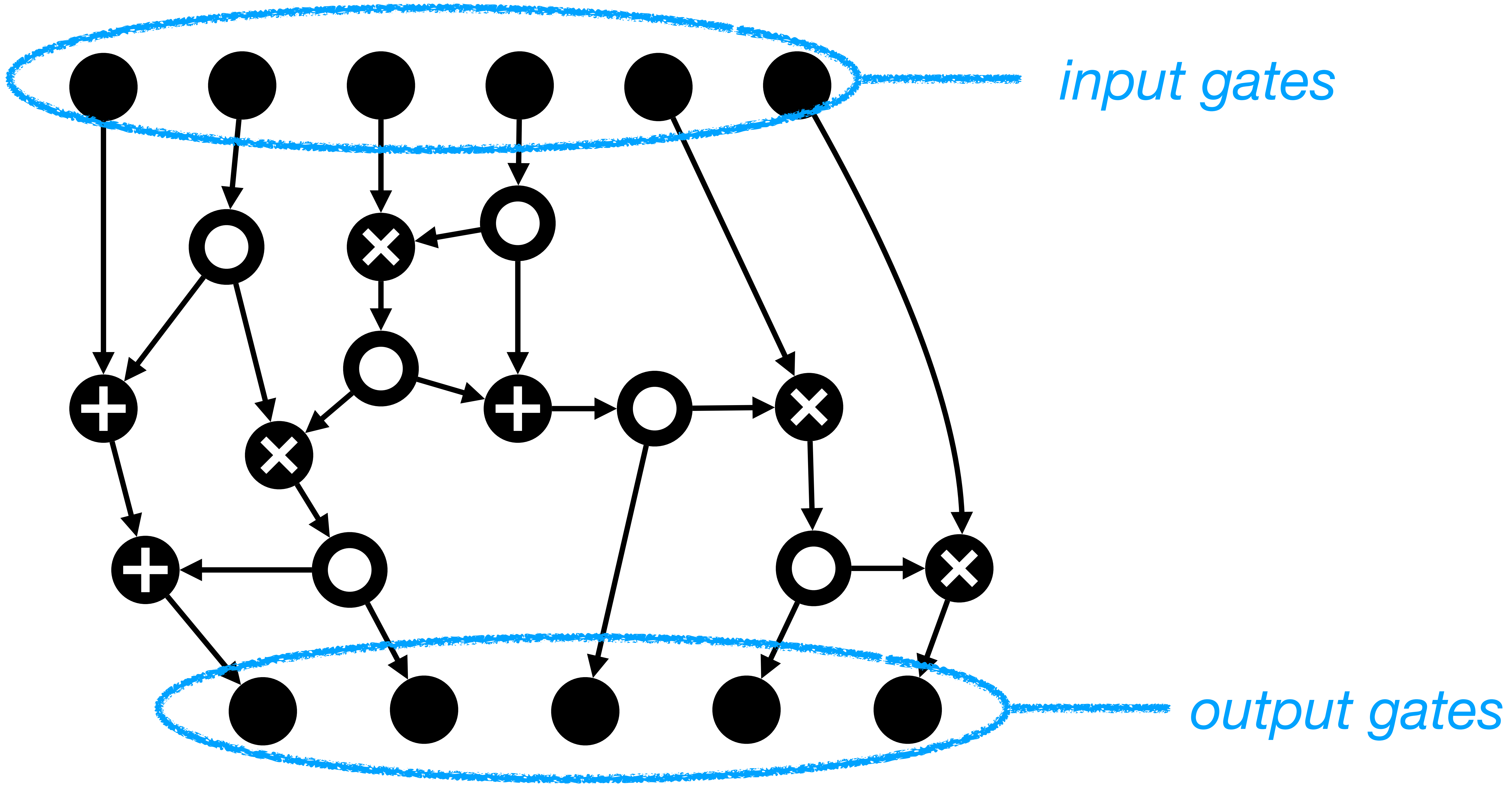
Circuit model

Crypto computation modelled as an arithmetic circuit on $\mathbb{K}$



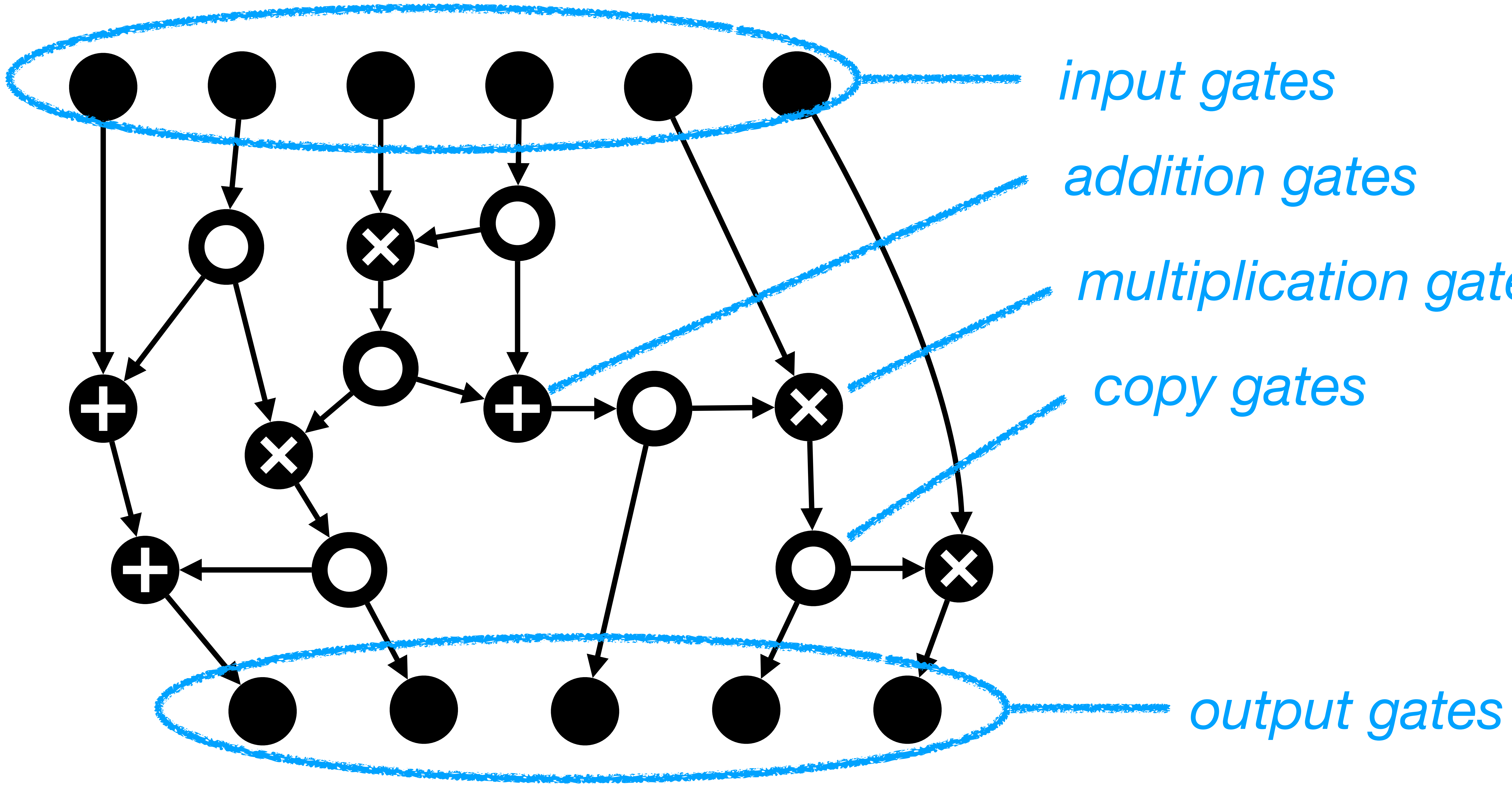
Circuit model

Crypto computation modelled as an arithmetic circuit on $\mathbb{K}$



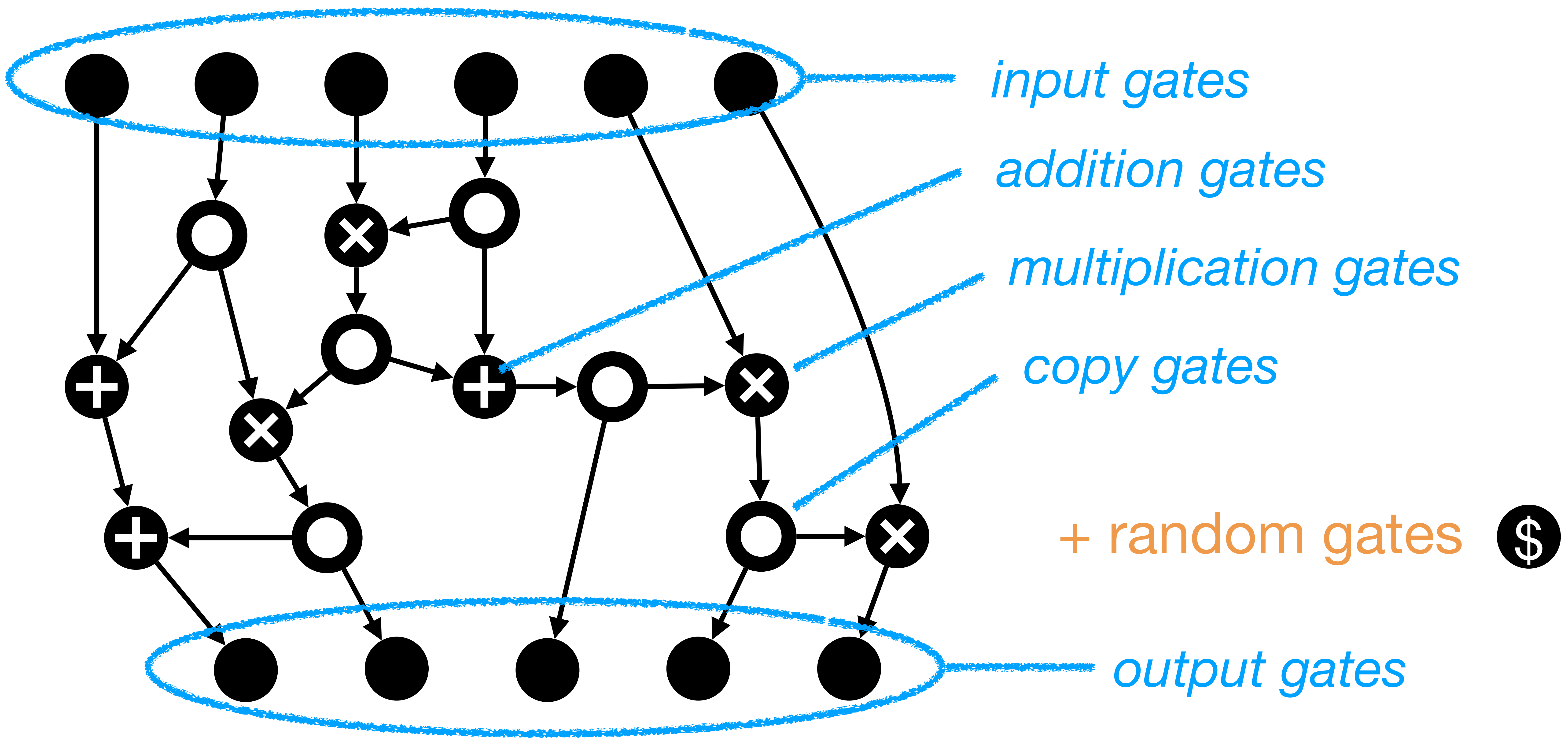
Circuit model

Crypto computation modelled as an arithmetic circuit on $\mathbb{K}$

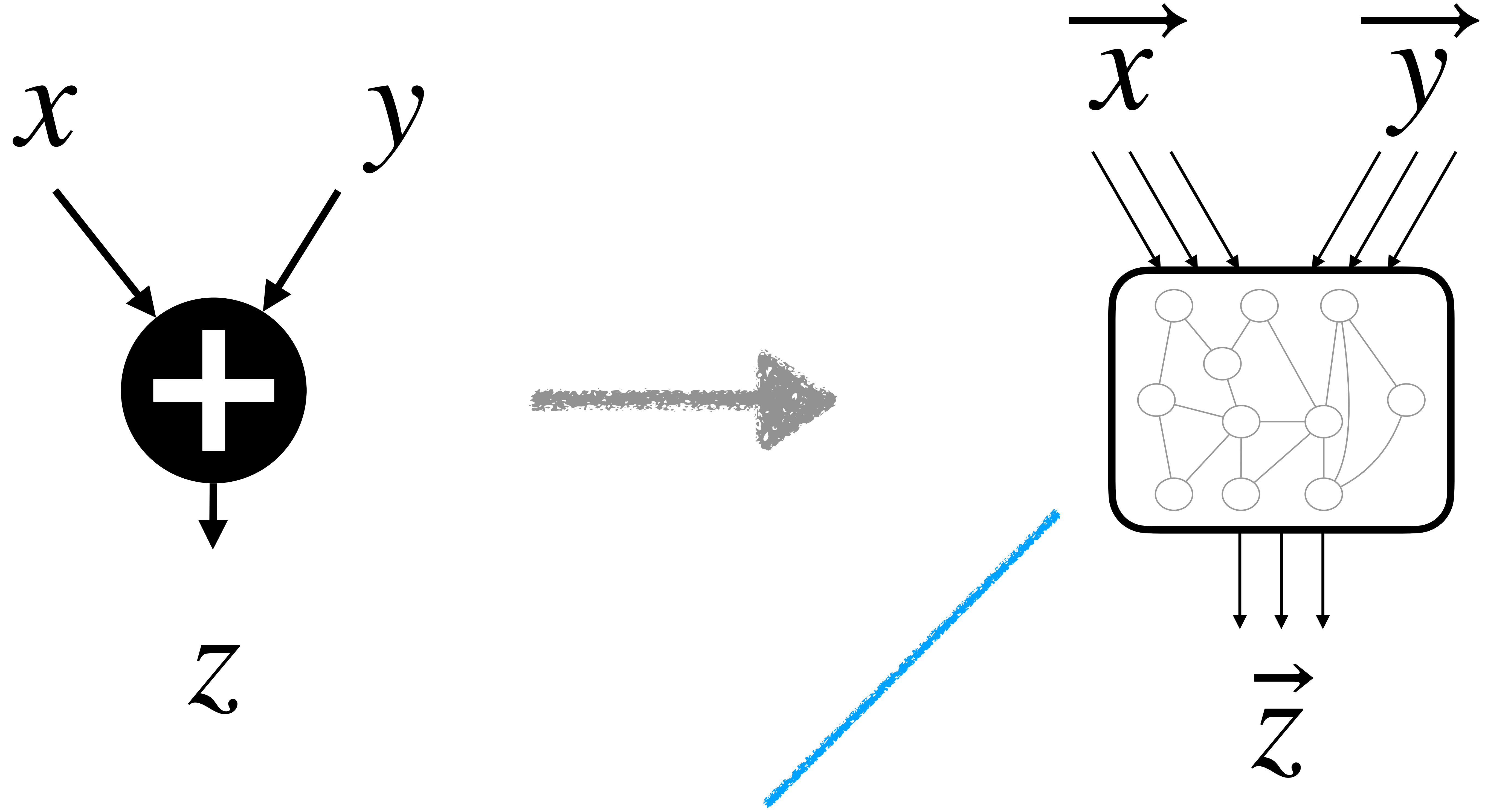


Circuit model

Crypto computation modelled as an arithmetic circuit on $\mathbb{K}$

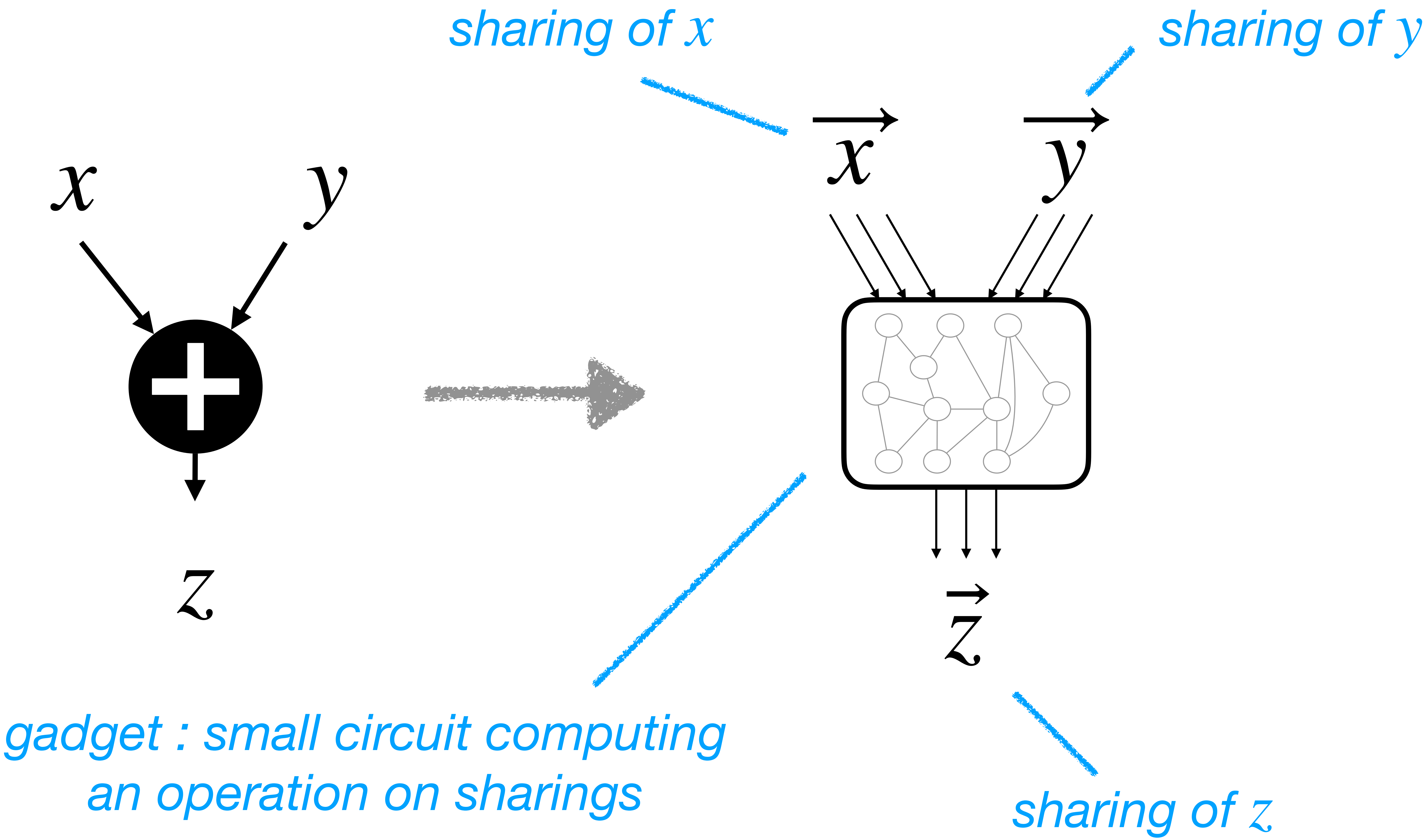


Gadgets



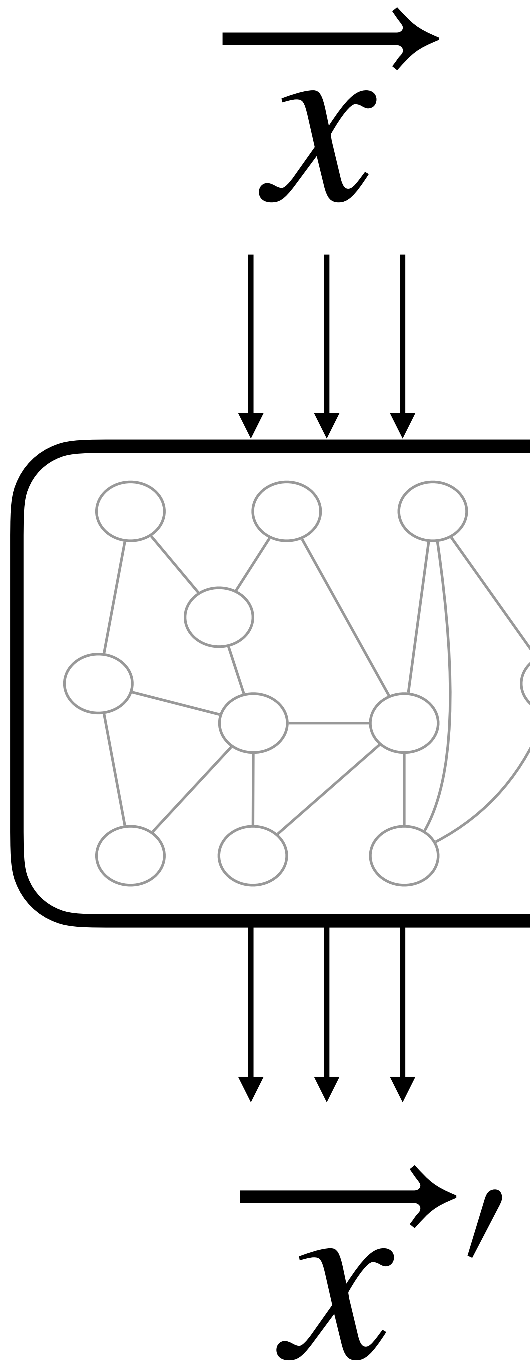
gadget : small circuit computing
an operation on sharings

Gadgets

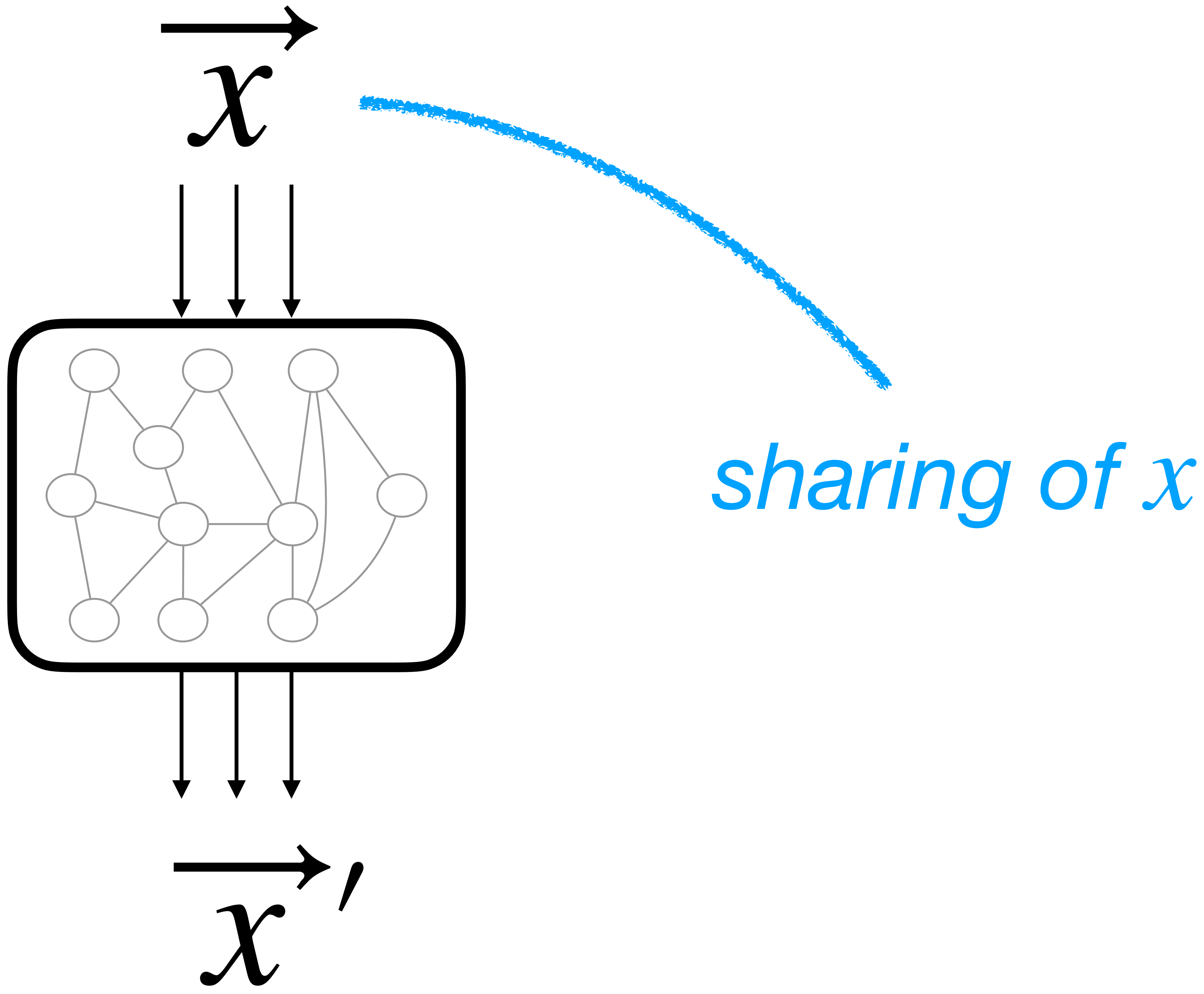


sharing of x
sharing of y
x
y
z
$\vec{x}$
$\vec{y}$
$\vec{z}$
gadget : small circuit computing
an operation on sharings
sharing of z

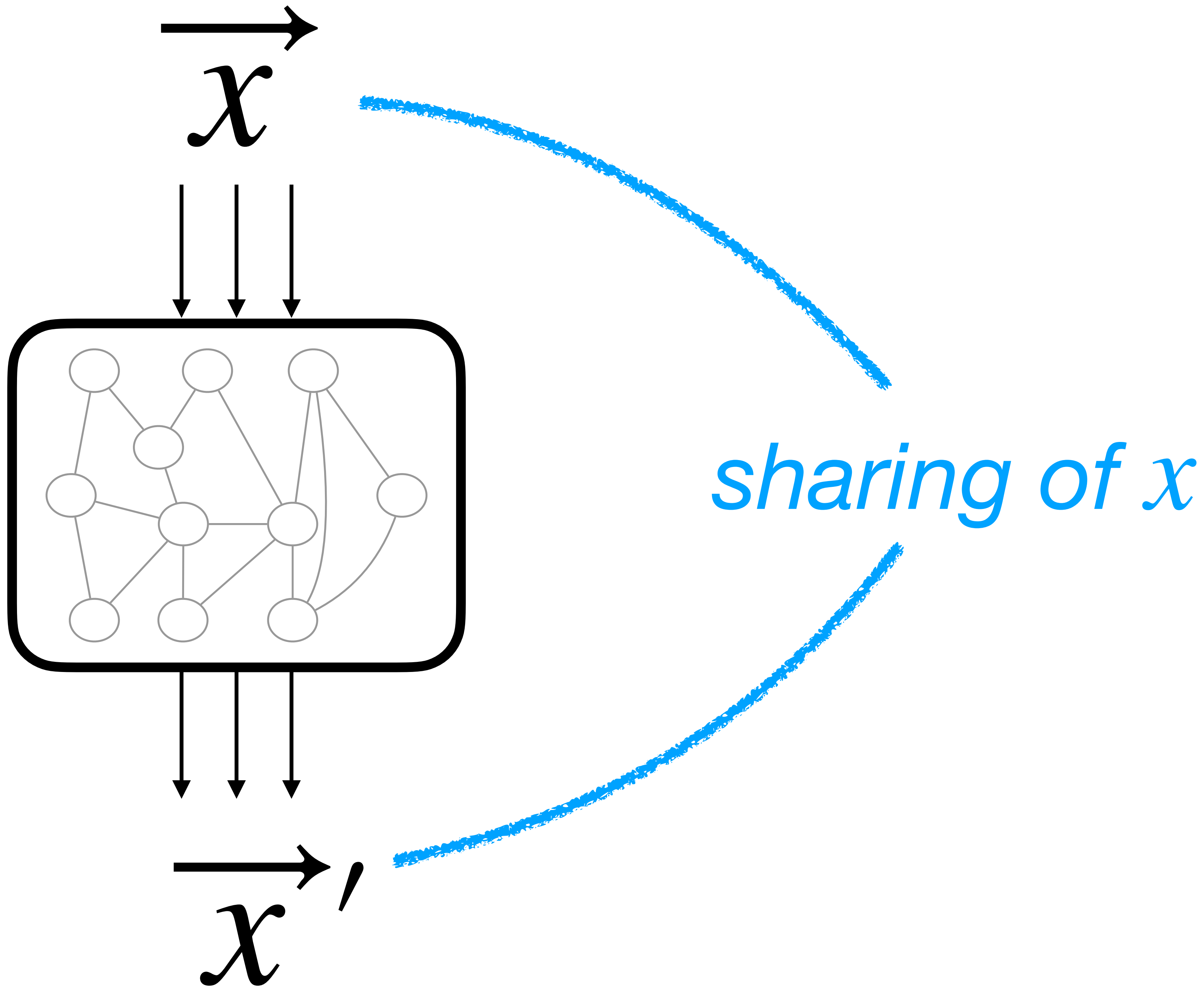
Refresh gadgets



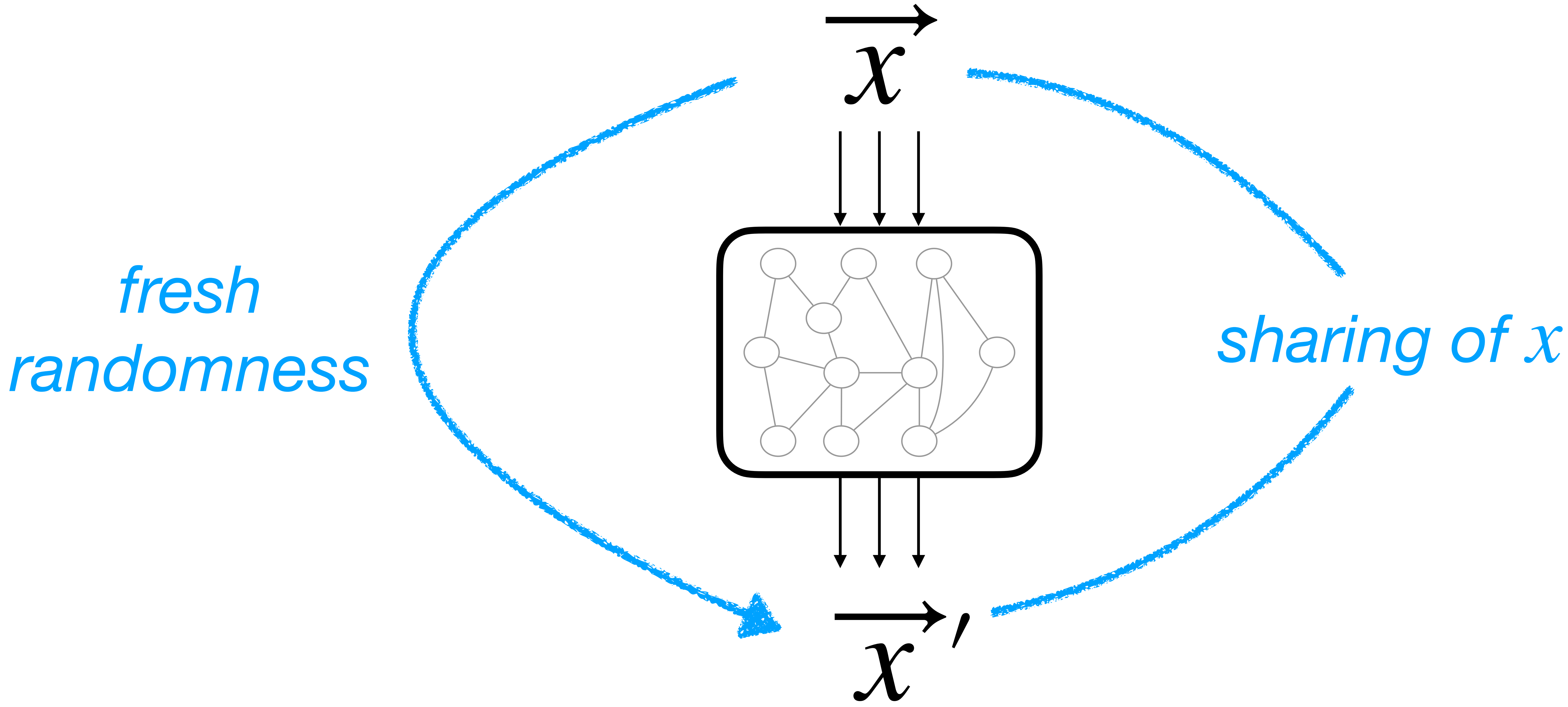
Refresh gadgets



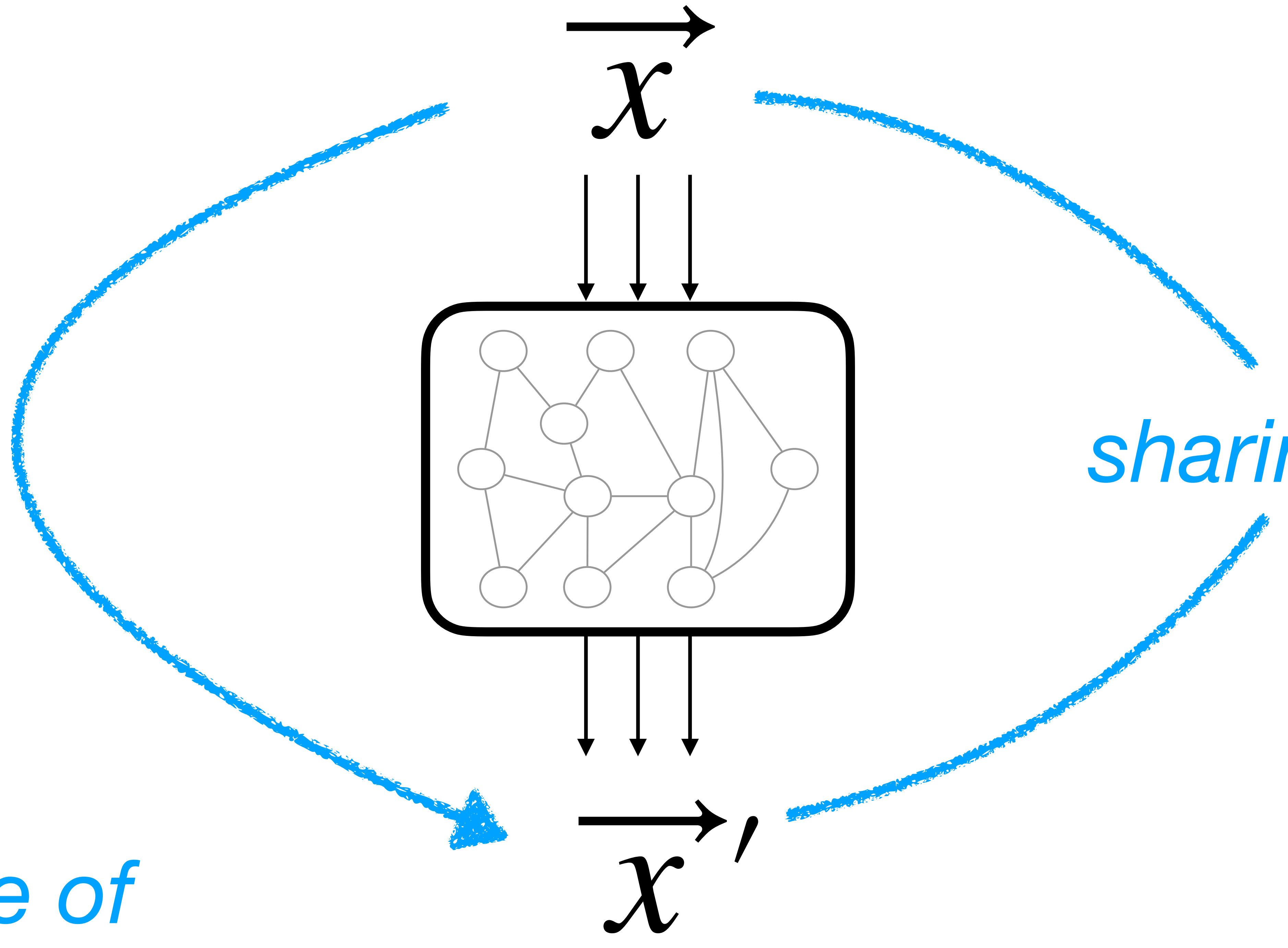
Refresh gadgets



Refresh gadgets



Refresh gadgets



fresh randomness

$\Rightarrow$

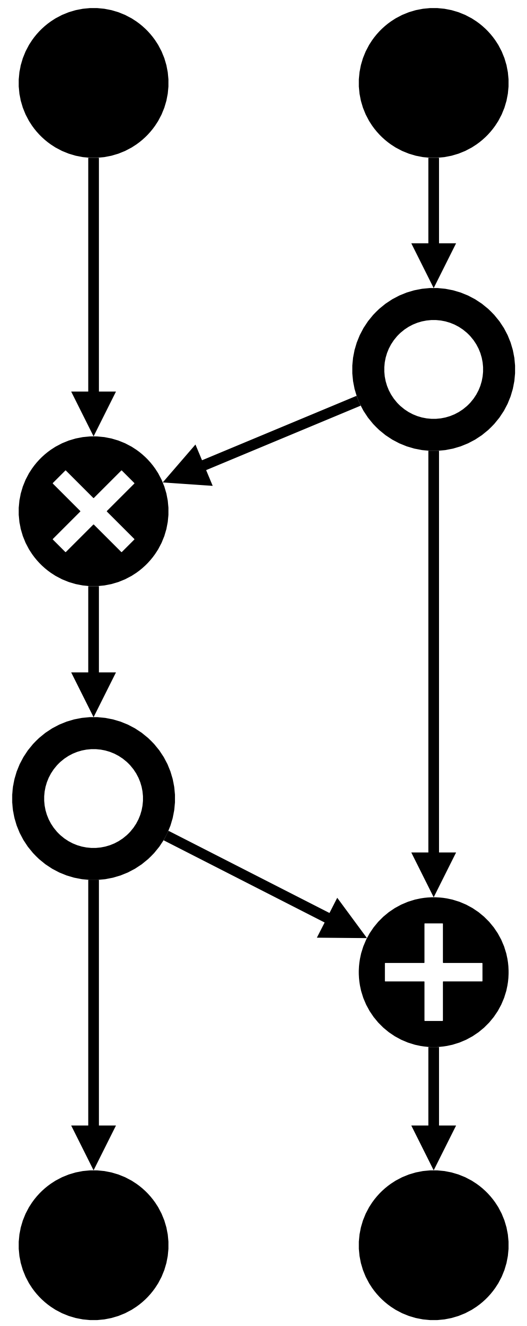
statistical independence of $(\overrightarrow{x} \,|\, x)$ *and* $(\overrightarrow{x}' \,|\, x)$

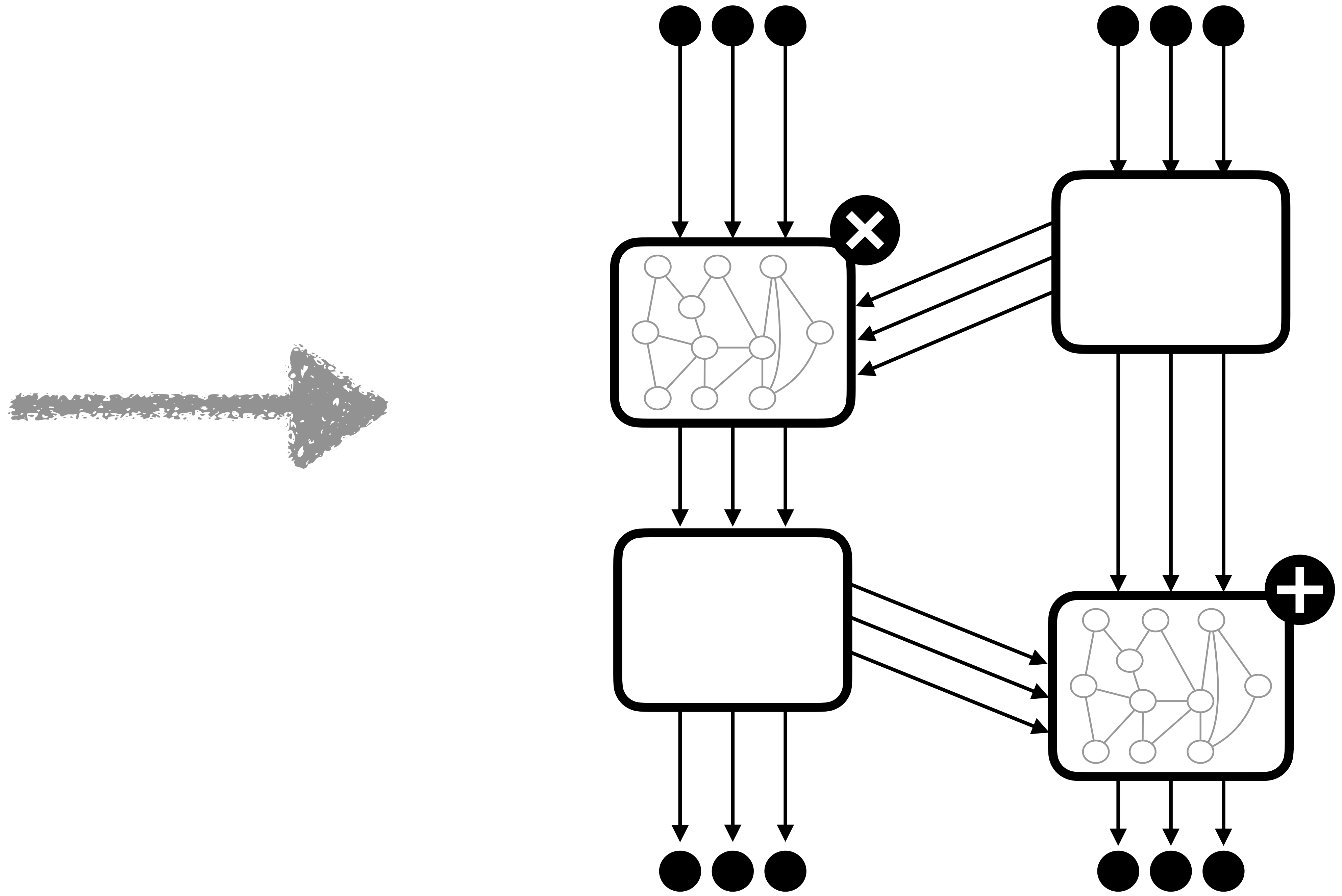
sharing of x

Standard circuit compiler

wire → n wires (sharing)

gate → gadget

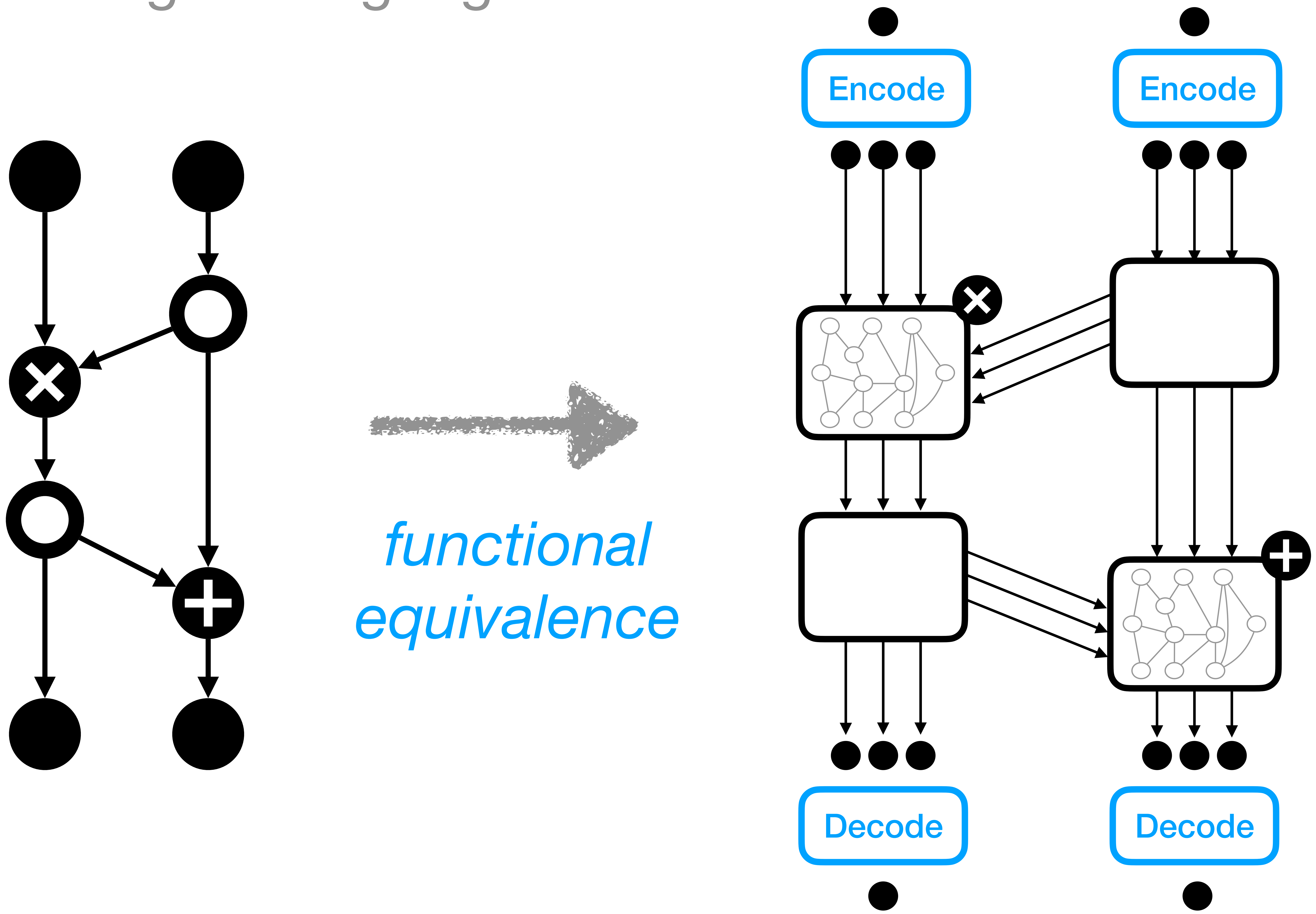




Standard circuit compiler

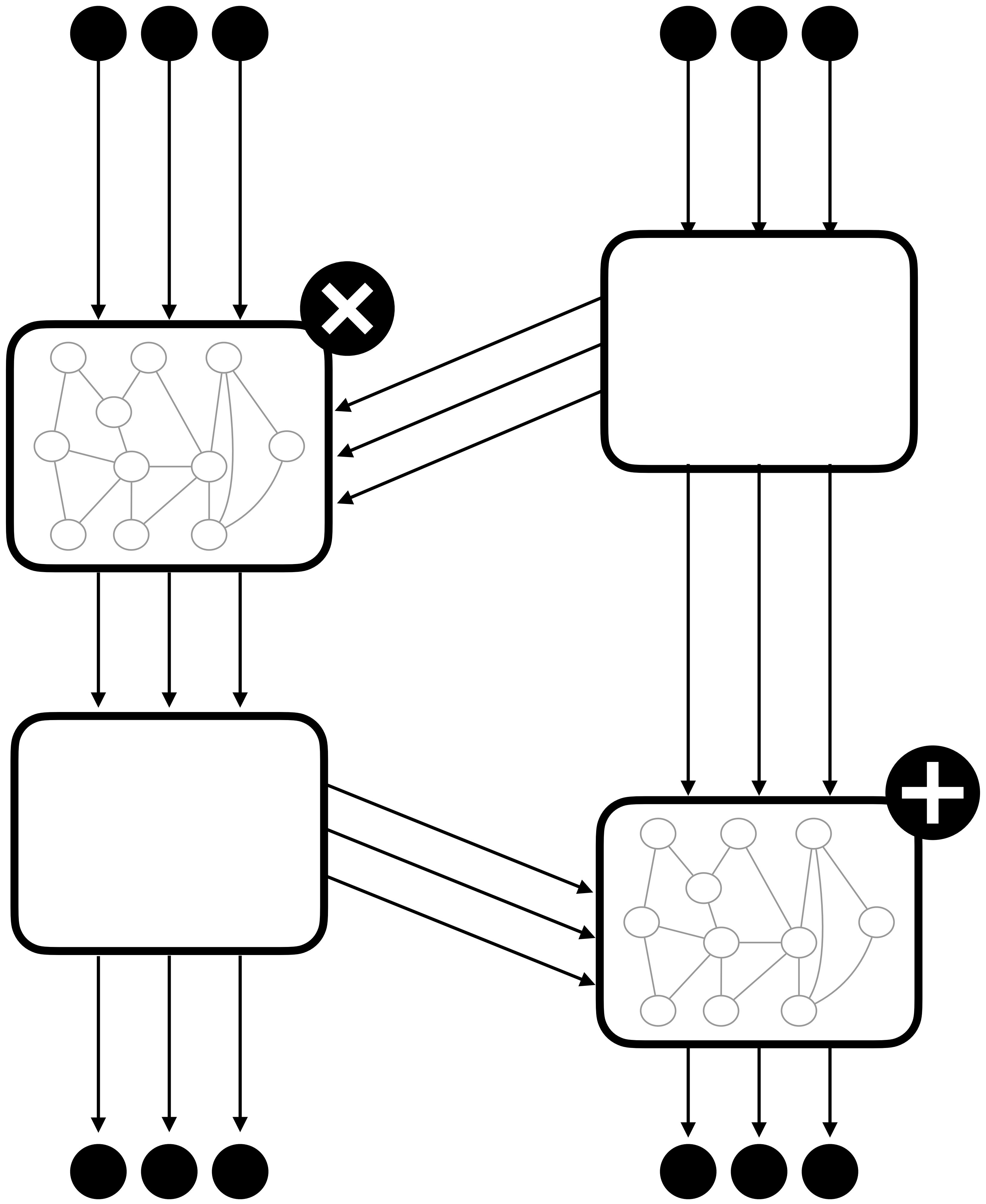
wire → n wires (sharing)

gate → gadget



Standard circuit compiler …

… with full refreshing

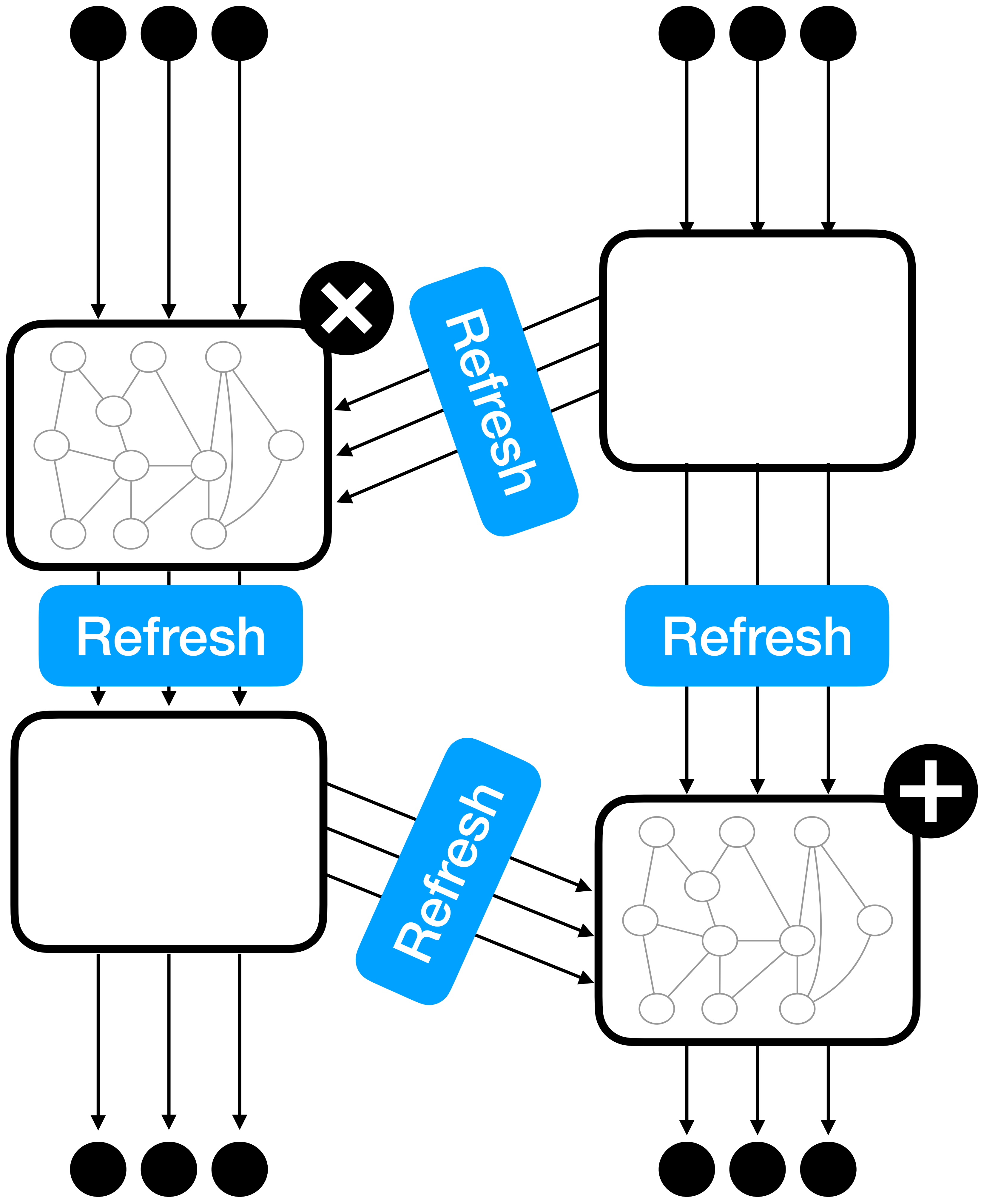


Standard circuit compiler …

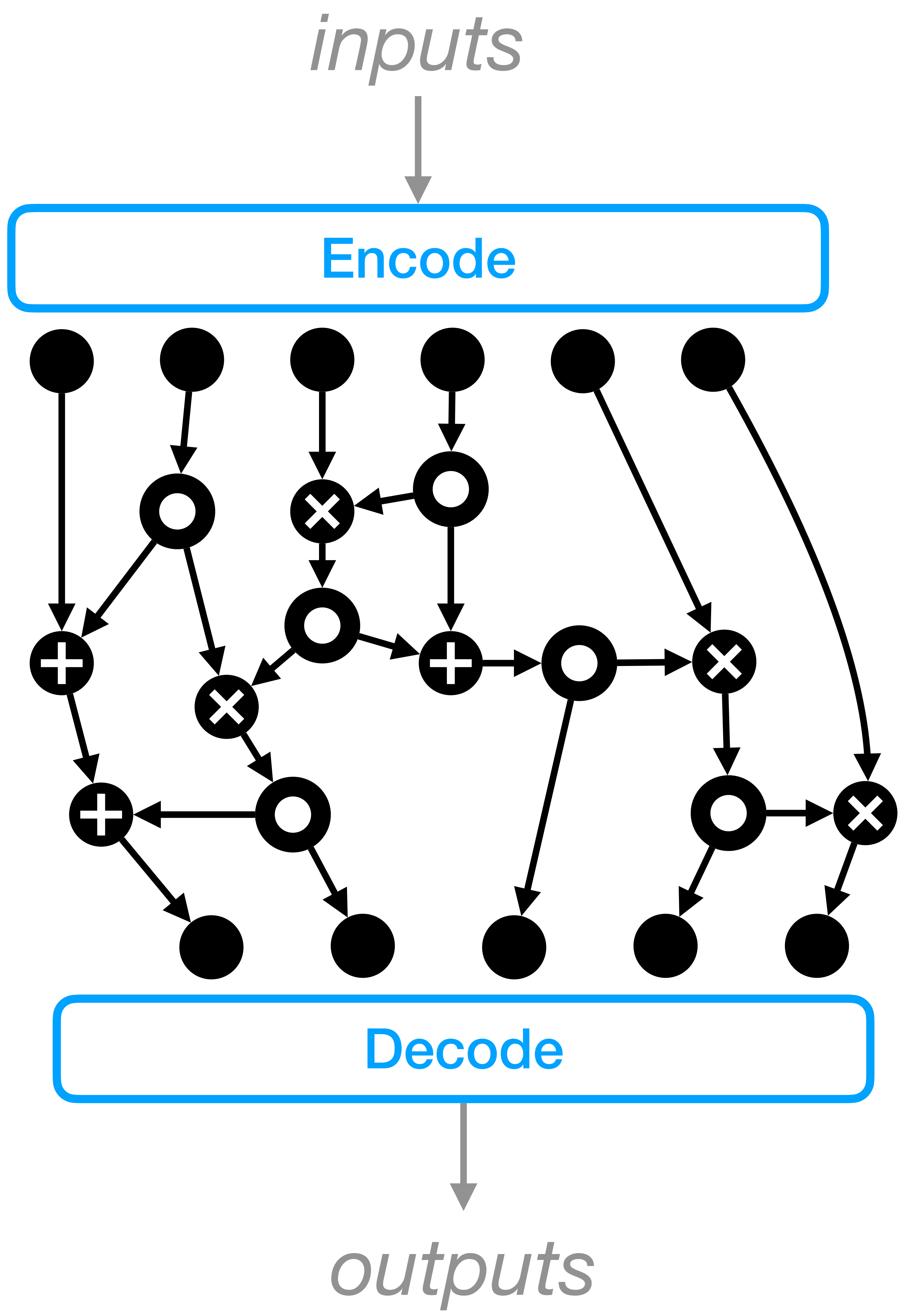
… with full refreshing

⇒

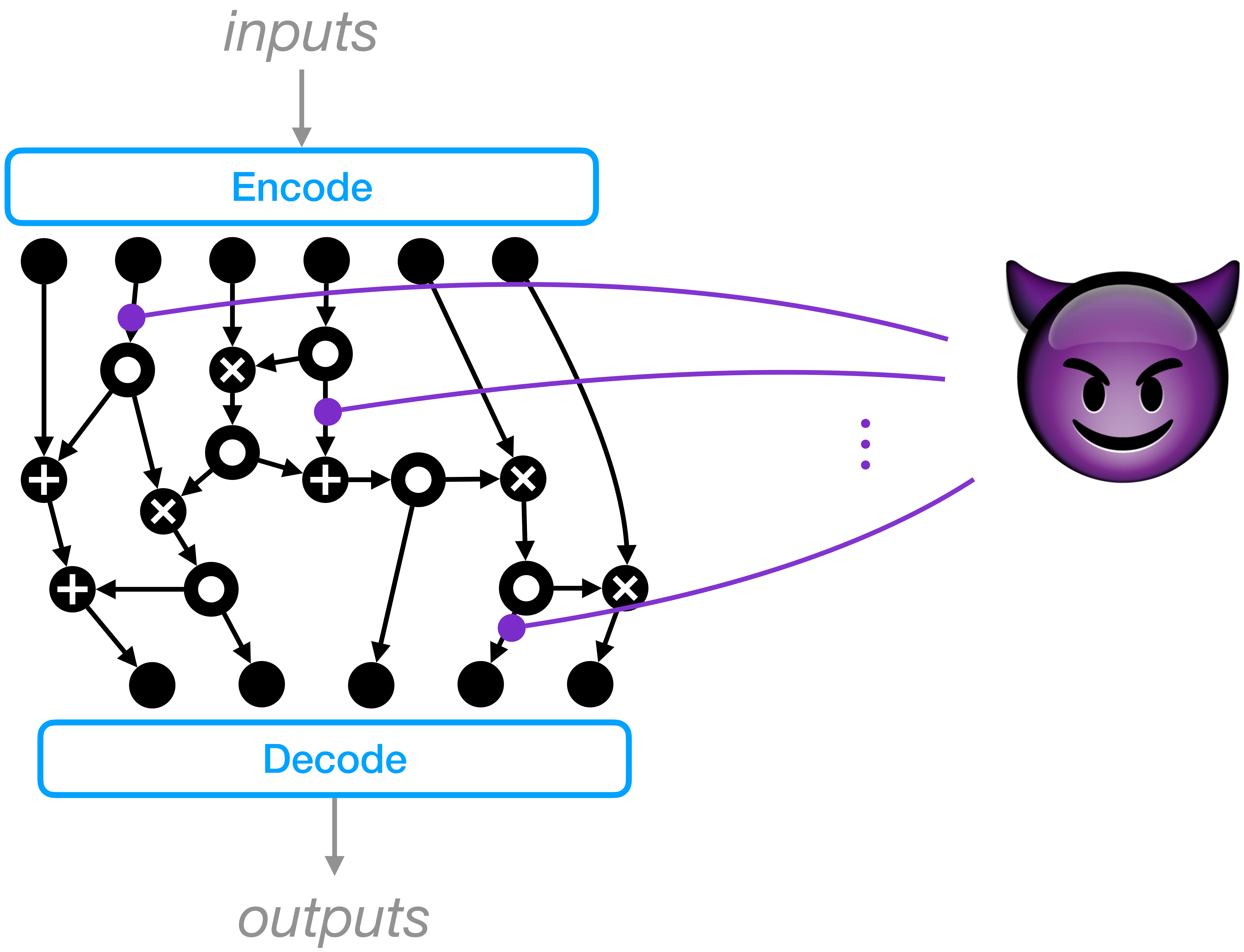
introduce a refresh gadget between any two gadgets



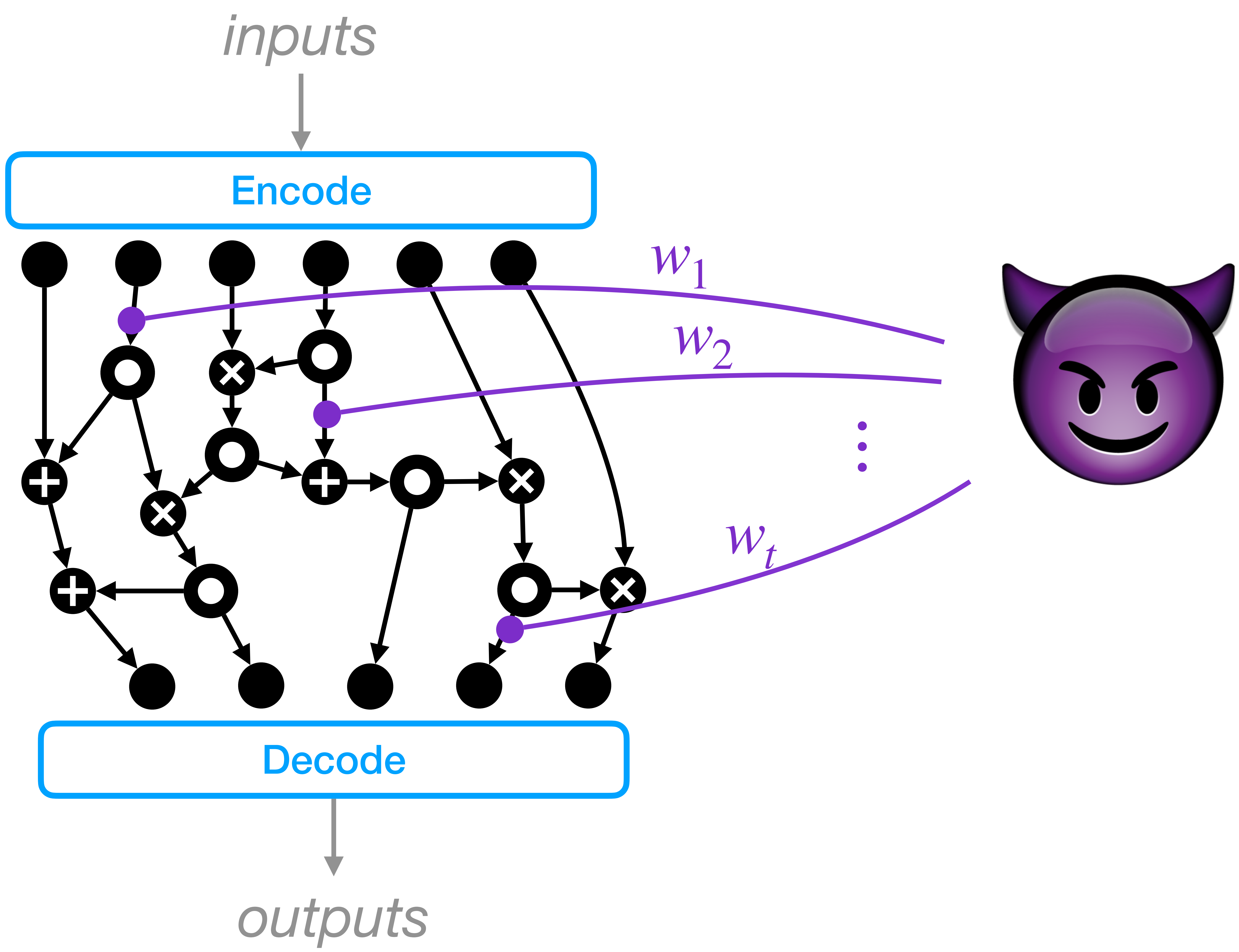
Probing security



Probing security

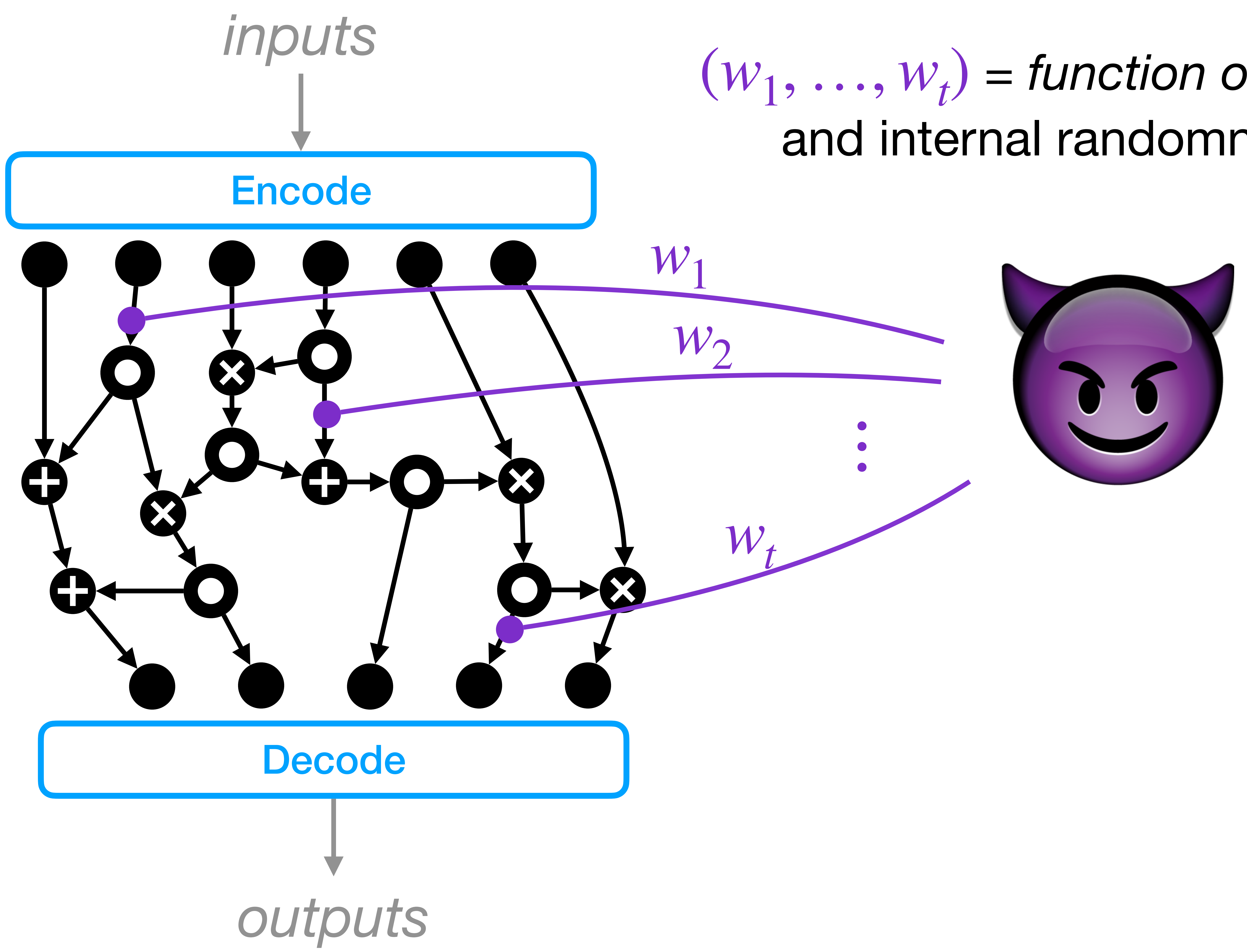


Probing security

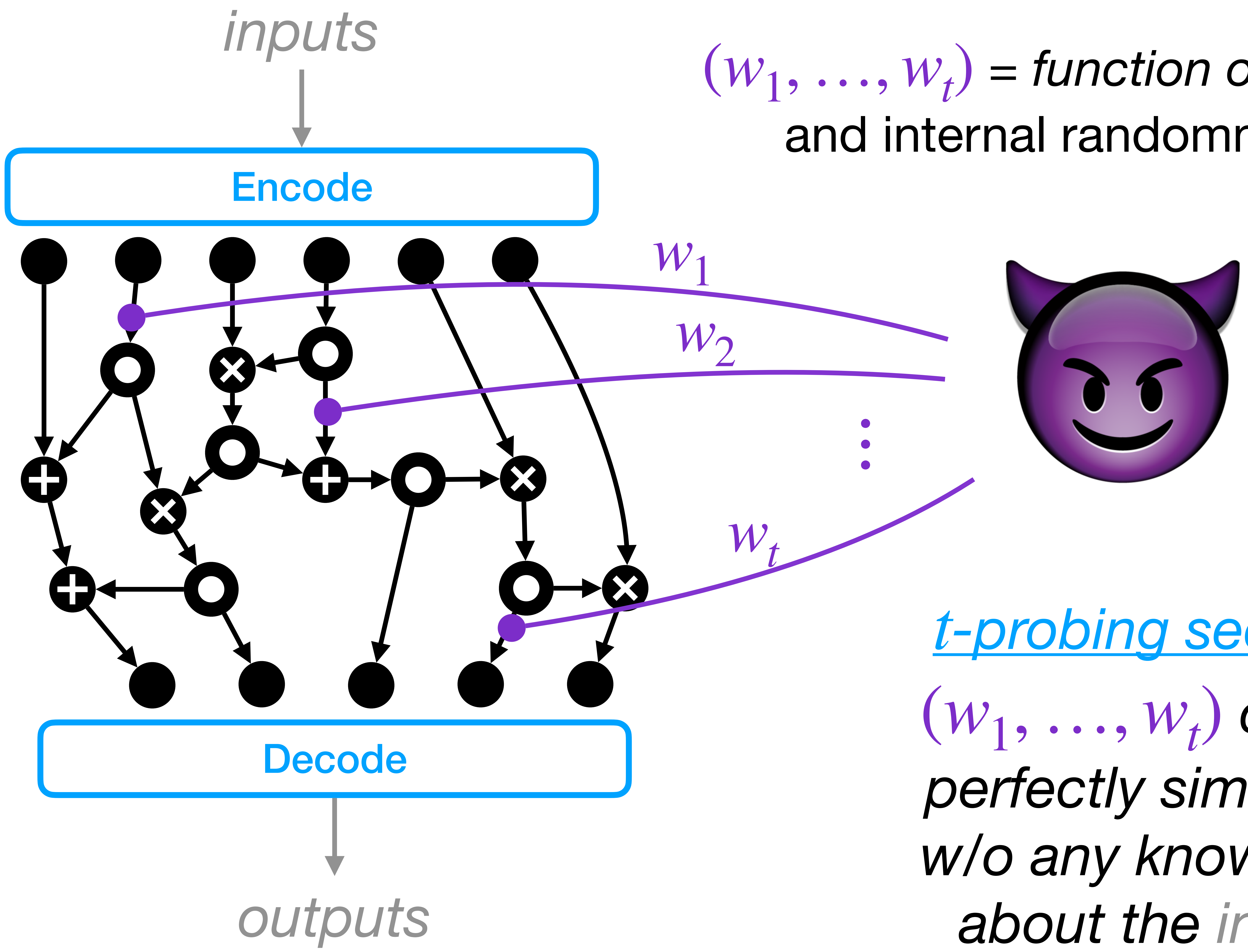


Probing security

$(w_1, \ldots, w_t)$ = *function of inputs* and internal randomness



Probing security

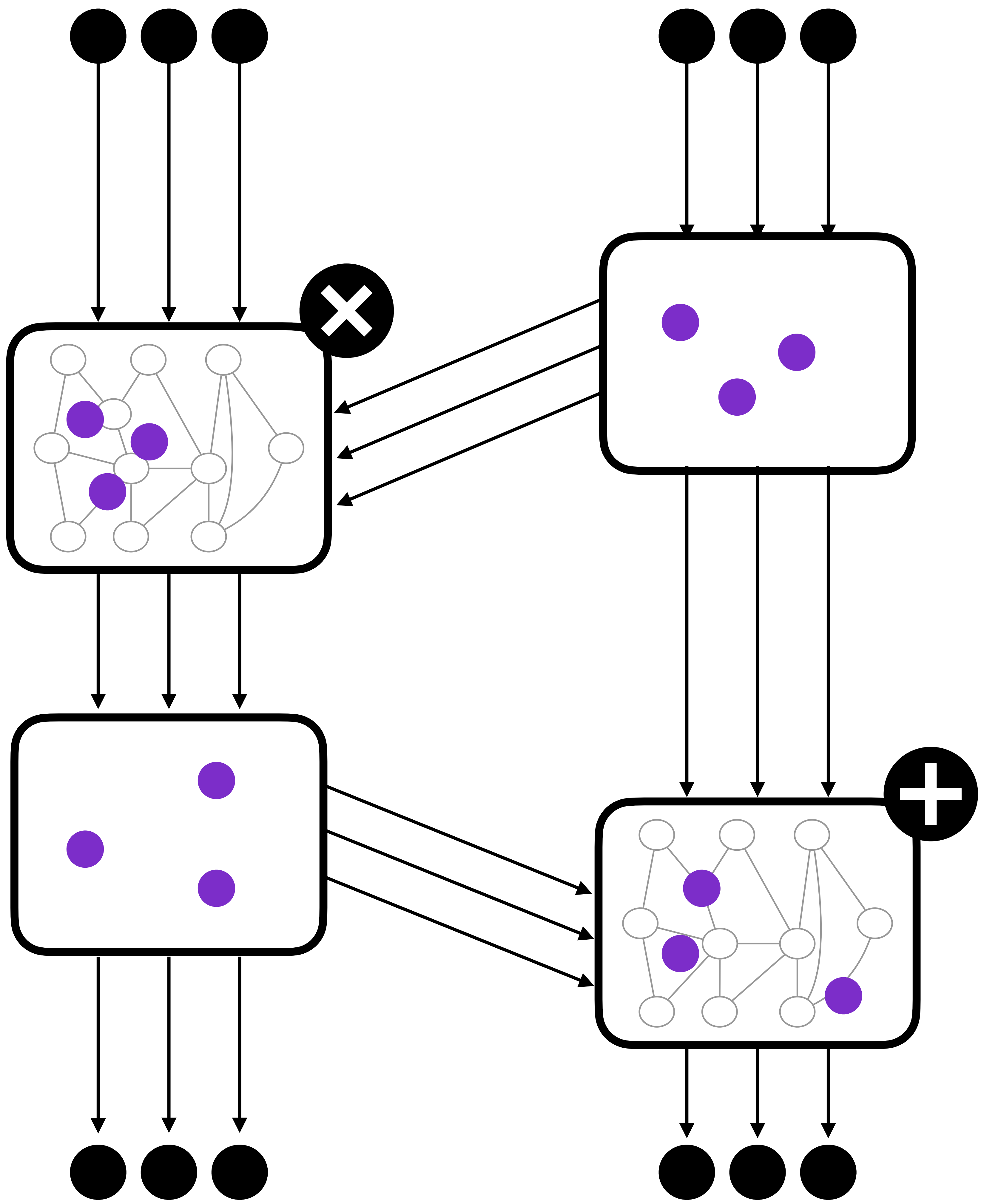


$(w_1, \ldots, w_t)$ = *function of* inputs and internal randomness

t-probing security:

$(w_1, \ldots, w_t)$ *can be perfectly simulated w/o any knowledge about the* inputs

Region probing security



t probes per gadget (or region)

with $t = r \times |G|$

rate

number of wires in G

$\Rightarrow$

r-region probing security

Why region probing security?

r-region probing security

$\Rightarrow$ p-random probing security

$\Rightarrow$ δ-noisy leakage security

Why region probing security?

r-region probing security

Each wire leak with probability p

$\Rightarrow$ p-random probing security

$\Rightarrow$ δ-noisy leakage security

Why region probing security?

r-region probing security

Each wire leak with probability p

$\Rightarrow$ p-random probing security

Each wire leak some noisy information

$\Rightarrow$ δ-noisy leakage security

Why region probing security?

r-region probing security

Each wire leak with probability p

$\Rightarrow$ p-random probing security

$p \approx r$

Chernoff bound

Each wire leak some noisy information

$\Rightarrow$ δ-noisy leakage security

Why region probing security?

r-region probing security

Each wire leak with probability p

$\Rightarrow$ p-random probing security

$p \approx r$

Chernoff bound

Each wire leak some noisy information

$\Rightarrow$ δ-noisy leakage security

$\delta \approx p \approx r$

Duc-Dziembowski-Faust [EC'14]

Why region probing security?

r-region probing security

Each wire leak with probability p

$\Rightarrow$ p-random probing security

$p \approx r$

Chernoff bound

Each wire leak some noisy information

$\Rightarrow$ δ-noisy leakage security

$\delta \approx p \approx r$

Duc-Dziembowski-Faust [EC'14]

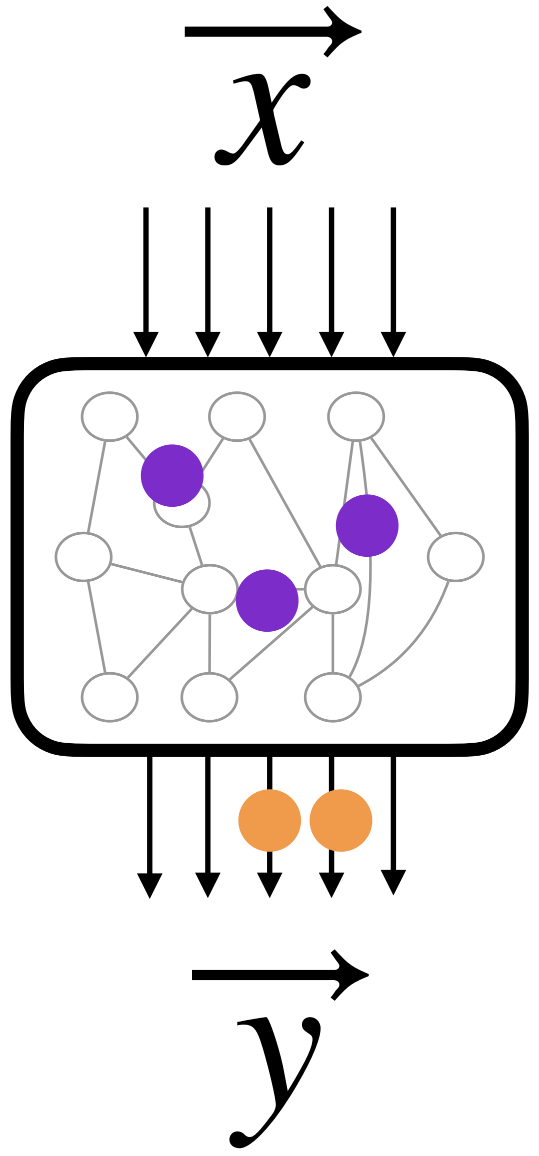
more realistic to capture power and EM leakages

Composition

- Use gadgets achieving composition properties (stronger than PS)
- Obtain the (region) PS for the composition

Composition

- Use gadgets achieving composition properties (stronger than PS)
- Obtain the (region) PS for the composition
- Example: strong non-interference (SNI) notion

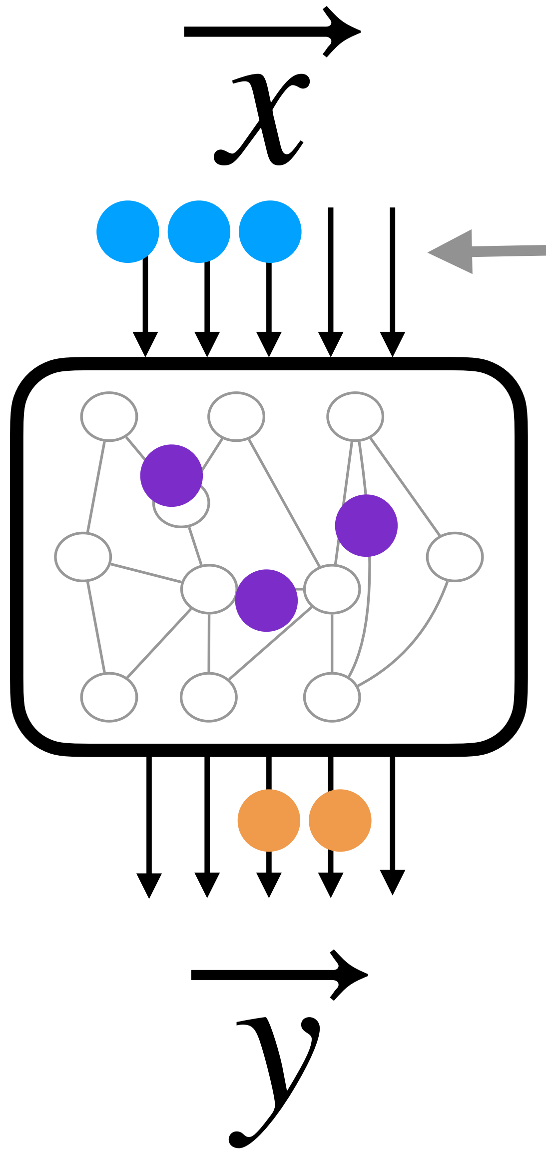


t_1 *internal probes*

t_2 *output probes*

Composition

- Use gadgets achieving composition properties (stronger than PS)
- Obtain the (region) PS for the composition
- Example: strong non-interference (SNI) notion



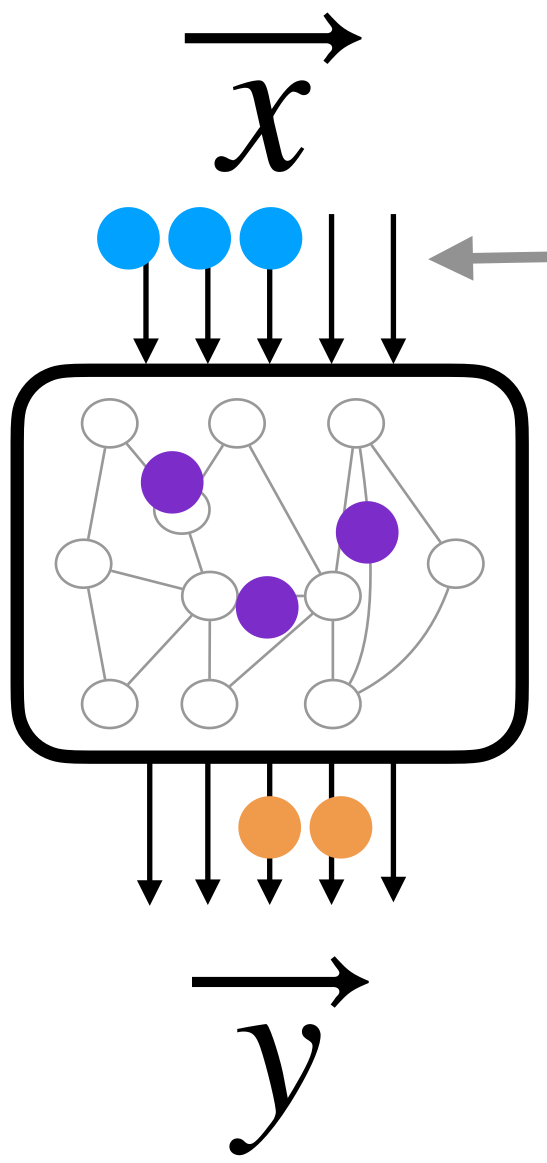
t_1 *internal probes*

t_2 *output probes*

can be perfectly simulated from the knowledge of t_1 *input shares*

Composition

- Use gadgets achieving composition properties (stronger than PS)
- Obtain the (region) PS for the composition
- Example: strong non-interference (SNI) notion



t_1 internal probes

t_2 output probes

can be perfectly simulated from the knowledge of t_1 input shares

SNI gadgets $\Rightarrow$ $(n-1)$-PS

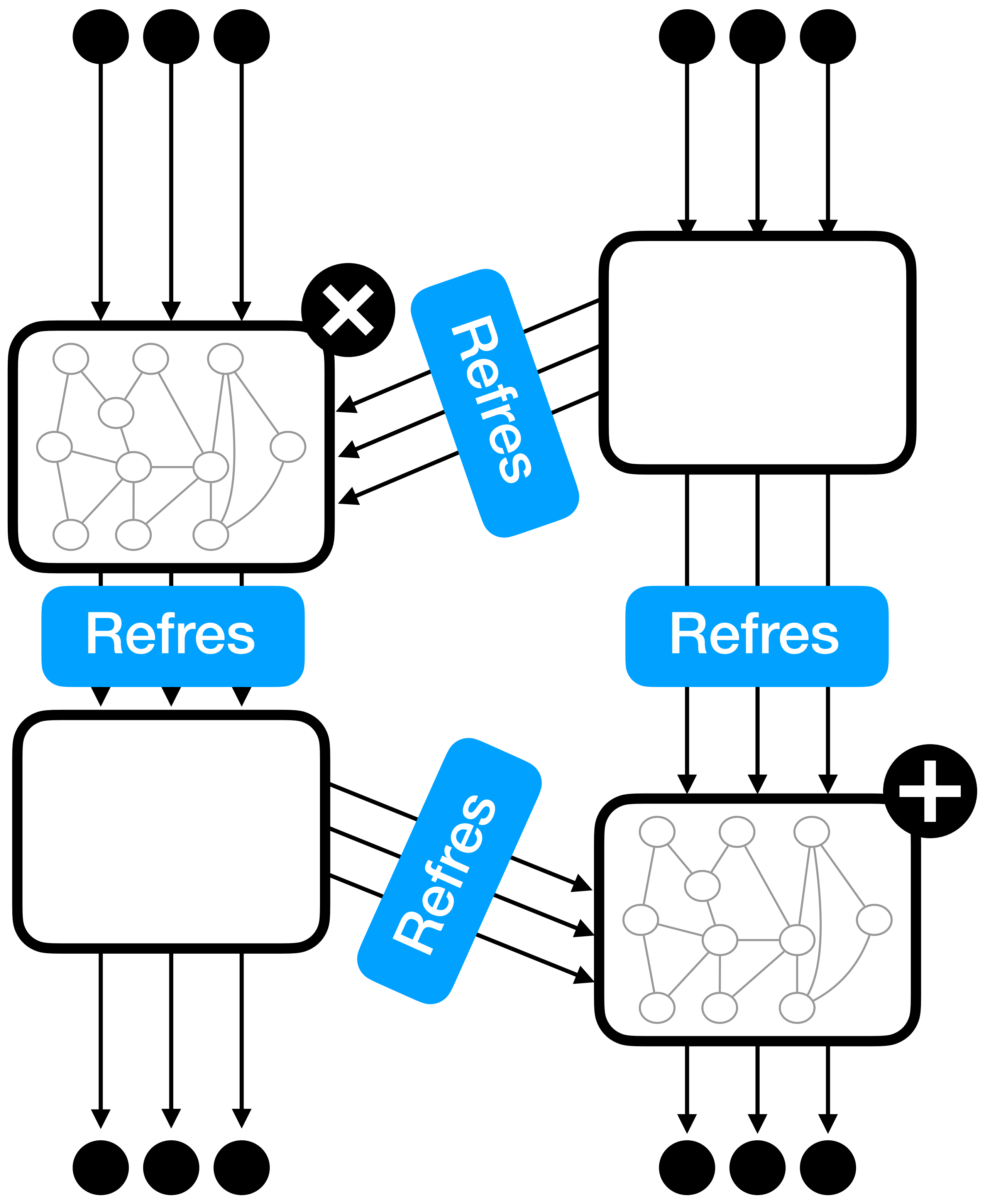
$\Rightarrow$ region PS

Our composition approach

- We only require a composition property for the refresh gadget
- Other gadgets only need to be probing secure

Our composition approach

- We only require a composition property for the refresh gadget
- Other gadgets only need to be probing secure
- We use full refreshing



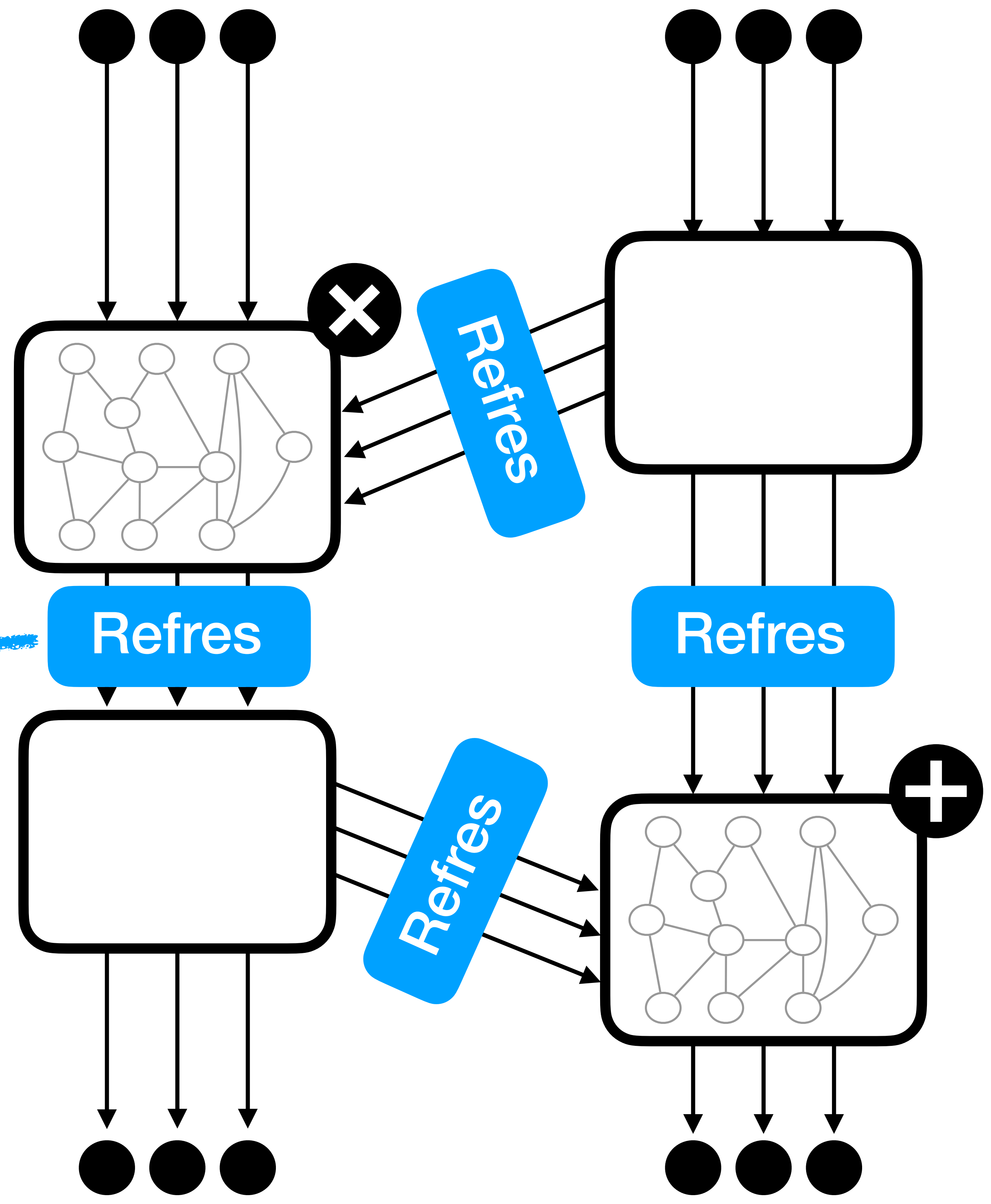
Our composition approach

- We only require a composition property for the refresh gadget
- Other gadgets only need to be probing secure
- We use full refreshing

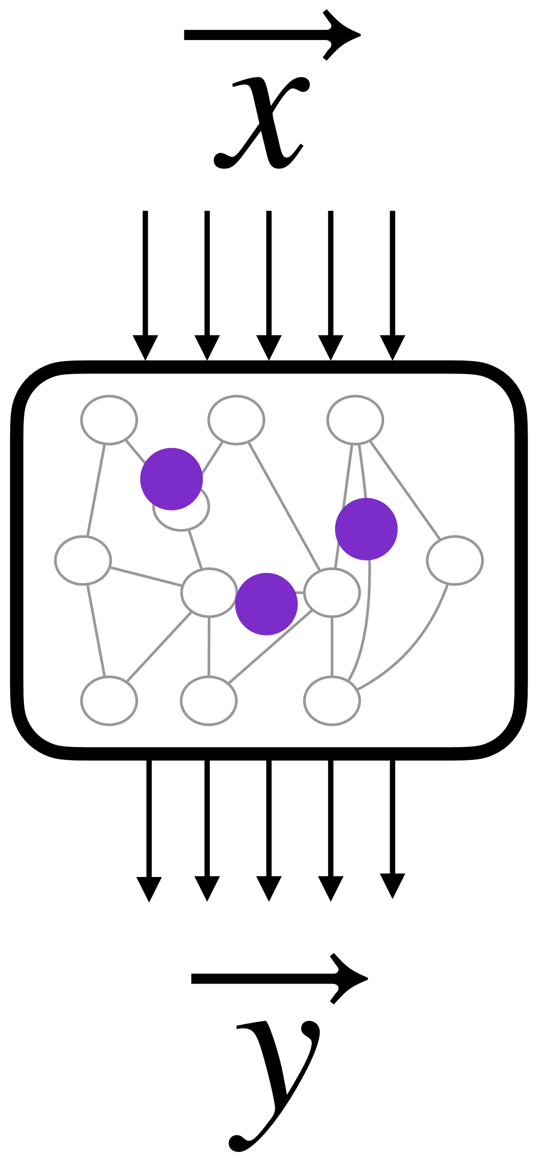
simple probing security

IOS property + uniformity

⇒ region probing security

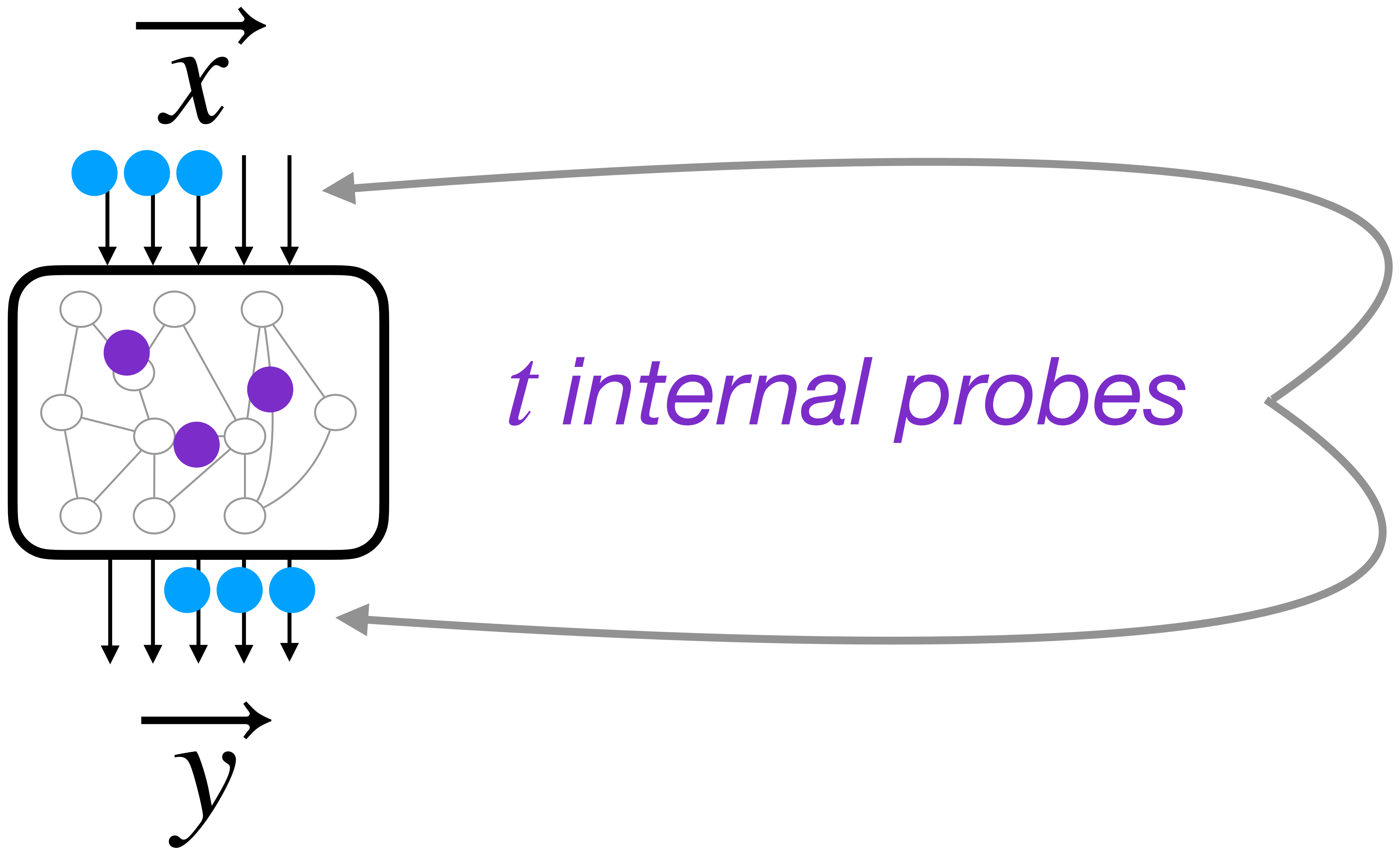


Input-Output Separation (IOS)



t internal probes

Input-Output Separation (IOS)



t internal probes

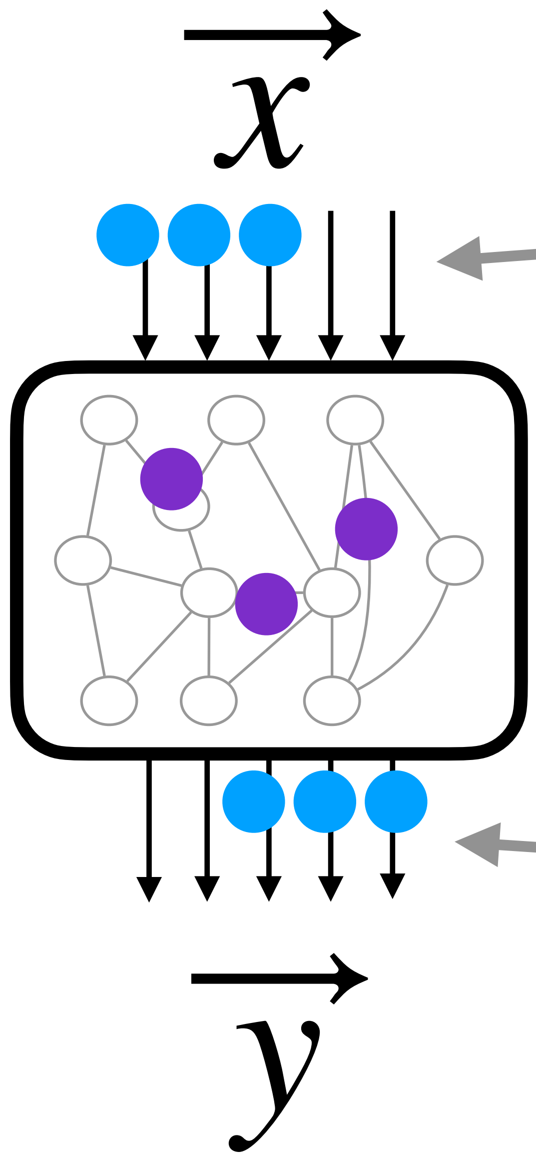
can be perfectly simulated from the knowledge of

t input shares

and

t output shares

Input-Output Separation (IOS)



t internal probes

can be perfectly simulated from the knowledge of

t input shares

and

t output shares

IOS is weaker than previous composition notions

NI (+uniformity)

SNI (+uniformity) $\Rightarrow$ IOS (+uniformity)

PINI (+uniformity)

SNI $\Rightarrow$ NI $\Rightarrow$ IOS; SNI $\Rightarrow$ PINI $\Rightarrow$ IOS

Composition theorem

t_R *probes per refresh gadget*

+ t_{op} *probes per operation gadget*

can be perfectly simulated from

$t_{op} + 3t_R$ *probes per operation gadget*

IOS refreshing

Composition theorem

t_R probes per refresh gadget

$+ \ t_{op}$ probes per operation gadget

can be perfectly simulated from

IOS refreshing

$t_{op} + 3t_R$ probes per operation gadget

can be perfectly simulated

uniform refreshing

nothing

assuming $(t_{op} + 3t_R)$-PS

of operation gadgets

Composition theorem

Obtained rate:

$$\min\left(\frac{t_R}{|G_R|}, \frac{t_{op}}{|G_{op}|}\right)$$

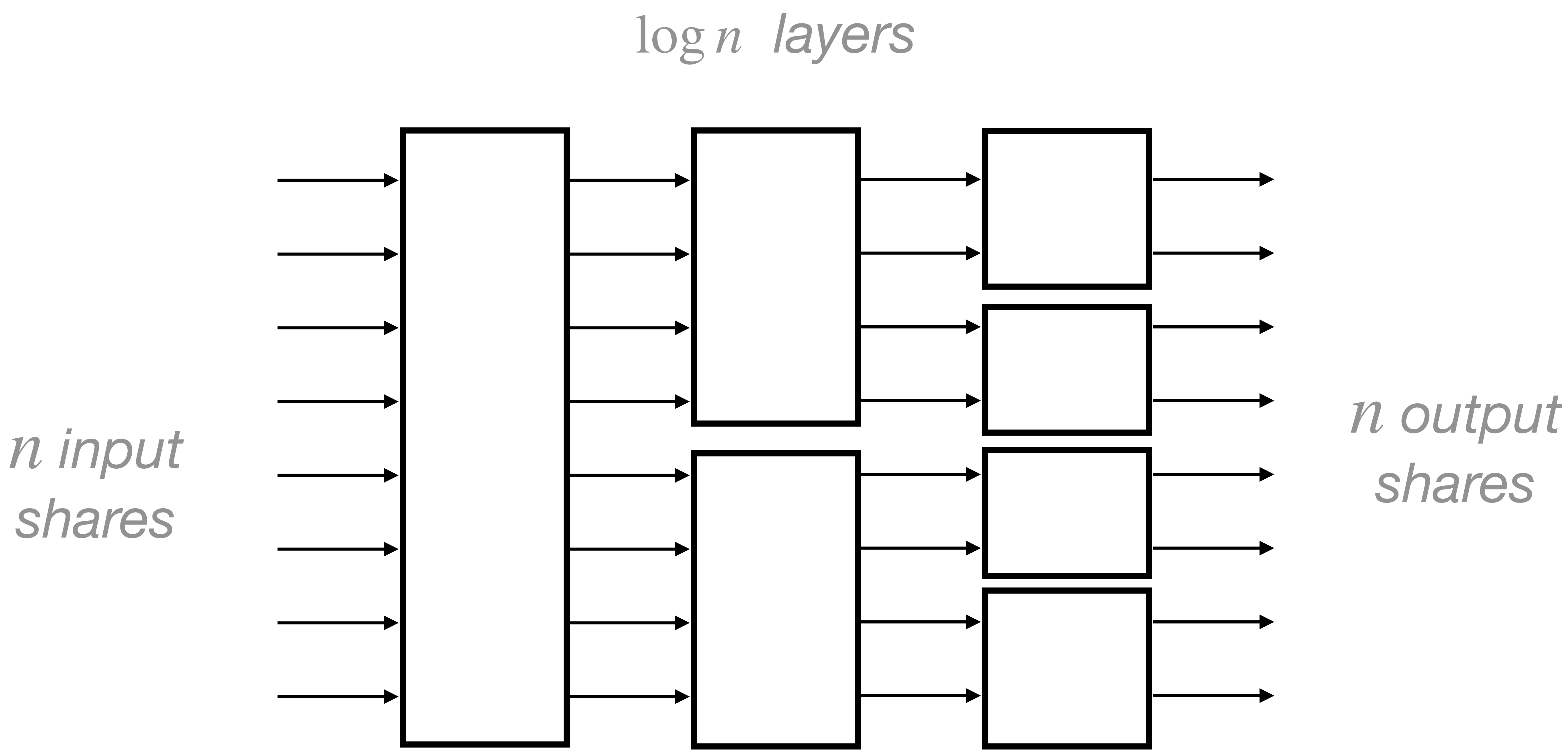
Composition theorem

Obtained rate:

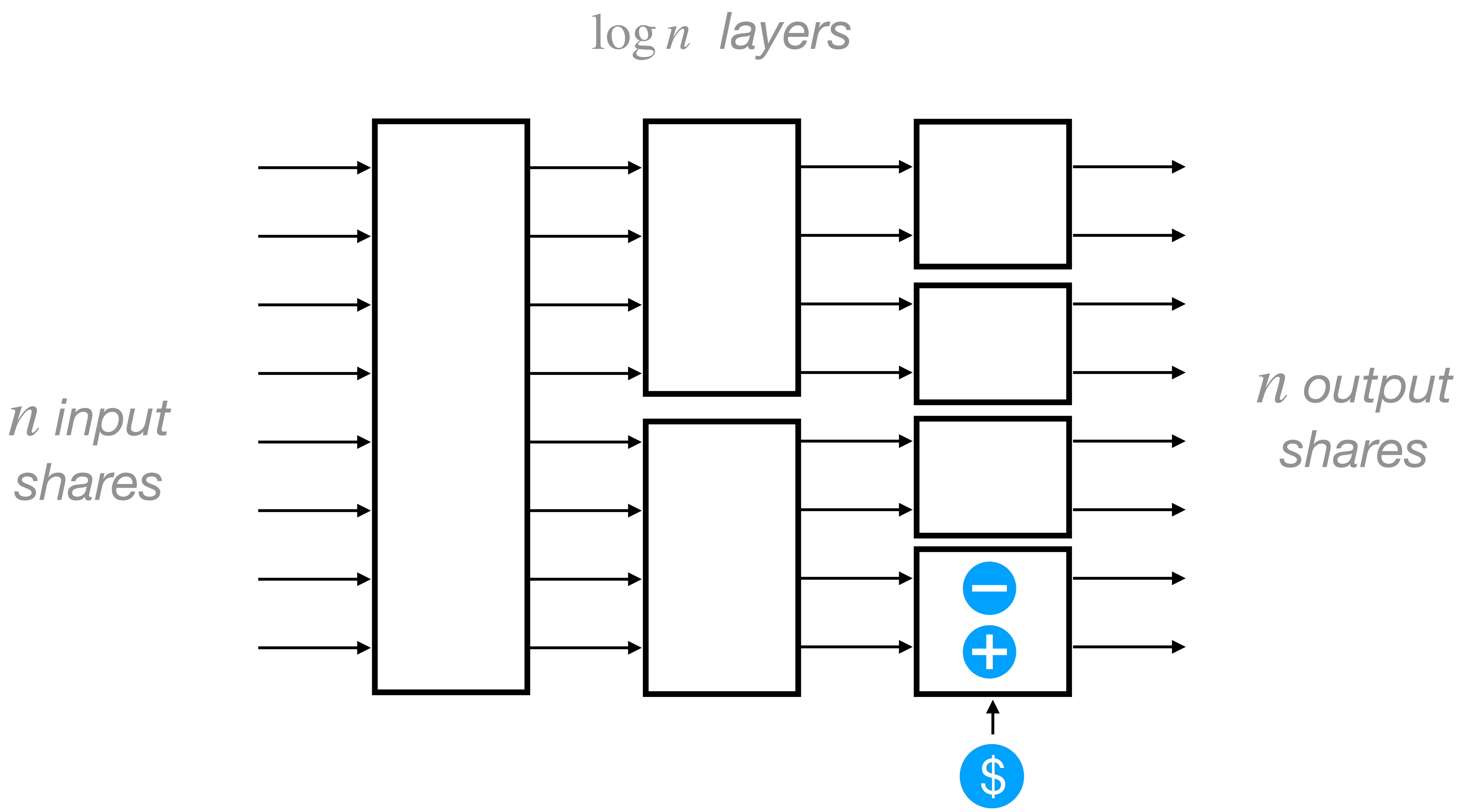
$$\max_{t_R, t_{op}} \quad \min\left(\frac{t_R}{|G_R|}, \frac{t_{op}}{|G_{op}|}\right)$$

with $\quad t_R < n \quad$ and $\quad (t_{op} + 3t_R) \leq t_{PS}$

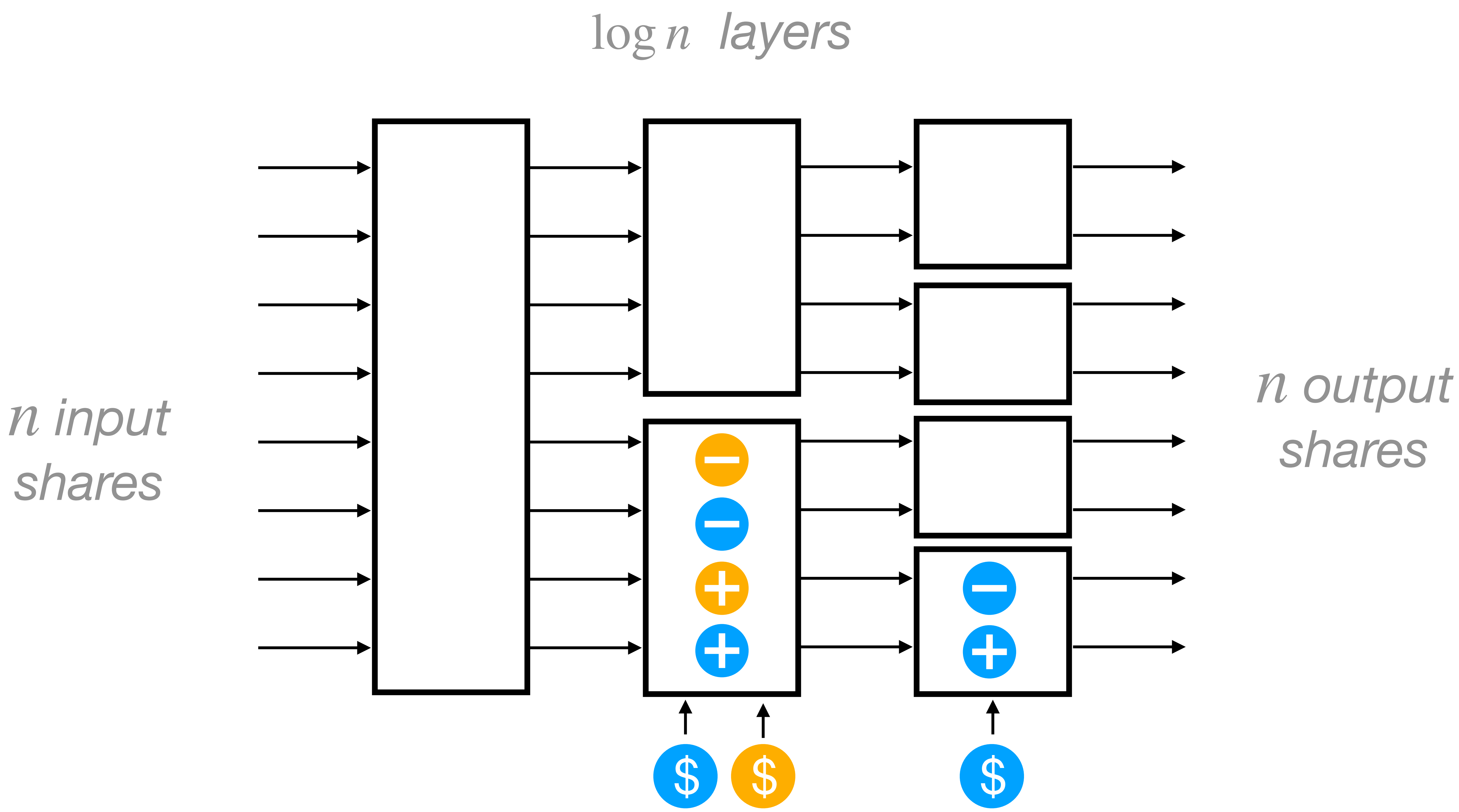
An IOS refresh gadget



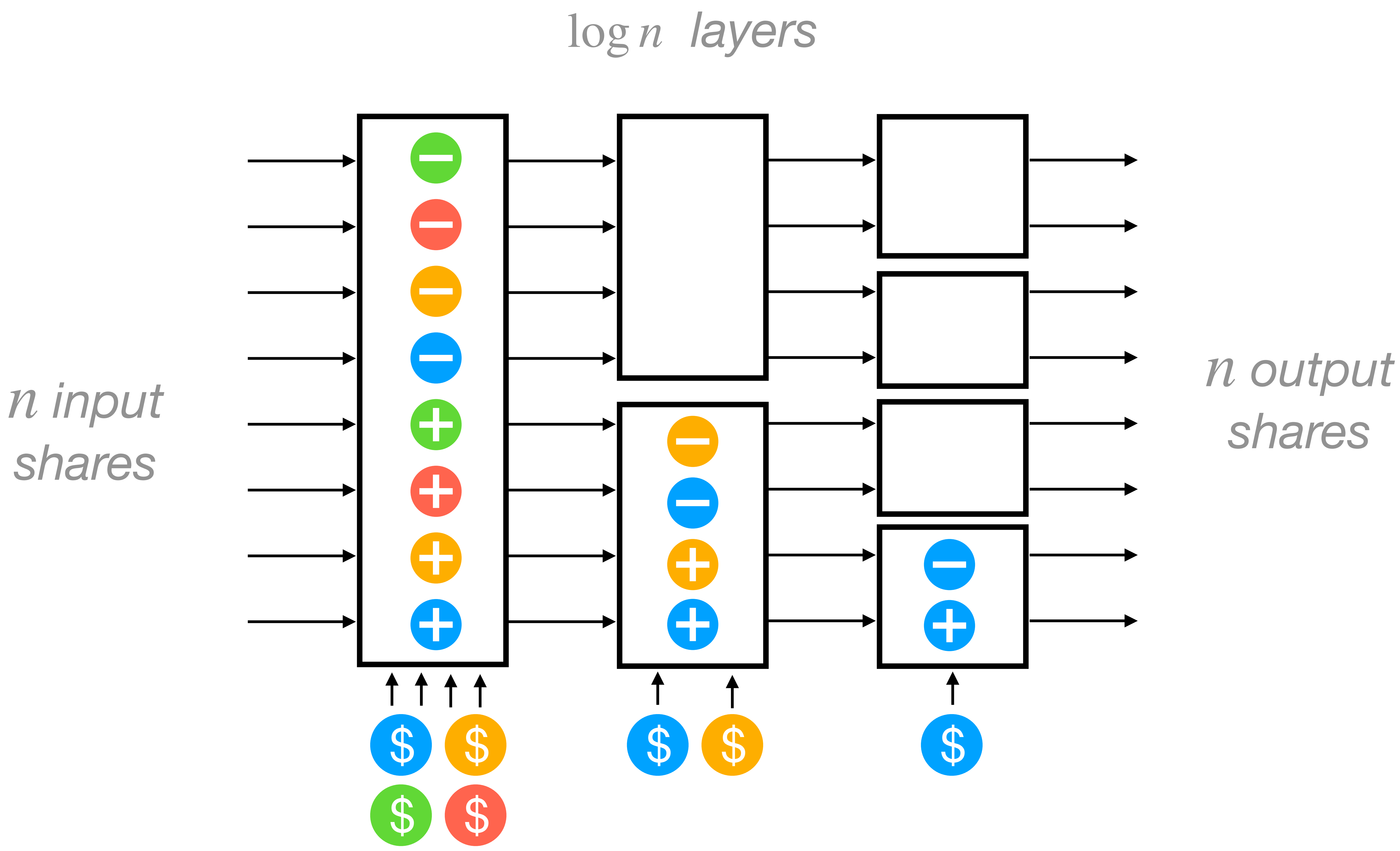
An IOS refresh gadget



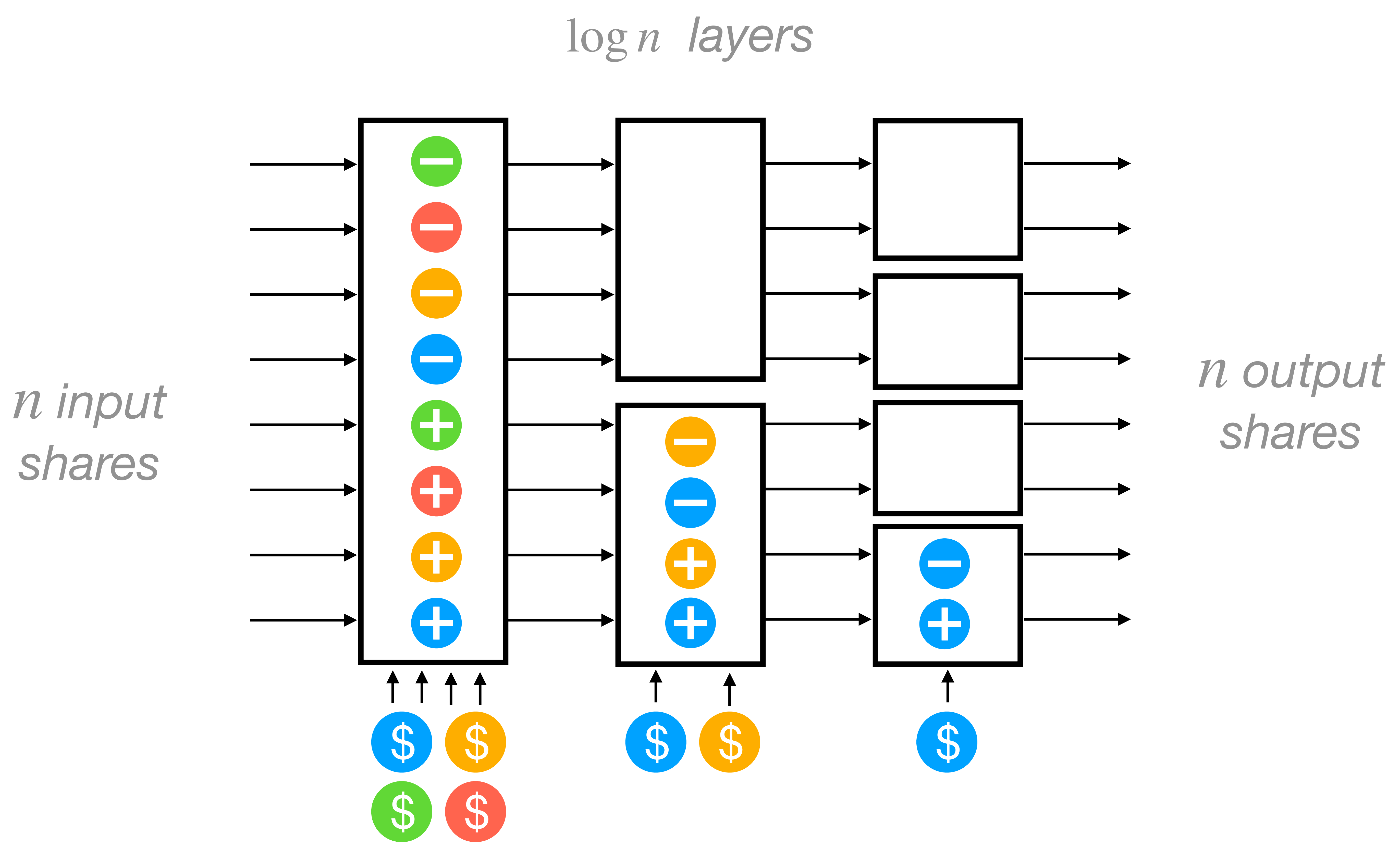
An IOS refresh gadget



An IOS refresh gadget

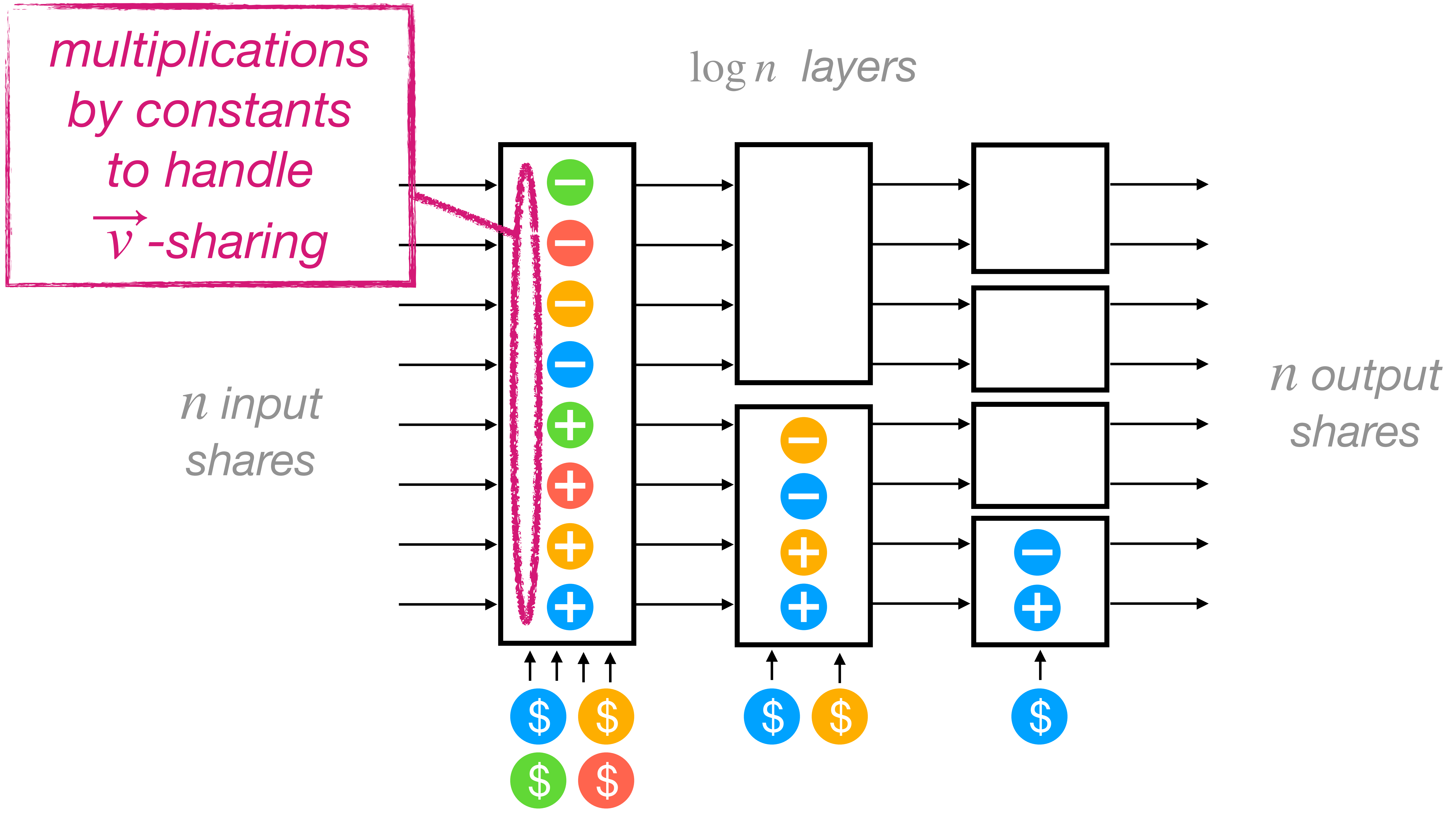


An IOS refresh gadget



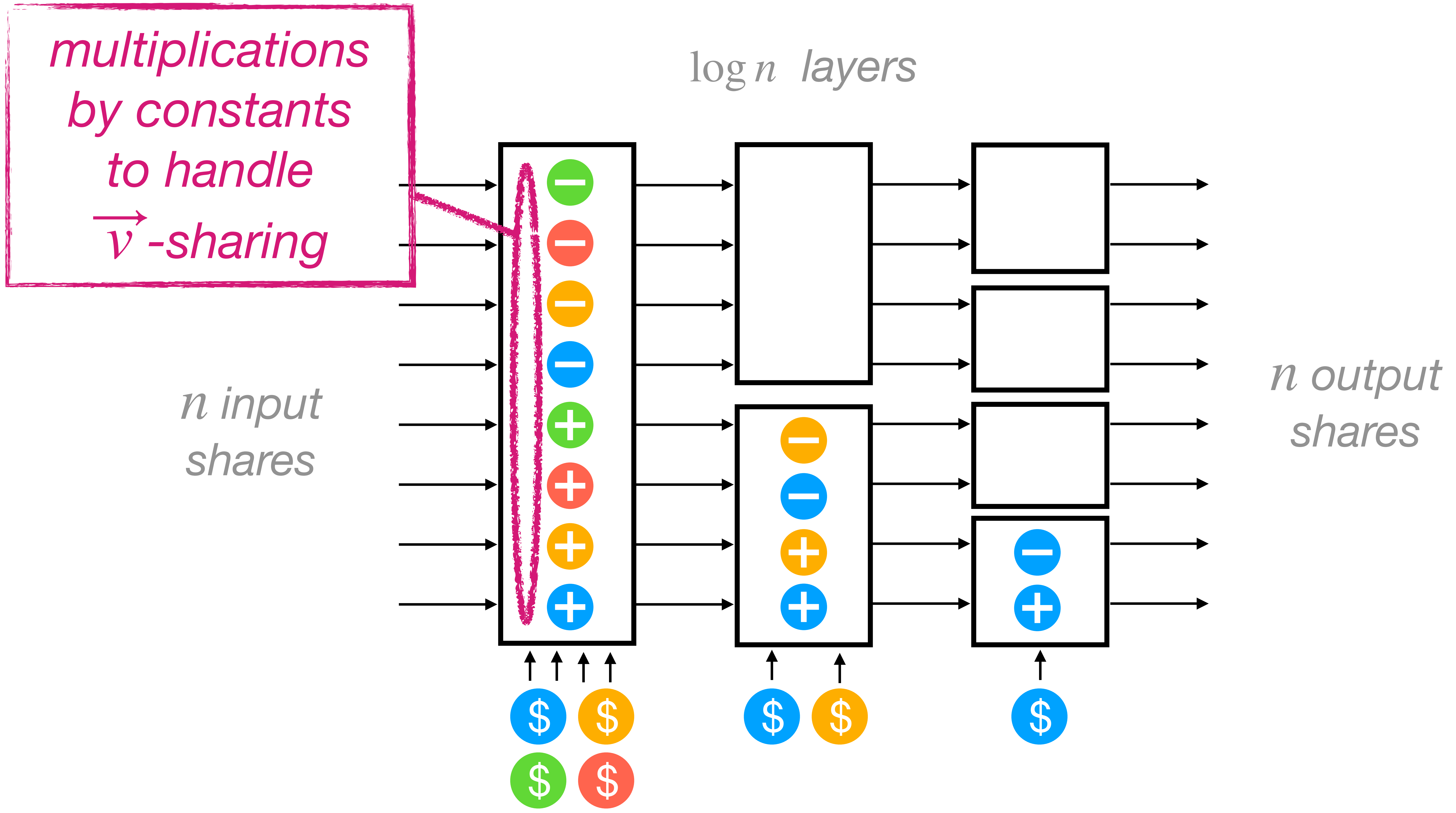
Batistello-Coron-Prouff-Zeitoun
refresh gadget [CHES'16]

An IOS refresh gadget



Batistello-Coron-Prouff-Zeitoun
refresh gadget [CHES'16]

An IOS refresh gadget



Batistello-Coron-Prouff-Zeitoun refresh gadget [CHES'16]

Only half of the layers for IOS

Quasilinear masking

- We extend the Goudarzi-Joux-Rivain (GJR) scheme [AC'18]
  - complexity $O(n \log n)$ against $O(n^2)$ for many probing secure scheme
  - proof of p-random probing security with $p = O(1/\log n)$
  - defined over fields $\mathbb{F}_p$ with $p = 2^{\lceil \log n \rceil + 1}\alpha + 1$
- Our extension enjoys
  - base field $\mathbb{K}$ of any form
  - proof in the (stronger) r-region probing model (still with $r = O(1/\log n)$)
  - we patch a flaw in the security proof thanks to the IOS approach

Quasilinear masking

- GJR scheme uses $\overrightarrow{v}$-sharings with

$$\overrightarrow{v} = (1,\omega, \omega^2, \ldots, \omega^{n-1})$$

- A sharing of x

$$\overrightarrow{x} = (x_0, x_1, \ldots, x_{n-1})$$

satisfies

$$\langle \overrightarrow{v}, \overrightarrow{x} \rangle = \sum_{i=0}^{n-1} x_i \cdot \omega^i = x$$

Quasilinear masking

- GJR scheme uses $\overrightarrow{v}$-sharings with

$$\overrightarrow{v} = (1, \omega, \omega^2, \ldots, \omega^{n-1})$$

- A sharing of x

$$\overrightarrow{x} = (x_0, x_1, \ldots, x_{n-1})$$

satisfies

$$\langle \overrightarrow{v}, \overrightarrow{x} \rangle = \sum_{i=0}^{n-1} x_i \cdot \omega^i = x$$

polynomial $P_{\overrightarrow{x}}(\omega)$

shares = coefficients

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

- We get

$$\sum_{i=0}^{2n-1} t_i \omega^i = P_{\vec{t}}(\omega) = P_{\vec{x}}(\omega) \cdot P_{\vec{y}}(\omega) = x \cdot y$$

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

- We get

$$\sum_{i=0}^{2n-1} t_i \omega^i = P_{\vec{t}}(\omega) = P_{\vec{x}}(\omega) \cdot P_{\vec{y}}(\omega) = x \cdot y$$

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

- We get

$$\sum_{i=0}^{2n-1} t_i \omega^i = P_{\vec{t}}(\omega) = P_{\vec{x}}(\omega) \cdot P_{\vec{y}}(\omega) = x \cdot y$$

$\vec{t}$ is a $(\underbrace{1, \ldots, \omega^{n-1}}_{\vec{v}}, \omega^n, \ldots, \omega^{2n-1})$-sharing of $x \cdot y$

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

- We get

$$\sum_{i=0}^{2n-1} t_i \omega^i = P_{\vec{t}}(\omega) = P_{\vec{x}}(\omega) \cdot P_{\vec{y}}(\omega) = x \cdot y$$

$\vec{t}$ is a $(\underbrace{1,\ldots,\omega^{n-1}}_{\vec{v}}, \omega^n, \ldots, \omega^{2n-1})$-sharing of $x \cdot y$

- Compression:

$$\vec{z} = (t_0, \ldots, t_{n-1}) + \omega^n \cdot (t_n, \ldots, t_{2n-1})$$

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

- We get

$$\sum_{i=0}^{2n-1} t_i \omega^i = P_{\vec{t}}(\omega) = P_{\vec{x}}(\omega) \cdot P_{\vec{y}}(\omega) = x \cdot y$$

$\vec{t}$ is a $\underbrace{(1,\ldots,\omega^{n-1}}_{\vec{v}}, \omega^n, \ldots, \omega^{2n-1})$-sharing of $x \cdot y$

- Compression:

$$\vec{z} = (t_0, \ldots, t_{n-1}) + \omega^n \cdot (t_n, \ldots, t_{2n-1})$$

$$\sum_{i=0}^{n-1} (t_i + t_{n+i}\,\omega^n)\,\omega^i$$

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

Evaluation-interpolation using FFT

- We get

$$\sum_{i=0}^{2n-1} t_i \omega^i = P_{\vec{t}}(\omega) = P_{\vec{x}}(\omega) \cdot P_{\vec{y}}(\omega) = x \cdot y$$

$\vec{t}$ is a $(\underbrace{1, \ldots, \omega^{n-1}}_{\vec{v}}, \omega^n, \ldots, \omega^{2n-1})$-sharing of $x \cdot y$

- Compression:

$$\vec{z} = (t_0, \ldots, t_{n-1}) + \omega^n \cdot (t_n, \ldots, t_{2n-1})$$

$$\sum_{i=0}^{n-1} (t_i + t_{n+i}\,\omega^n)\,\omega^i$$

Multiplication gadget

- Let $\vec{t}$ such that

$$P_{\vec{t}}(W) = P_{\vec{x}}(W) \cdot P_{\vec{y}}(W)$$

Evaluation-interpolation using FFT

- We get

$$\sum_{i=0}^{2n-1} t_i \omega^i = P_{\vec{t}}(\omega) = P_{\vec{x}}(\omega) \cdot P_{\vec{y}}(\omega) = x \cdot y$$

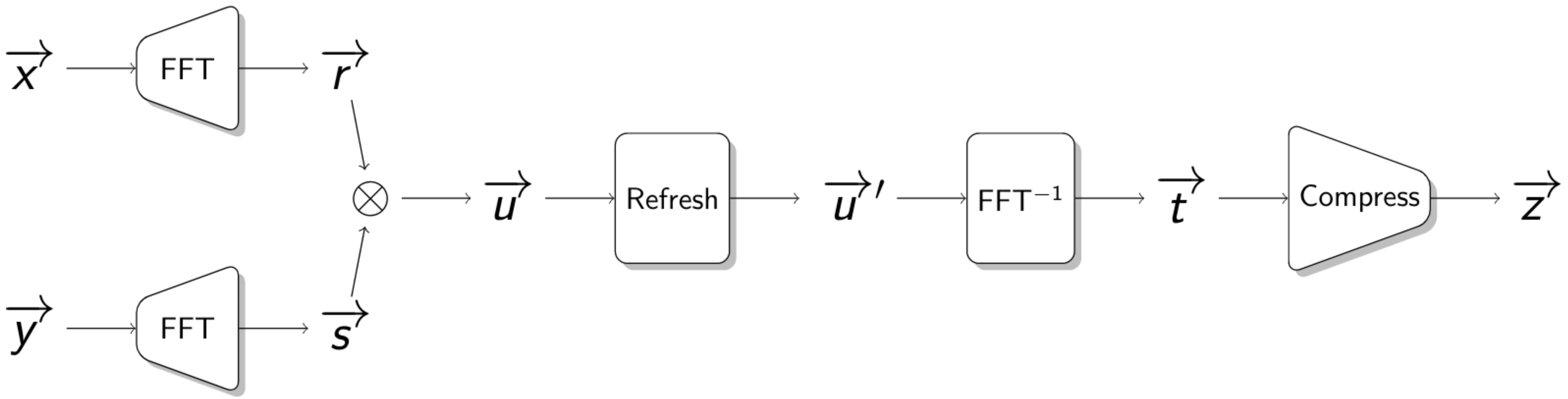
$\vec{t}$ is a $\underbrace{(1,\ldots,\omega^{n-1}}_{\vec{v}}, \omega^n, \ldots, \omega^{2n-1})$-sharing of $x \cdot y$

- Compression:

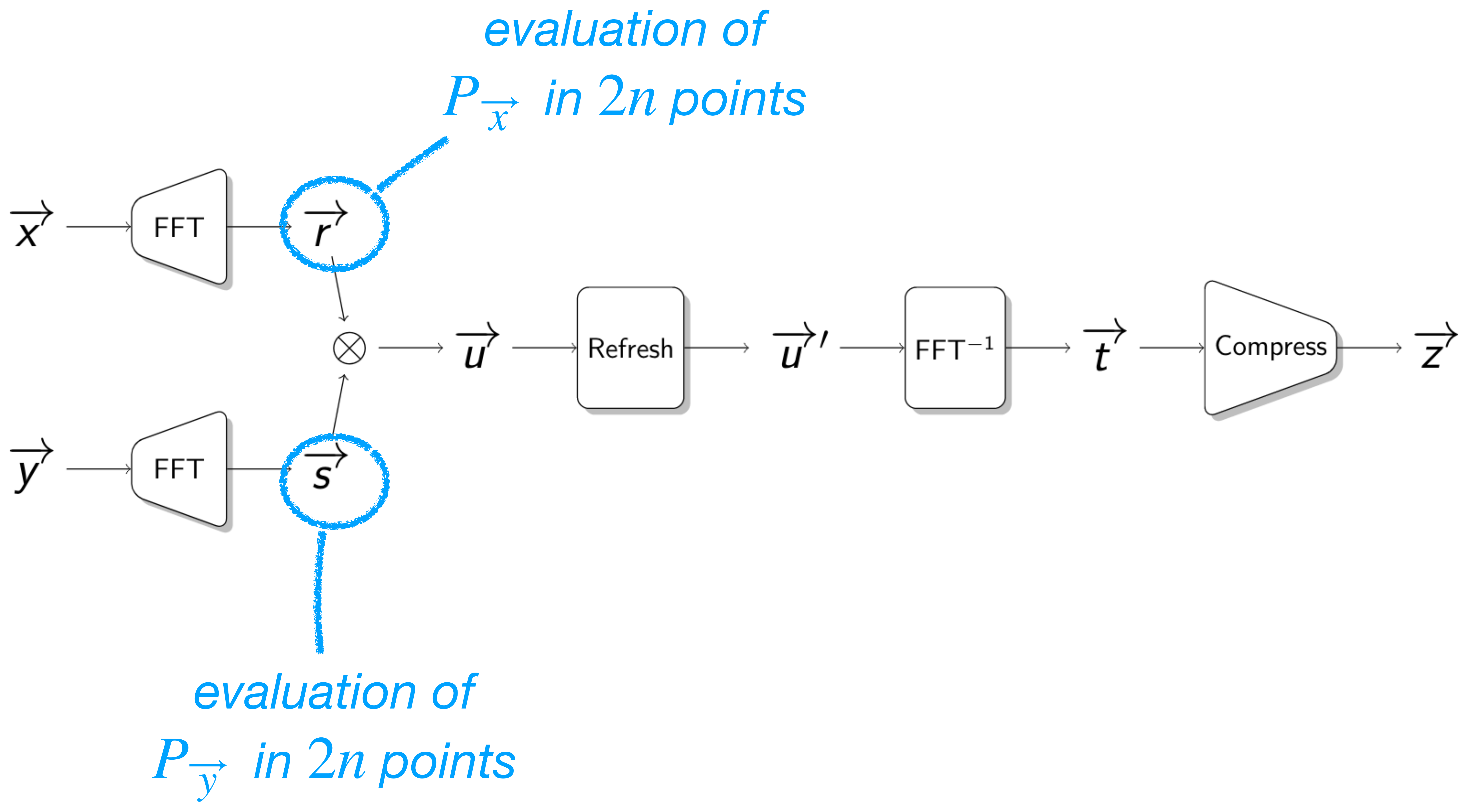
$$\vec{z} = (t_0, \ldots, t_{n-1}) + \omega^n \cdot (t_n, \ldots, t_{2n-1})$$

$$\sum_{i=0}^{n-1} (t_i + t_{n+i}\,\omega^n)\,\omega^i$$

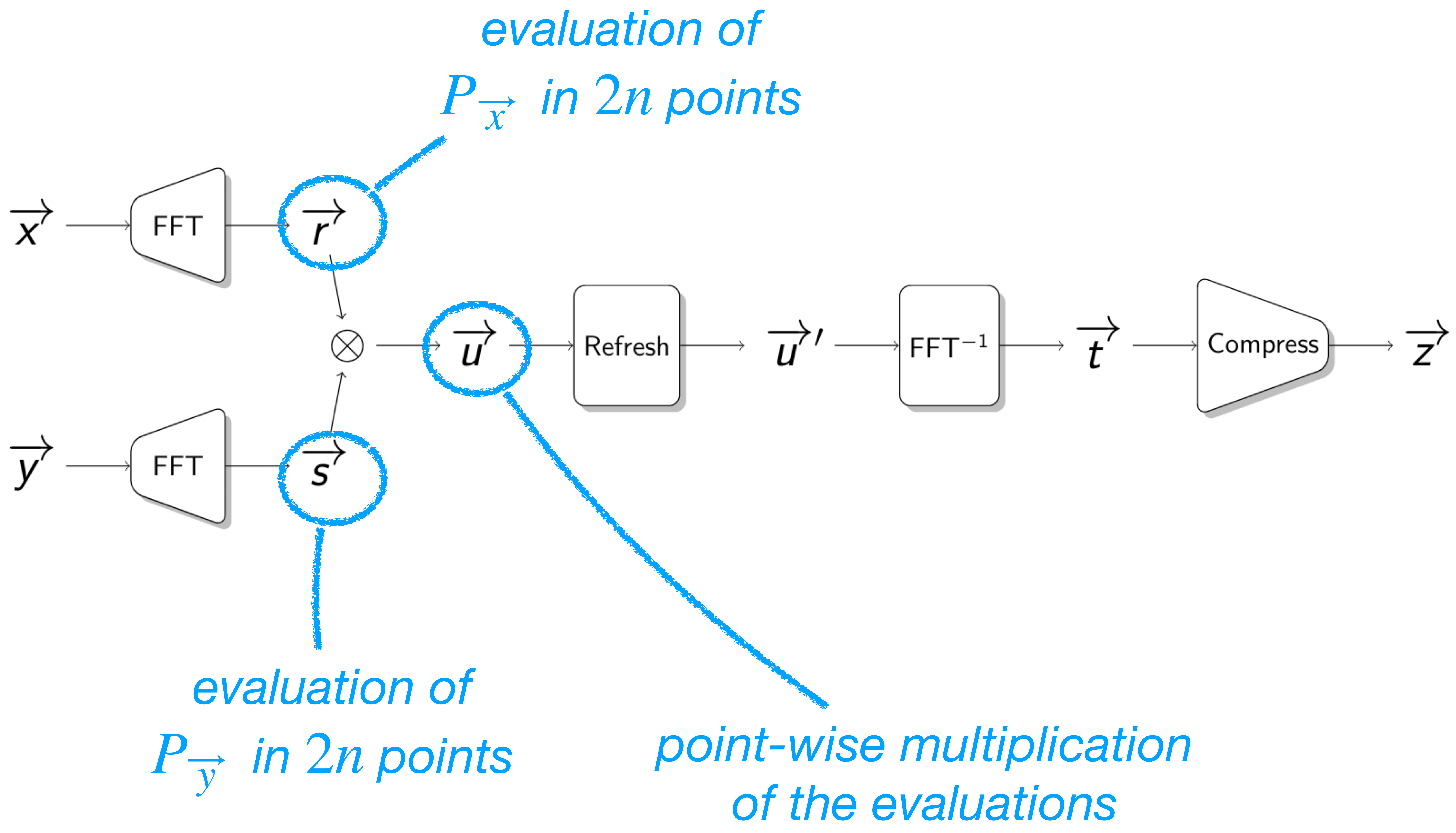
Multiplication gadget



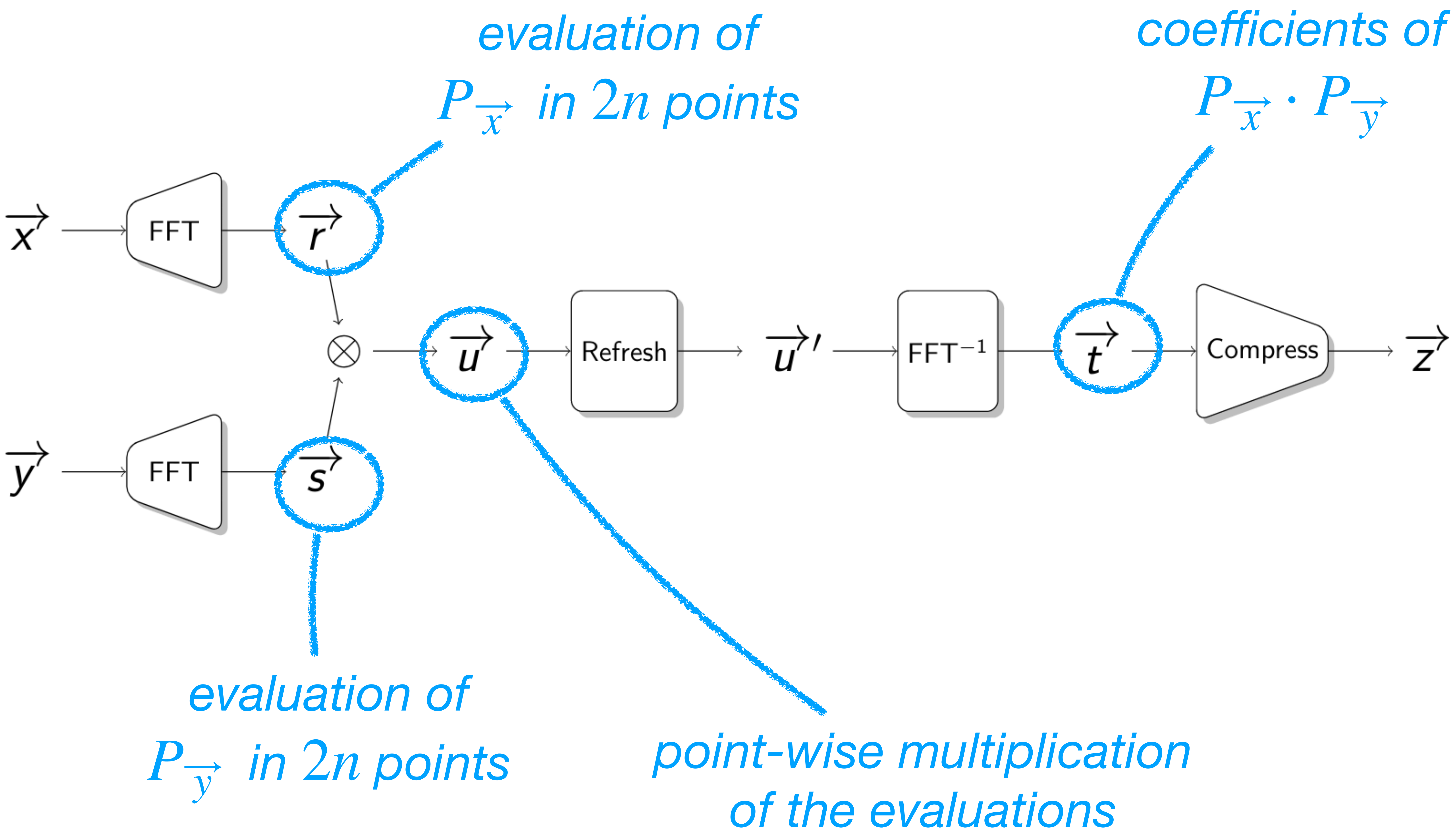
Multiplication gadget



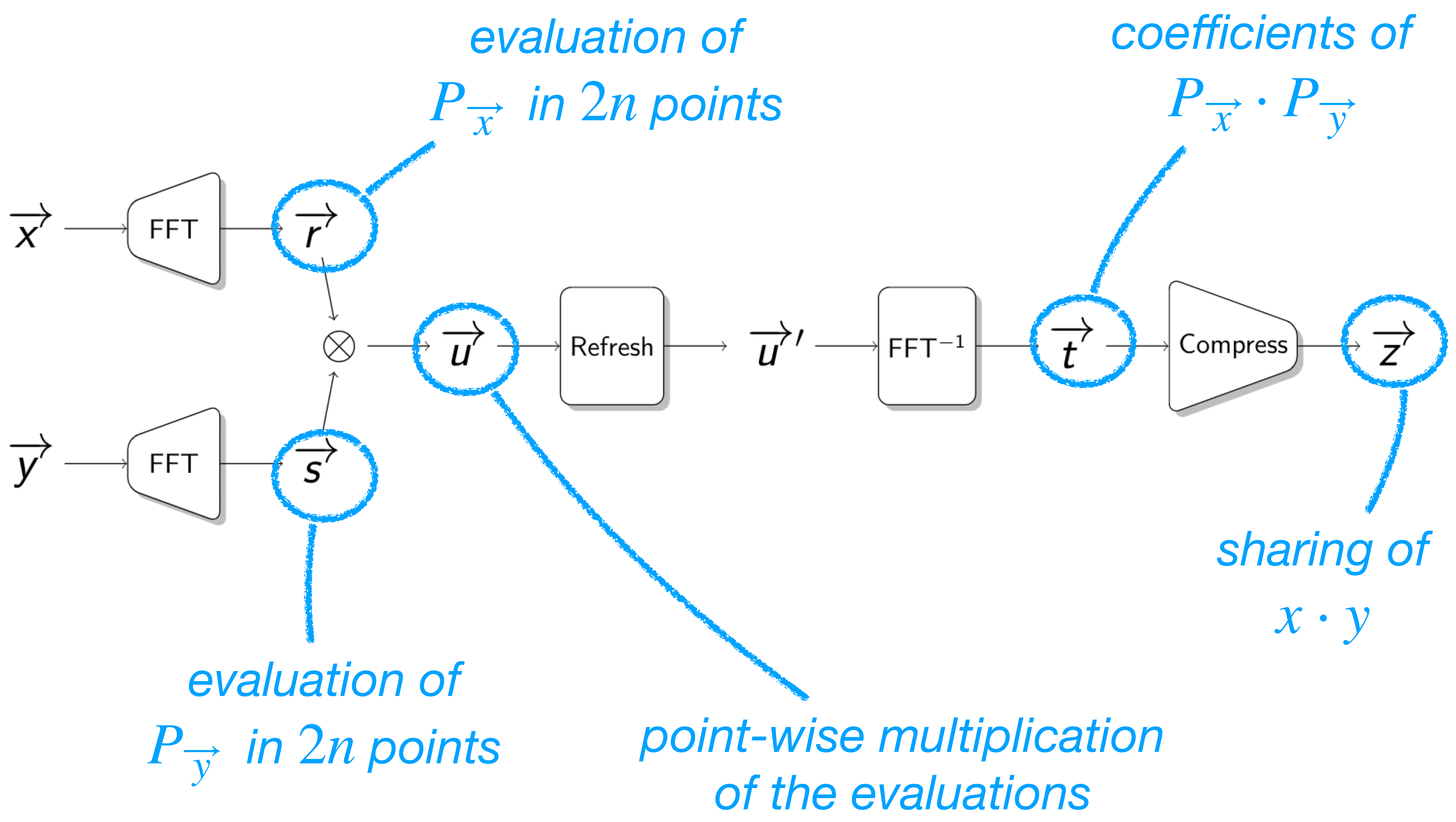
Multiplication gadget



Multiplication gadget



Multiplication gadget



Security

- We have sharewise addition / subtraction / copy gadgets

  $\Rightarrow$ inherently probing secure

- Multiplication gadgets composed of
  - sharewise blocks
  - FFT blocks
  - refresh blocks

Security

- We have sharewise addition / subtraction / copy gadgets

  $\Rightarrow$ inherently probing secure

- Multiplication gadgets composed of
  - sharewise blocks
  - FFT blocks
  - refresh blocks
- We can apply the IOS composition approach

Security

- We have sharewise addition / subtraction / copy gadgets

  $\Rightarrow$ inherently probing secure

- Multiplication gadgets composed of
  - sharewise blocks
  - FFT blocks
  - refresh blocks
- We can apply the IOS composition approach

  ⚠️ assuming the FFT blocks are probing-secure

Security

- We have sharewise addition / subtraction / copy gadgets

  $\Rightarrow$ inherently probing secure

- Multiplication gadgets composed of
  - sharewise blocks
  - FFT blocks
  - refresh blocks
- We can apply the IOS composition approach

⚠️ assuming the FFT blocks are probing-secure

Security reduction: PS FFT $\Rightarrow$ region PS scheme

Statistical security (GJR)

- Pick a random ω over $\mathbb{K}$

Statistical security (GJR)

- Pick a random ω over $\mathbb{K}$
- Use a "linear" FFT
  - e.g. NTT, Cantor / Gao-Mateer additive FFT

Statistical security (GJR)

- Pick a random ω over $\mathbb{K}$
- Use a “linear” FFT
  - e.g. NTT, Cantor / Gao-Mateer additive FFT
- Any $n-1$ probes can be perfectly simulated

with proba $1-\dfrac{n}{|\mathbb{K}|}$ (over the random choice of ω)

Statistical security (GJR)

- Pick a random ω over $\mathbb{K}$
- Use a “linear” FFT
  - e.g. NTT, Cantor / Gao-Mateer additive FFT
- Any $n-1$ probes can be perfectly simulated

with proba $1 - \frac{n}{|\mathbb{K}|}$ (over the random choice of ω)

should be negligible

Statistical security (GJR)

- Pick a random ω over $\mathbb{K}$
- Use a "linear" FFT
  - e.g. NTT, Cantor / Gao-Mateer additive FFT
- Any $n-1$ probes can be perfectly simulated

  with proba $1 - \frac{n}{|\mathbb{K}|}$ (over the random choice of ω)

  should be negligible
- Constraint: $|\mathbb{K}| \approx n2^\lambda$ for λ-bit security

  $\Rightarrow (\lambda + \log n)$-bit field elements

Statistical security (GJR)

- Pick a random ω over $\mathbb{K}$
- Use a "linear" FFT
  - e.g. NTT, Cantor / Gao-Mateer additive FFT
- Any $n-1$ probes can be perfectly simulated

  with proba $1-\frac{n}{|\mathbb{K}|}$ (over the random choice of ω)

  should be negligible
- Constraint: $|\mathbb{K}| \approx n2^{\lambda}$ for λ-bit security

  $\Rightarrow (\lambda + \log n)$-bit field elements
- Open problem: probing secure FFT on smaller fields

Application to AES and MiMC

- We apply
  - GJR+ (our variant with IOS composition)

    $\Rightarrow$ $O(n \log n)$ complexity / $O(1/\log n)$ leakage rate
  - ISW+ (ISW mult. & BPCZ refresh)

    $\Rightarrow$ $O(n^2)$ complexity / $O(1/n)$ leakage rate
- To
  - AES: $\mathbb{K} = \mathbb{F}_{256}$ $\Rightarrow$ Gao-Mateer additive FFT
  - MiMC: $\mathbb{K} = \mathbb{F}_p$ $\Rightarrow$ Number Theoretic Transform (NTT)

Application to AES and MiMC

— Results for AES —

n		Mul	Add.	Random
8	Full AES with ISW^+	64896	297088	123520
	Full AES with GJR^+	157056	257408	110080
	Efficiency ratio (GJR^+/ISW^+)	2.43	0.87	0.9
16	Full AES with ISW^+	211712	926976	372480
	Full AES with GJR^+	396032	683776	286720
	Efficiency ratio (GJR^+/ISW^+)	1.88	0.74	0.77
32	Full AES with ISW^+	751104	2847232	1077760
	Full AES with GJR^+	955904	1725952	706560
	Efficiency ratio (GJR^+/ISW^+)	1.28	0.61	0.66
64	Full AES with ISW^+	2812928	8991744	3148800
	Full AES with GJR^+	2239488	4209664	1679360
	Efficiency ratio (GJR^+/ISW^+)	0.8	0.47	0.54
128	Full AES with ISW^+	10868736	29820928	9594880
	Full AES with GJR^+	5134336	10016768	3891200
	Efficiency ratio (GJR^+/ISW^+)	0.48	0.34	0.41

Application to AES and MiMC

— Results for AES —

n		Mul	Add.	Random
8	Full AES with ISW^+	64896	297088	123520
	Full AES with GJR^+	157056	257408	110080
	Efficiency ratio (GJR^+/ISW^+)	2.43	0.87	0.9
16	Full AES with ISW^+	211712	926976	372480
	Full AES with GJR^+	396032	683776	286720
	Efficiency ratio (GJR^+/ISW^+)	1.88	0.74	0.77
32	Full AES with ISW^+	751104	2847232	1077760
	Full AES with GJR^+	955904	1725952	706560
	Efficiency ratio (GJR^+/ISW^+)	1.28	0.61	0.66
64	Full AES with ISW^+	2812928	8991744	3148800
	Full AES with GJR^+	2239488	4209664	1679360
	Efficiency ratio (GJR^+/ISW^+)	0.8	0.47	0.54
128	Full AES with ISW^+	10868736	29820928	9594880
	Full AES with GJR^+	5134336	10016768	3891200
	Efficiency ratio (GJR^+/ISW^+)	0.48	0.34	0.41

⚠️ The field should be large for GJR+

Application to AES and MiMC

— Results for MiMC —

n		Mul	Add.	Random
8	Full MiMC with ISW$^+$	10416.0	45408.0	17544.0
	Full MiMC with GJR$^+$	40512.0	66128.0	20100.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	3.89	1.46	1.15
16	Full MiMC with ISW$^+$	41600.0	153056.0	55856.0
	Full MiMC with GJR$^+$	100796.0	165968.0	51872.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	2.43	1.09	0.93
32	Full MiMC with ISW$^+$	166208.0	513536.0	173984.0
	Full MiMC with GJR$^+$	240812.0	399360.0	127088.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	1.45	0.78	0.74
64	Full MiMC with ISW$^+$	664320.0	1773696.0	555456.0
	Full MiMC with GJR$^+$	559740.0	933568.0	300864.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	0.85	0.53	0.55
128	Full MiMC with ISW$^+$	2656000.0	6367744.0	1857664.0
	Full MiMC with GJR$^+$	1275388.0	2136832.0	695104.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	0.49	0.34	0.38

Thank you for watching!

🙏

For any questions:

matthieu.rivain@cryptoexperts.com