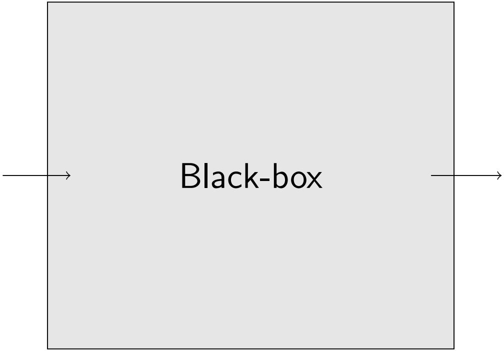
Combined Fault and Leakage Resilience: Composability, Constructions and Compiler

Sebastian Berndt Thomas Eisenbarth Sebastian Faust Marc Gourjon
Maximilian Orlt Okan Seker

Fault and Side-Channel Attacks

Fault and Side-Channel Attacks

Classical setting: Black-box model

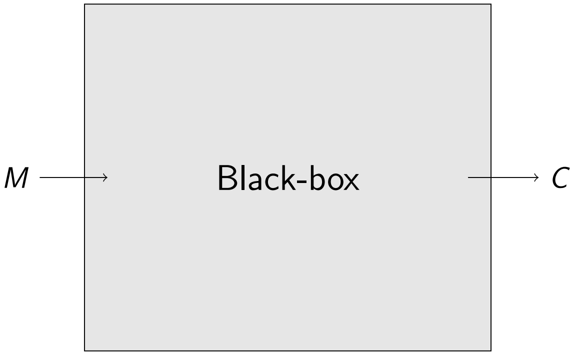


Fault and Side-Channel Attacks

Classical setting: Black-box model

- Adversary learns Input/Output

  E.g. plain-text/cipher-text (M, C)



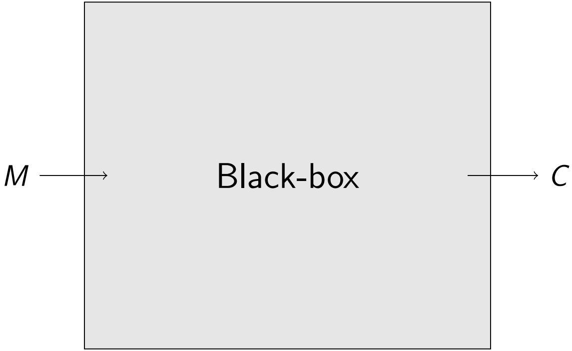
Fault and Side-Channel Attacks

Classical setting: Black-box model

- Adversary learns Input/Output

  E.g. plain-text/cipher-text
  (M, C)

Real Adversary is more powerful



Fault and Side-Channel Attacks

Classical setting: Black-box model

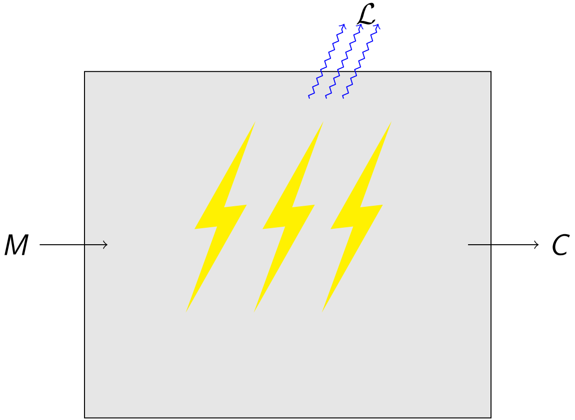
- Adversary learns Input/Output

  E.g. plain-text/cipher-text (M, C)

Real Adversary is more powerful

- Side-Channel Attack

  E.g. Power consumption



Fault and Side-Channel Attacks

Classical setting: Black-box model

- Adversary learns Input/Output

  E.g. plain-text/cipher-text (M, C)

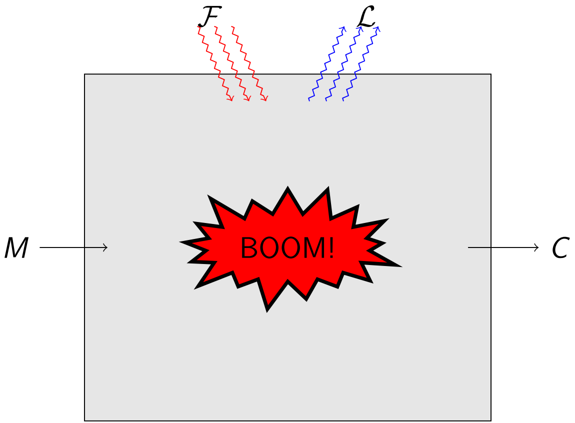
Real Adversary is more powerful

- Side-Channel Attack

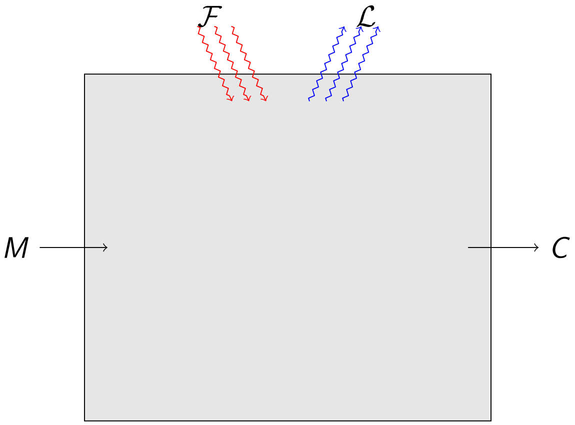
  E.g. Power consumption

- Fault Attacks

  E.g. Electromagnetic Pulses

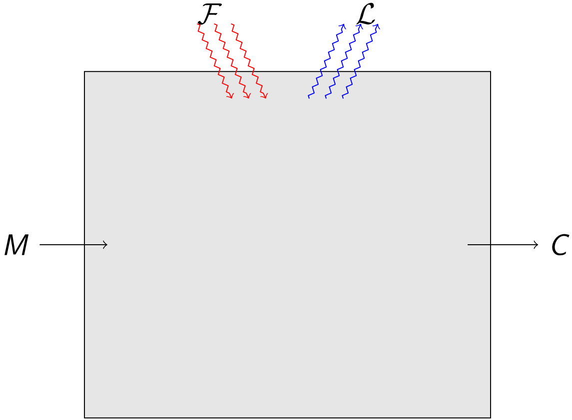


Security Model



Security Model

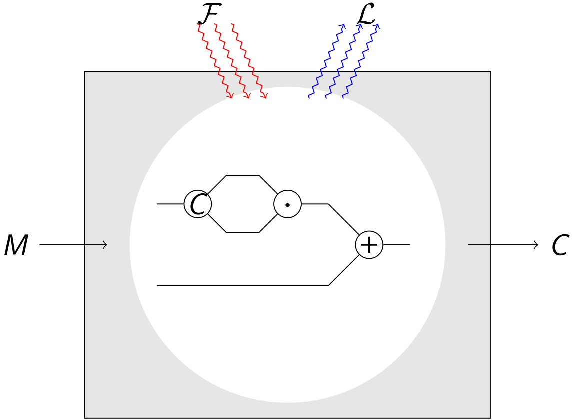
Computational model



Security Model

Computational model

- Arithmetic circuit

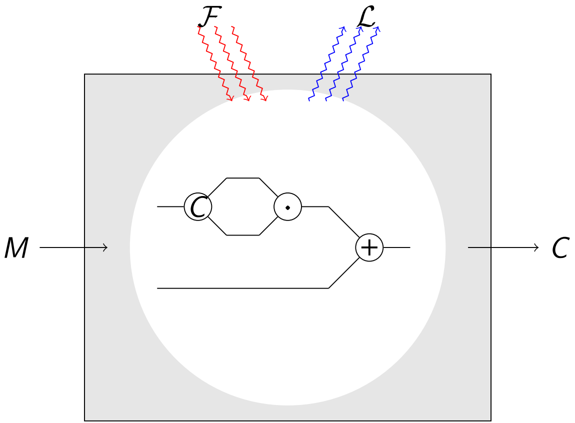


Security Model

Computational model

- Arithmetic circuit

Adversarial model



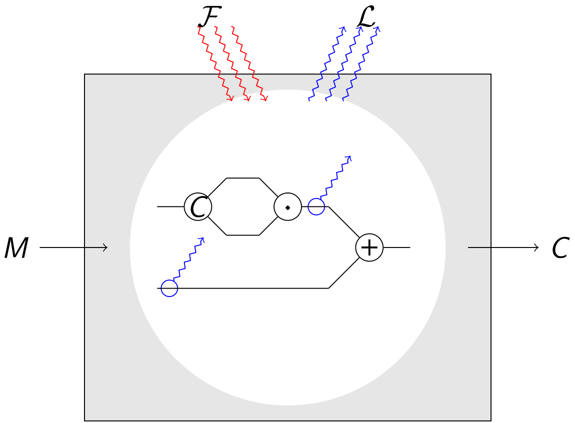
Security Model

Computational model

- Arithmetic circuit

Adversarial model

- Leakage model
  d arbitrary wires can be probed



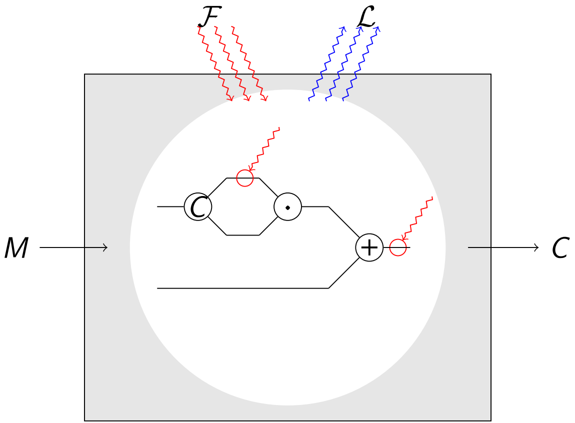
Security Model

Computational model

- Arithmetic circuit

Adversarial model

- Leakage model
  d arbitrary wires can be probed
- Fault model
  e arbitrary wires can be faulted



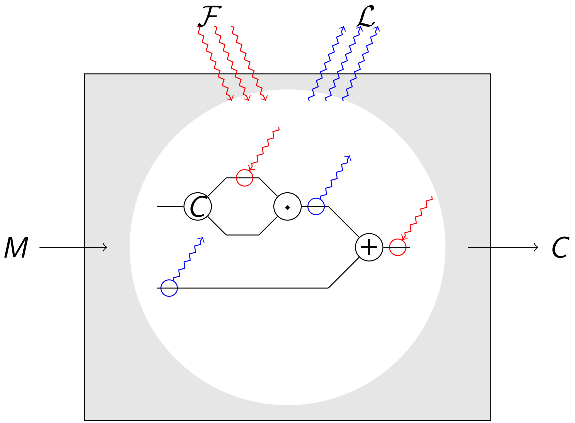
Security Model

Computational model

- Arithmetic circuit

Adversarial model

- Leakage model
  d arbitrary wires can be probed
- Fault model
  e arbitrary wires can be faulted
- Combined model



State of the Art: Encodings

State of the Art: Encodings

Duplicated Encoding

State of the Art: Encodings

Duplicated Encoding

- $d+1$ out of $d+1$ secret sharing (Masking)
  d probes do not reveal the secret

State of the Art: Encodings

Duplicated Encoding

- $d+1$ out of $d+1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e+1$ times (Duplication)
  e faults can be detected

State of the Art: Encodings

Duplicated Encoding

- $d+1$ out of $d+1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e+1$ times (Duplication)
  e faults can be detected

$\Rightarrow O(d \cdot e)$ shares

State of the Art: Encodings

Duplicated Encoding

- $d + 1$ out of $d + 1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e + 1$ times (Duplication)
  e faults can be detected

$\Rightarrow O(d \cdot e)$ shares

Shamir Secret Sharing

State of the Art: Encodings

Duplicated Encoding

- $d + 1$ out of $d + 1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e + 1$ times (Duplication)
  e faults can be detected

$\Rightarrow O(d \cdot e)$ shares

Shamir Secret Sharing

- $d + 1$ out of $d + e + 1$ secret sharing

State of the Art: Encodings

Duplicated Encoding

- $d + 1$ out of $d + 1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e + 1$ times (Duplication)
  e faults can be detected

$\Rightarrow O(d \cdot e)$ shares

Shamir Secret Sharing

- $d + 1$ out of $d + e + 1$ secret sharing
  d probes do not reveal the secret

State of the Art: Encodings

Duplicated Encoding

- $d+1$ out of $d+1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e+1$ times (Duplication)
  e faults can be detected

$\Rightarrow O(d \cdot e)$ shares

Shamir Secret Sharing

- $d+1$ out of $d+e+1$ secret sharing
  d probes do not reveal the secret
  e faults can be detected

State of the Art: Encodings

Duplicated Encoding

- $d+1$ out of $d+1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e+1$ times (Duplication)
  e faults can be detected

$\Rightarrow O(d \cdot e)$ shares

Shamir Secret Sharing

- $d+1$ out of $d+e+1$ secret sharing
  d probes do not reveal the secret
  e faults can be detected

$\Rightarrow O(d+e)$ shares

State of the Art: Encodings

Duplicated Encoding

- $d+1$ out of $d+1$ secret sharing (Masking)
  d probes do not reveal the secret
- Copy everything $e+1$ times (Duplication)
  e faults can be detected

$\Rightarrow O(d \cdot e)$ shares

Shamir Secret Sharing

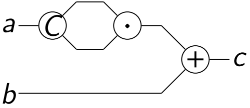
- $d+1$ out of $d+e+1$ secret sharing
  d probes do not reveal the secret
  e faults can be detected

$\Rightarrow O(d+e)$ shares

$O(d+e) \leq O(d \cdot e) \Rightarrow$ This work uses Shamir Secret Sharing

Countermeasure: Compute on Encodings

Countermeasure: Compute on Encodings



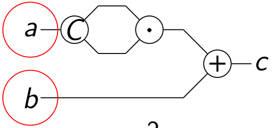
$$c = a^2 + b$$

Countermeasure: Compute on Encodings

- Encode the inputs a and b

a_0

a_1



$$c = a^2 + b$$

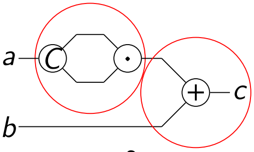
$\xrightarrow{\text{Compiler}}$

b_0

b_1

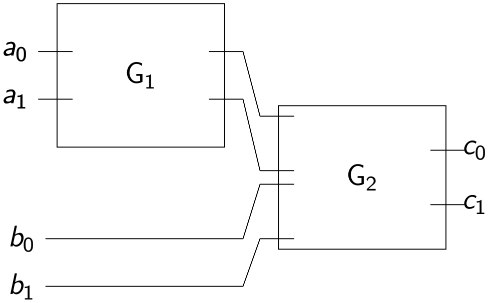
Countermeasure: Compute on Encodings

- Encode the inputs a and b
- Only compute on encodings



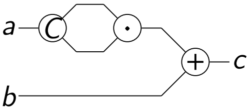
$$c = a^2 + b$$

$$\xrightarrow{\text{Compiler}}$$



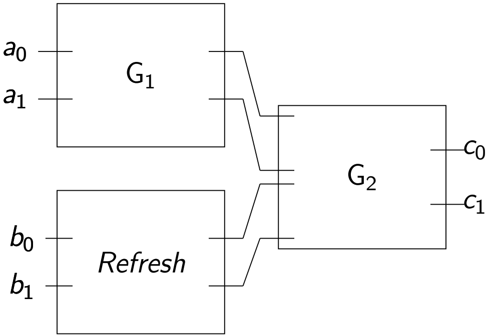
Countermeasure: Compute on Encodings

- Encode the inputs a and b
- Only compute on encodings
- Randomize the circuit



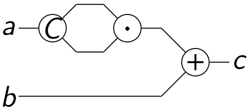
$$c = a^2 + b$$

$$\xrightarrow{\text{Compiler}}$$



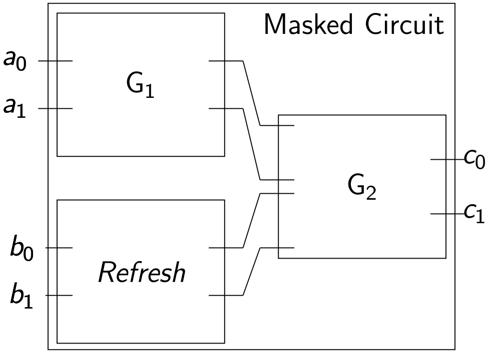
Countermeasure: Compute on Encodings

- Encode the inputs a and b
- Only compute on encodings
- Randomize the circuit



$c = a^2 + b$

$\xrightarrow{\text{Compiler}}$



Contribution vs State of the Art

Contribution vs State of the Art

Compiler secure against d probes and e faults

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18] [DN19]

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18] [DN19]

- $n = 2d + e + 1$ shares

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18] [DN19]

- $n = 2d + e + 1$ shares
- $O(n^2)$ Complexity

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18]

- $n = 2d + e + 1$ shares
- $O(n^2)$ Complexity

[DN19]

- $n = d + e + 1$ shares

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18]

- $n = 2d + e + 1$ shares
- $O(n^2)$ Complexity

[DN19]

- $n = d + e + 1$ shares
- $> O(n^3)$ Complexity

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18]

- $n = 2d + e + 1$ shares
- $O(n^2)$ Complexity

$$\xrightarrow{\text{Improve number of shares}}$$

[DN19]

- $n = d + e + 1$ shares
- $> O(n^3)$ Complexity

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18]

- $n = 2d + e + 1$ shares
- $O(n^2)$ Complexity

[DN19]

- $n = d + e + 1$ shares
- $> O(n^3)$ Complexity

$\xrightarrow{\text{Improve number of shares}}$

$\xleftarrow{\text{Improve complexity}}$

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18]	This Work	[DN19]
▶ $n = 2d + e + 1$ shares	▶ $n = d + e + 1$ shares	▶ $n = d + e + 1$ shares
	Improve number of shares $\longrightarrow$	
▶ $O(n^2)$ Complexity	▶ $O(n^2)$ Complexity	▶ $> O(n^3)$ Complexity
	$\longleftarrow$ Improve complexity	

Contribution vs State of the Art

Compiler secure against d probes and e faults

[SFRES18]	This Work	[DN19]
$n = 2d + e + 1$ shares	$n = d + e + 1$ shares	$n = d + e + 1$ shares
	$\xrightarrow{\text{Improve number of shares}}$	
$O(n^2)$ Complexity	$O(n^2)$ Complexity	$> O(n^3)$ Complexity
	$\xleftarrow{\text{Improve complexity}}$	

Improved Security: This work even allows $d/2$ probes in each gadget.

Paper Overview

Paper Overview

New Gadgets:
Mult., Add. & Ref.

Paper Overview

New Gadgets:
Mult., Add. & Ref.

Leakage Resilience:
d probes in the circuit
$d/2$ probes in each gadget

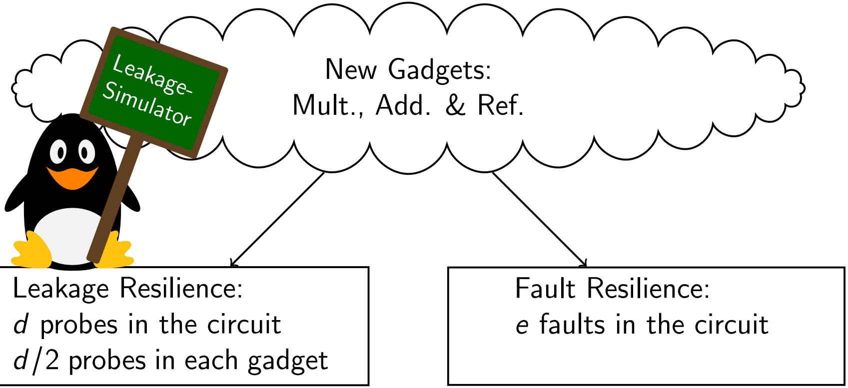
Paper Overview

New Gadgets:
Mult., Add. & Ref.

Leakage Resilience:
d probes in the circuit
$d/2$ probes in each gadget

Fault Resilience:
e faults in the circuit

Paper Overview



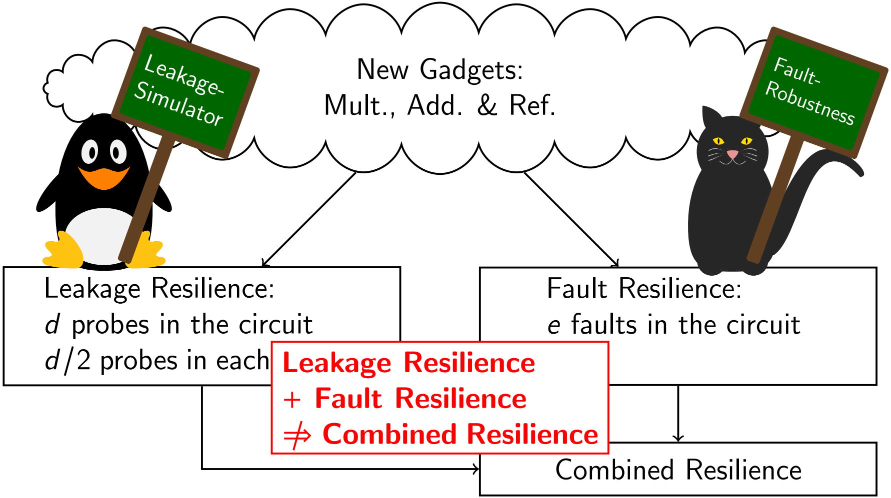
Paper Overview

New Gadgets:
Mult., Add. & Ref.

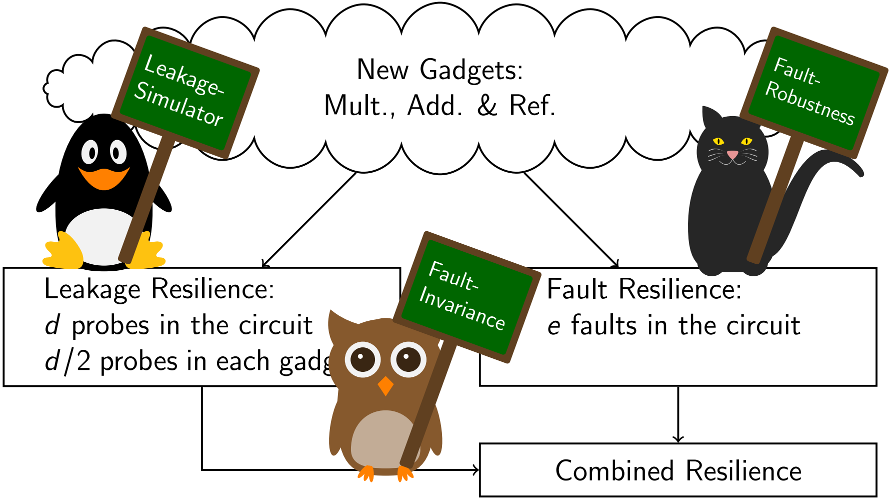
Leakage Resilience:
d probes in the circuit
$d/2$ probes in each gadget

Fault Resilience:
e faults in the circuit

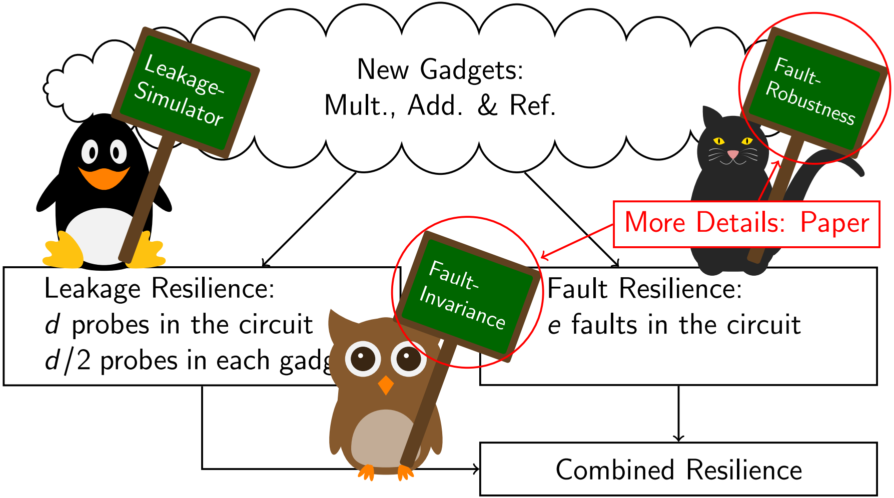
Paper Overview



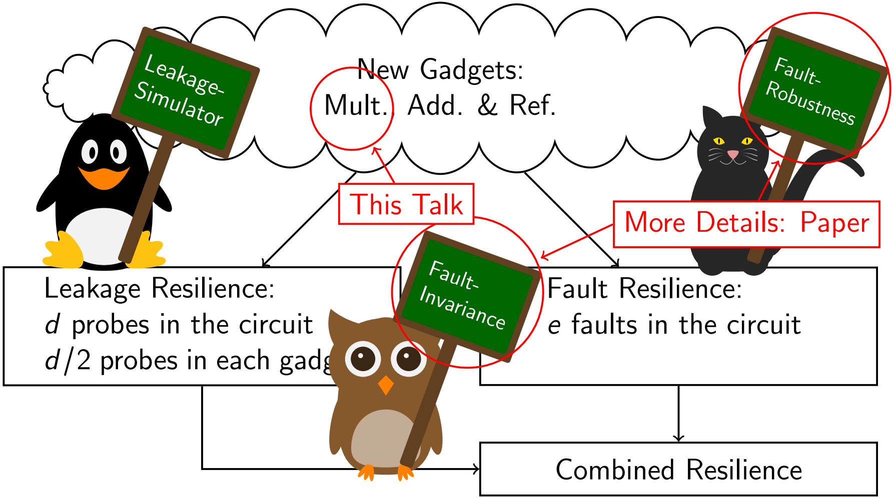
Paper Overview



Paper Overview



Paper Overview



Recap: Shamir Secret Sharing

Recap: Shamir Secret Sharing

- Take n different α_i with $\alpha_i \neq 0$

Recap: Shamir Secret Sharing

- Take n different α_i with $\alpha_i \neq 0$

$\underline{\mathsf{Enc}(s)}$

$r_1 \ldots r_d \leftarrow\!\$\ \mathbb{F}$

for $j = 0, \ldots, n-1$ **:**

$$s_j \leftarrow \sum_{i=1}^{d} r_i \alpha_j^i + s$$

return $[\![s]\!]_d^n := (s_0, \ldots, s_{n-1})$

Recap: Shamir Secret Sharing

- Take n different α_i with $\alpha_i \neq 0$
- $[\![s]\!]_d^n = (s_i)_{i \in [n]} \leftarrow \mathsf{Enc}(s)$

$\mathsf{Enc}(s)$

$r_1 \dots r_d \leftarrow\$ \mathbb{F}$

for $j = 0, \dots, n-1$ **:**

$$s_j \leftarrow \sum_{i=1}^{d} r_i \alpha_j^i + s$$

return $[\![s]\!]_d^n := (s_0, \dots, s_{n-1})$

Recap: Shamir Secret Sharing

- Take n different α_i with $\alpha_i \neq 0$
- $[\![s]\!]_d^n = (s_i)_{i \in [n]} \leftarrow \mathsf{Enc}(s)$
  $s_i = f(\alpha_i)$ with $f(x) = \sum_{i=1}^{d} r_i x^i + s$

$\mathsf{Enc}(s)$

$r_1 \ldots r_d \leftarrow\$ \mathbb{F}$

for $j = 0, \ldots, n-1$ **:**

$$s_j \leftarrow \sum_{i=1}^{d} r_i \alpha_j^i + s$$

return $[\![s]\!]_d^n := (s_0, \ldots, s_{n-1})$

Recap: Shamir Secret Sharing

- Take n different α_i with $\alpha_i \neq 0$
- $[\![s]\!]_d^n = (s_i)_{i \in [n]} \leftarrow \mathsf{Enc}(s)$
  $s_i = f(\alpha_i)$ with $f(x) = \sum_{i=1}^{d} r_i x^i + s$
- $s = f(0) \leftarrow \mathsf{Dec}([\![s]\!]_d^n)$

$\mathsf{Enc}(s)$

$r_1 \dots r_d \leftarrow\!\$\ \mathbb{F}$

for $j = 0, \dots, n-1$ **:**

$$s_j \leftarrow \sum_{i=1}^{d} r_i \alpha_j^i + s$$

return $[\![s]\!]_d^n := (s_0, \dots, s_{n-1})$

Recap: Shamir Secret Sharing

- Take n different α_i with $\alpha_i \neq 0$
- $[\![s]\!]_d^n = (s_i)_{i\in[n]} \leftarrow \mathsf{Enc}(s)$
  $s_i = f(\alpha_i)$ with $f(x) = \sum_{i=1}^{d} r_i x^i + s$
- $s = f(0) \leftarrow \mathsf{Dec}([\![s]\!]_d^n)$

$\mathsf{Enc}(s)$

$r_1 \dots r_d \leftarrow\$ \mathbb{F}$

for $j = 0, \dots, n-1$:

$$s_j \leftarrow \sum_{i=1}^{d} r_i \alpha_j^i + s$$

return $[\![s]\!]_d^n := (s_0, \dots, s_{n-1})$

- Leakage Resilience: Any set of d shares s_i is uniformly, independently distributed.

Recap: Shamir Secret Sharing

- Take n different α_i with $\alpha_i \neq 0$
- $[\![s]\!]_d^n = (s_i)_{i\in[n]} \leftarrow \mathrm{Enc}(s)$
  $s_i = f(\alpha_i)$ with $f(x) = \sum_{i=1}^{d} r_i x^i + s$
- $s = f(0) \leftarrow \mathrm{Dec}([\![s]\!]_d^n)$

$\underline{\mathrm{Enc}(s)}$

$r_1 \dots r_d \leftarrow\$ \mathbb{F}$

for $j = 0, \dots, n-1$:

$$s_j \leftarrow \sum_{i=1}^{d} r_i \alpha_j^i + s$$

return $[\![s]\!]_d^n := (s_0, \dots, s_{n-1})$

- Leakage Resilience: Any set of d shares s_i is uniformly, independently distributed.
- Fault Resilience: If only e shares of $d + e + 1$ shares are faulted the polynomial has a degree $> d$!

Fixed SotA Compiler for Affine Circuits

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$

$$\left.\vphantom{\begin{cases}a\\b\end{cases}}\right\} \quad \begin{cases} (\sum_{i=1}^{d} r_i x^i + a) + (\sum_{i=1}^{d} r_i x^i + b) \\ = \sum_{i=1}^{d} (r_i + r_i^{\prime}) x^i + (a + b) \end{cases}$$

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$
- Result $[\![c]\!]_d^n = [\![a + b]\!]_d^n$

$$\Bigg\} \quad \Bigg\{ \begin{array}{c} (\sum_{i=1}^{d} r_i x^i + a) + (\sum_{i=1}^{d} r_i x^i + b) \\ = \sum_{i=1}^{d} (r_i + r_i^{\prime}) x^i + (a + b) \end{array}$$

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$
- Result $[\![c]\!]_d^n = [\![a+b]\!]_d^n$

$$\left.\vphantom{\begin{array}{c} X \\ X \\ X \end{array}}\right\} \quad \left\{ \begin{array}{c} (\sum_{i=1}^{d} r_i x^i + a) + (\sum_{i=1}^{d} r_i x^i + b) \\ = \sum_{i=1}^{d} (r_i + r_i') x^i + (a+b) \end{array} \right.$$

Refresh of $[\![a]\!]_d^n$

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$
- Result $[\![c]\!]_d^n = [\![a+b]\!]_d^n$

$$\left.\begin{array}{c} \\ \\ \end{array}\right\} \quad \left\{ \begin{array}{c} (\sum_{i=1}^{d} r_i x^i + a) + (\sum_{i=1}^{d} r_i x^i + b) \\ = \sum_{i=1}^{d} (r_i + r_i^{\prime}) x^i + (a+b) \end{array}\right.$$

Refresh of $[\![a]\!]_d^n$

- Generate $[\![b]\!]_d^n \leftarrow \mathrm{Enc}(0)$

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$
- Result $[\![c]\!]_d^n = [\![a+b]\!]_d^n$

$$\Bigg\}\quad\Bigg\{\begin{array}{c}(\sum_{i=1}^{d} r_i x^i + a) + (\sum_{i=1}^{d} r_i x^i + b)\\ = \sum_{i=1}^{d}(r_i + r_i^{\prime})x^i + (a+b)\end{array}$$

Refresh of $[\![a]\!]_d^n$

- Generate $[\![b]\!]_d^n \leftarrow \mathrm{Enc}(0)$
- Compute $[\![c]\!]_d^n = [\![a+b]\!]_d^n$

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$
- Result $[\![c]\!]_d^n = [\![a+b]\!]_d^n$

$$\left.\begin{array}{l} \\ \\ \end{array}\right\} \left\{ \begin{array}{c} (\sum_{i=1}^d r_i x^i + a) + (\sum_{i=1}^d r_i x^i + b) \\ = \sum_{i=1}^d (r_i + r_i^{\prime}) x^i + (a+b) \end{array}\right.$$

Refresh of $[\![a]\!]_d^n$

- Generate $[\![b]\!]_d^n \leftarrow \mathrm{Enc}(0)$
- Compute $[\![c]\!]_d^n = [\![a+b]\!]_d^n$

Probing Attack: Our Paper

Fixed SotA Compiler for Affine Circuits

Addition of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i + b_i$
- Result $[\![c]\!]_d^n = [\![a + b]\!]_d^n$

$$\left.\right\} \left\{ \begin{array}{c} (\sum_{i=1}^{d} r_i x^i + a) + (\sum_{i=1}^{d} r_i x^i + b) \\ = \sum_{i=1}^{d} (r_i + r_i^{!}) x^i + (a + b) \end{array} \right.$$

Refresh of $[\![a]\!]_d^n$

- Generate $[\![b]\!]_d^n \leftarrow \mathsf{Enc}(0)$
- Compute $[\![c]\!]_d^n = [\![a + b]\!]_d^n$

Probing Attack: Our Paper
Fix: d Refreshes in a row

Fixed SotA Compiler with $n = 2d + e + 1$

Fixed SotA Compiler with $n = 2d + e + 1$

Multiplication of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

Fixed SotA Compiler with $n = 2d + e + 1$

Multiplication of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i \cdot b_i$

Fixed SotA Compiler with $n = 2d + e + 1$

Multiplication of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i \cdot b_i$
- Result $[\![c]\!]_{2d}^n = [\![a \cdot b]\!]_{2d}^n$

Fixed SotA Compiler with $n = 2d + e + 1$

Multiplication of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i \cdot b_i$
- Result $[\![c]\!]_{2d}^n = [\![a \cdot b]\!]_{2d}^n$

$$\left.\vphantom{\begin{array}{c} \\ \\ \\ \\ \end{array}}\right\} \quad \left\{ \begin{array}{c} (\sum_{i=1}^{d} r_i x^i + a) \cdot (\sum_{i=1}^{d} r_i x^i + b) \\ = \sum_{i=1}^{2d} (r_i'') x^i + (a \cdot b) \end{array} \right.$$

Fixed SotA Compiler with $n = 2d + e + 1$

Multiplication of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i \cdot b_i$
- Result $[\![c]\!]_{2d}^n = [\![a \cdot b]\!]_{2d}^n$
- Reduce $[\![c]\!]_{2d}^n$ down to $[\![c]\!]_d^n$

$$\left.\vphantom{\begin{array}{c}1\\1\\1\\1\end{array}}\right\} \left\{ \begin{array}{c} (\sum_{i=1}^{d} r_i x^i + a) \cdot (\sum_{i=1}^{d} r_i x^i + b) \\ = \sum_{i=1}^{2d} (r_i'') x^i + (a \cdot b) \end{array} \right.$$

Fixed SotA Compiler with $n = 2d + e + 1$

Multiplication of $[\![a]\!]_d^n$ and $[\![b]\!]_d^n$

- Compute $c_i \leftarrow a_i \cdot b_i$
- Result $[\![c]\!]_{2d}^n = [\![a \cdot b]\!]_{2d}^n$
- Reduce $[\![c]\!]_{2d}^n$ down to $[\![c]\!]_d^n$
- Problem: Requires $n = 2d + e + 1$

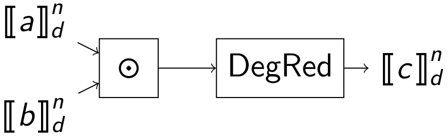
$$\left(\sum_{i=1}^{d} r_i x^i + a\right) \cdot \left(\sum_{i=1}^{d} r_i x^i + b\right) = \sum_{i=1}^{2d} (r_i'') x^i + (a \cdot b)$$

Compiler with $n = d + e + 1$

Compiler with $n = d + e + 1$

Instead of

- Compute $[\![c]\!]^n_{2d} = [\![a \cdot b]\!]^n_{2d}$
- Reduce $[\![c]\!]^n_{2d}$ down to $[\![c]\!]^n_d$



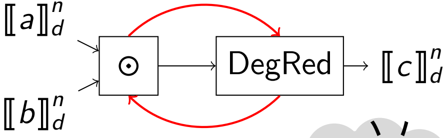
Compiler with $n = d + e + 1$

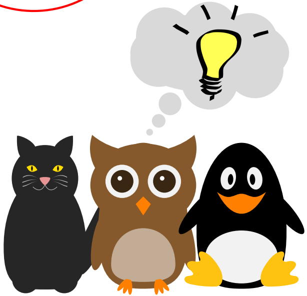
Instead of

- Compute $\llbracket c \rrbracket^n_{2d} = \llbracket a \cdot b \rrbracket^n_{2d}$
- Reduce $\llbracket c \rrbracket^n_{2d}$ down to $\llbracket c \rrbracket^n_{d}$

Reverse the steps

- Reduce $\llbracket a \rrbracket^n_{d}$, $\llbracket b \rrbracket^n_{d}$ down to $\llbracket a \rrbracket^n_{d/2}$, $\llbracket b \rrbracket^n_{d/2}$
- Compute $\llbracket c \rrbracket^n_{d} = \llbracket a \cdot b \rrbracket^n_{d}$





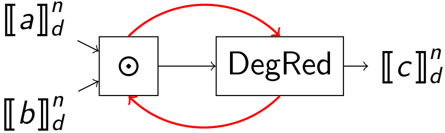
Compiler with $n = d + e + 1$

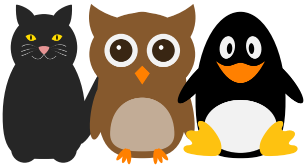
Instead of

- Compute $[\![c]\!]^n_{2d} = [\![a \cdot b]\!]^n_{2d}$
- Reduce $[\![c]\!]^n_{2d}$ down to $[\![c]\!]^n_d$

Reverse the steps

- Reduce $[\![a]\!]^n_d$, $[\![b]\!]^n_d$ down to $[\![a]\!]^n_{d/2}$, $[\![b]\!]^n_{d/2}$
- Compute $[\![c]\!]^n_d = [\![a \cdot b]\!]^n_d$
- Problem d probes in $[\![a]\!]^n_{d/2}$ reveal a





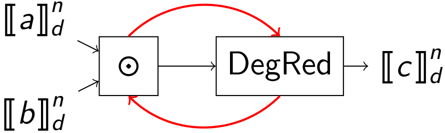
Compiler with $n = d + e + 1$

Instead of

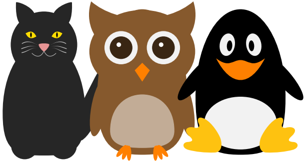
- Compute $[\![c]\!]^n_{2d} = [\![a \cdot b]\!]^n_{2d}$
- Reduce $[\![c]\!]^n_{2d}$ down to $[\![c]\!]^n_{d}$

Reverse the steps

- Reduce $[\![a]\!]^n_{d}$, $[\![b]\!]^n_{d}$ down to $[\![a]\!]^n_{d/2}$, $[\![b]\!]^n_{d/2}$
- Compute $[\![c]\!]^n_{d} = [\![a \cdot b]\!]^n_{d}$
- Problem d probes in $[\![a]\!]^n_{d/2}$ reveal a

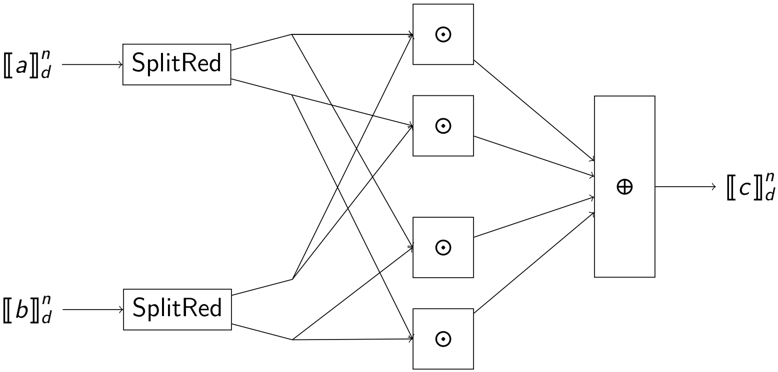


Solution: SplitRed splits $[\![a]\!]^n_{d}$ into $[\![a']\!]^n_{d}$ and $[\![a'']\!]^n_{d}$ with $[\![a']\!]^n_{d} + [\![a'']\!]^n_{d} = [\![a]\!]^n_{d/2}$

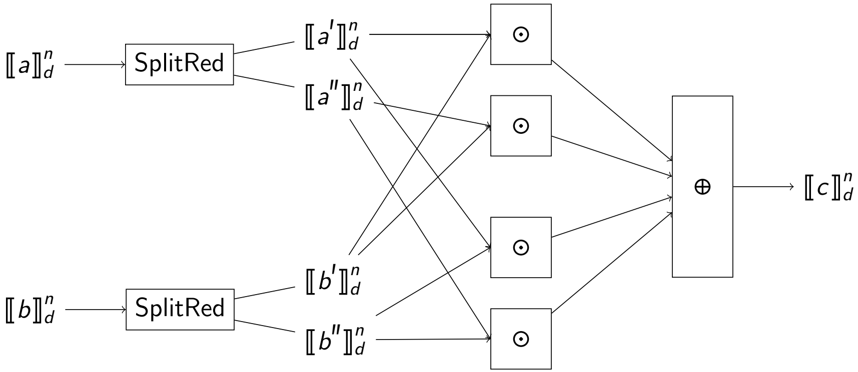


The Multiplication

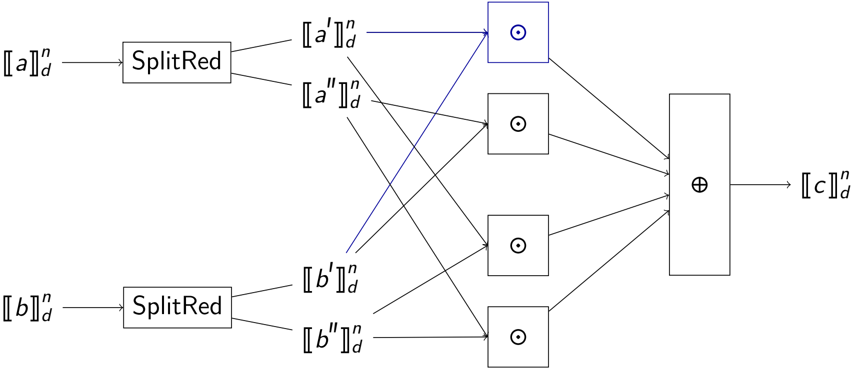
The Multiplication



The Multiplication

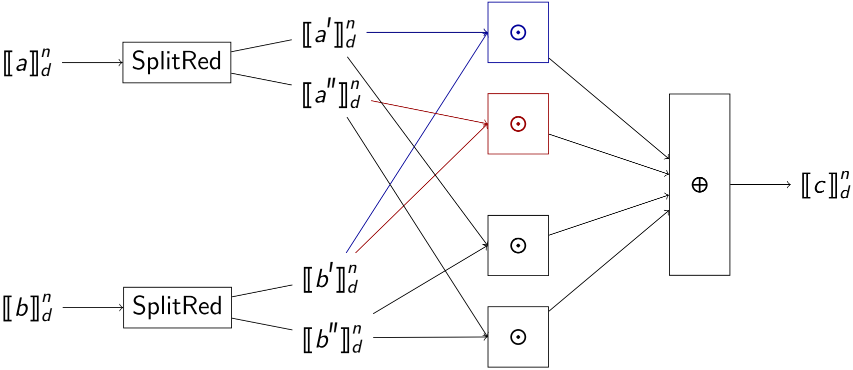


The Multiplication



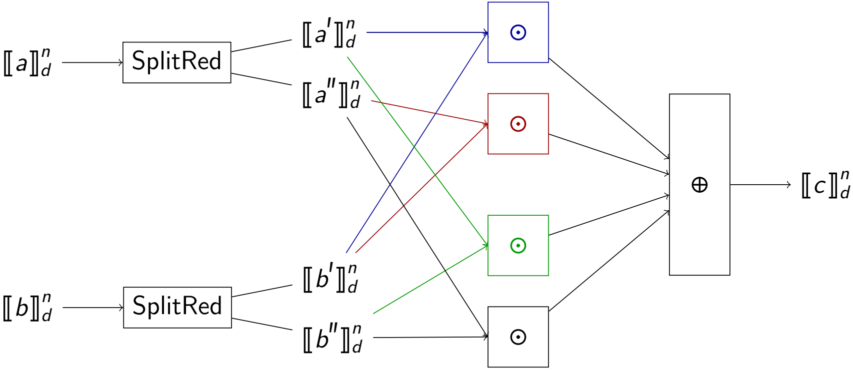
$$[\![a']\!]^n_d \cdot [\![b']\!]^n_d$$

The Multiplication



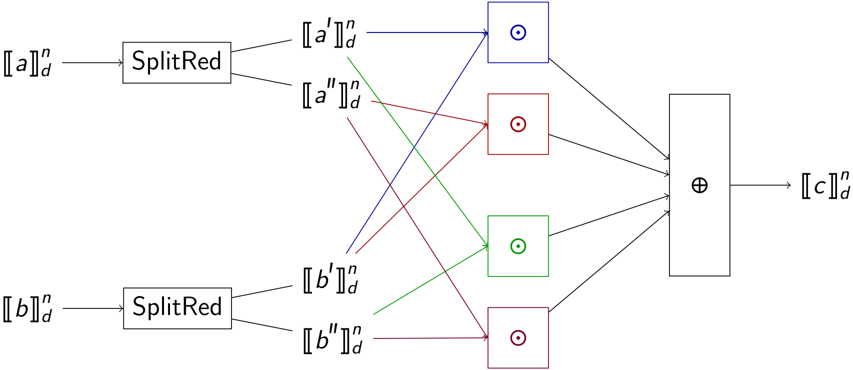
$[\![a']\!]_d^n \cdot [\![b']\!]_d^n \quad [\![a'']\!]_d^n \cdot [\![b']\!]_d^n$

The Multiplication



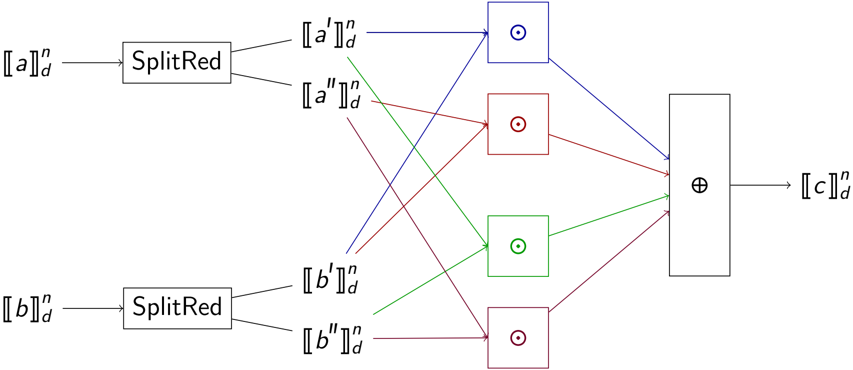
$[\![a']\!]_d^n \cdot [\![b']\!]_d^n \quad [\![a'']\!]_d^n \cdot [\![b']\!]_d^n \quad [\![a']\!]_d^n \cdot [\![b'']\!]_d^n$

The Multiplication



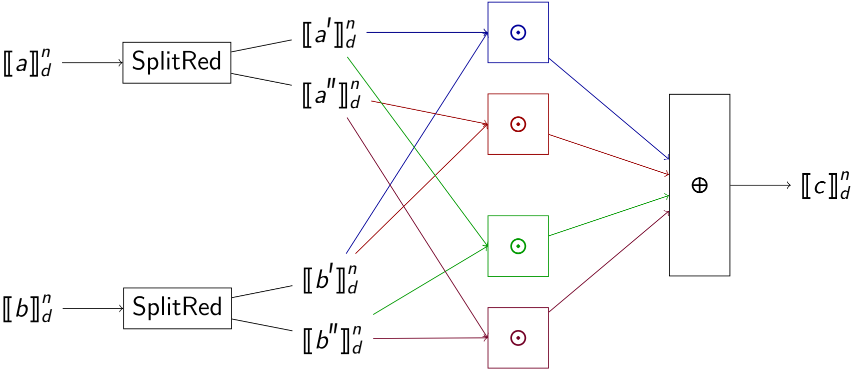
$[\![a']\!]_d^n \cdot [\![b']\!]_d^n \quad [\![a'']\!]_d^n \cdot [\![b']\!]_d^n \quad [\![a']\!]_d^n \cdot [\![b'']\!]_d^n \quad [\![a'']\!]_d^n \cdot [\![b'']\!]_d^n$

The Multiplication



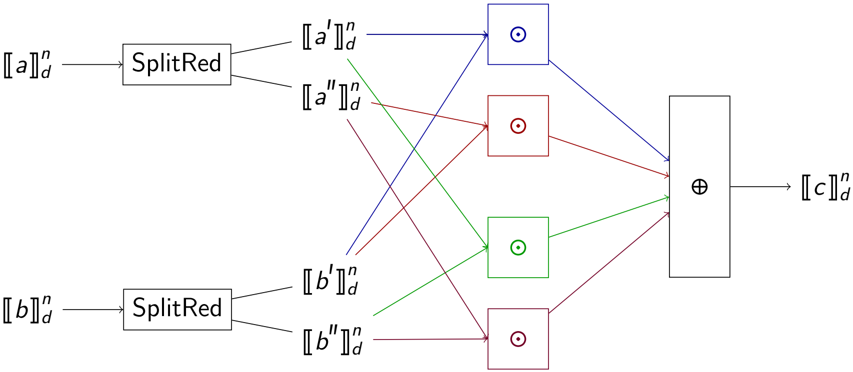
$$[\![c]\!]^n_d = \quad [\![a']\!]^n_d \cdot [\![b']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b']\!]^n_d + [\![a']\!]^n_d \cdot [\![b'']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b'']\!]^n_d$$

The Multiplication



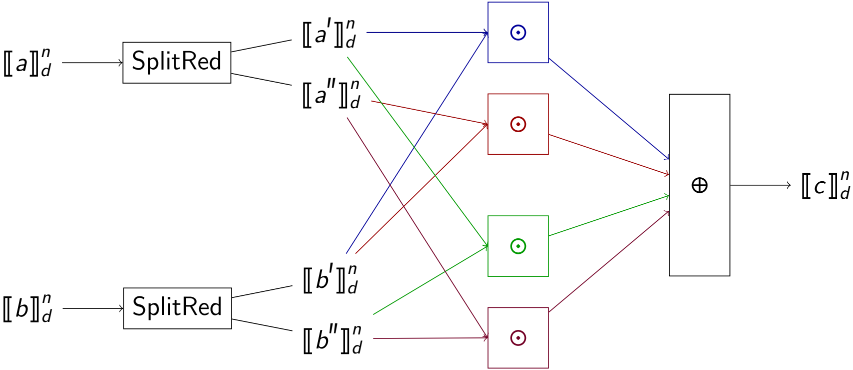
$$
\begin{aligned}
[\![c]\!]^n_d &= [\![a']\!]^n_d \cdot [\![b']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b']\!]^n_d + [\![a']\!]^n_d \cdot [\![b'']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b'']\!]^n_d \\
&= ([\![a']\!]^n_d + [\![a'']\!]^n_d) \cdot ([\![b']\!]^n_d + [\![b'']\!]^n_d)
\end{aligned}
$$

The Multiplication



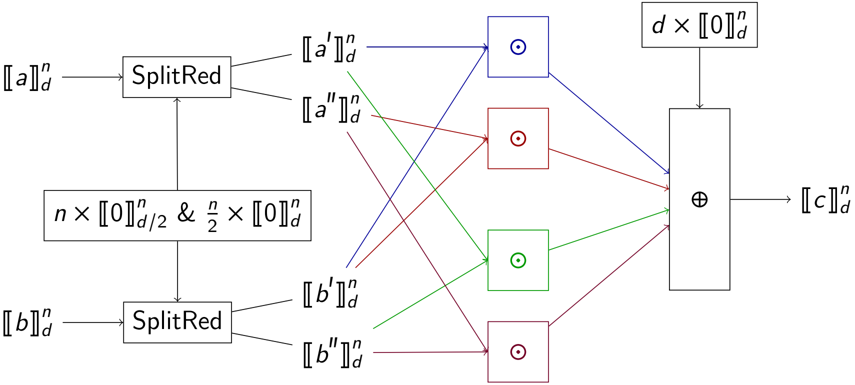
$$\begin{aligned}\llbracket c \rrbracket^n_d &= \llbracket a' \rrbracket^n_d \cdot \llbracket b' \rrbracket^n_d + \llbracket a'' \rrbracket^n_d \cdot \llbracket b' \rrbracket^n_d + \llbracket a' \rrbracket^n_d \cdot \llbracket b'' \rrbracket^n_d + \llbracket a'' \rrbracket^n_d \cdot \llbracket b'' \rrbracket^n_d \\ &= (\llbracket a' \rrbracket^n_d + \llbracket a'' \rrbracket^n_d) \cdot (\llbracket b' \rrbracket^n_d + \llbracket b'' \rrbracket^n_d) = \llbracket a \rrbracket^n_{d/2} \cdot \llbracket b \rrbracket^n_{d/2}\end{aligned}$$

The Multiplication



$$
\begin{aligned}
[\![c]\!]^n_d &= [\![a']\!]^n_d \cdot [\![b']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b']\!]^n_d + [\![a']\!]^n_d \cdot [\![b'']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b'']\!]^n_d \\
&= ([\![a']\!]^n_d + [\![a'']\!]^n_d) \cdot ([\![b']\!]^n_d + [\![b'']\!]^n_d) = [\![a]\!]^n_{d/2} \cdot [\![b]\!]^n_{d/2} = [\![a \cdot b]\!]^n_d
\end{aligned}
$$

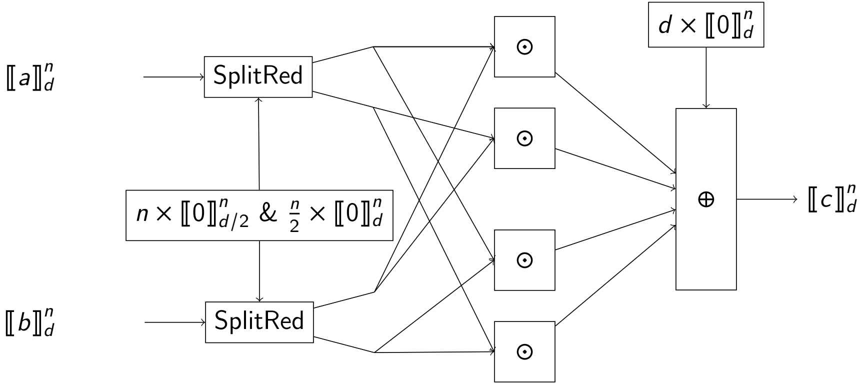
The Multiplication



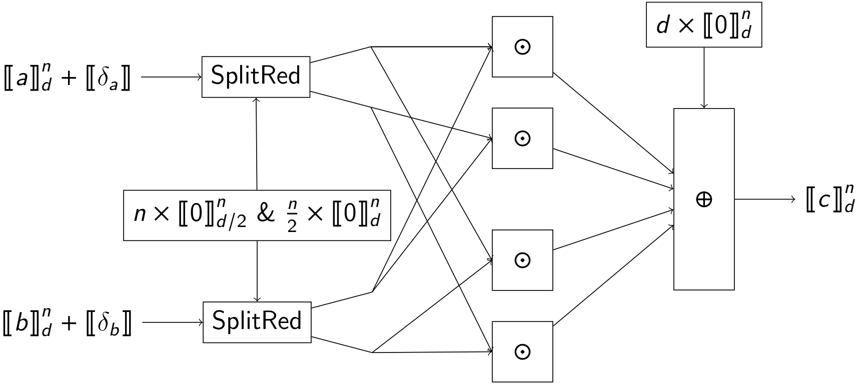
$$
\begin{aligned}
[\![c]\!]^n_d &= [\![a']\!]^n_d \cdot [\![b']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b']\!]^n_d + [\![a']\!]^n_d \cdot [\![b'']\!]^n_d + [\![a'']\!]^n_d \cdot [\![b'']\!]^n_d \\
&= ([\![a']\!]^n_d + [\![a'']\!]^n_d) \cdot ([\![b']\!]^n_d + [\![b'']\!]^n_d) = [\![a]\!]^n_{d/2} \cdot [\![b]\!]^n_{d/2} = [\![a \cdot b]\!]^n_d
\end{aligned}
$$

Security: Fault-Robustness

Security: Fault-Robustness

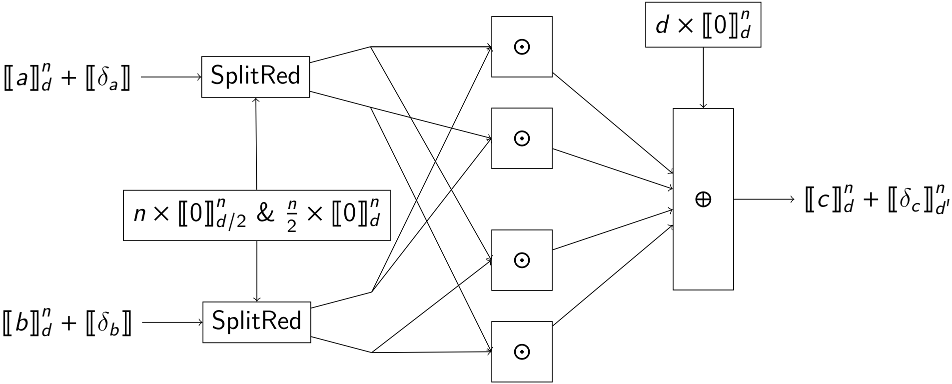


Security: Fault-Robustness



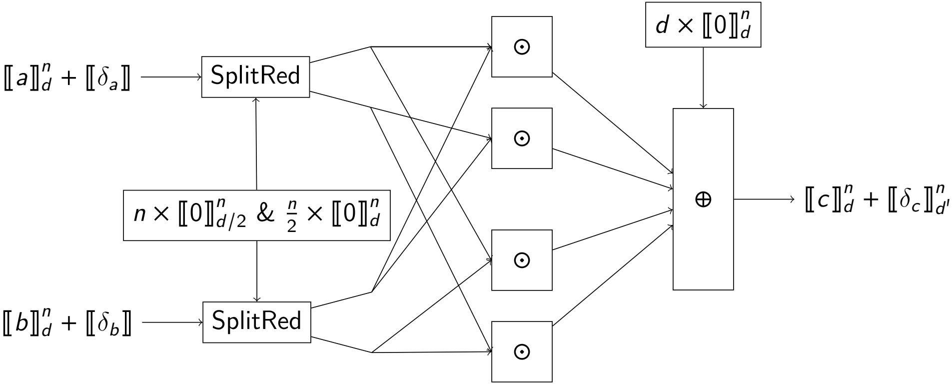
e faults

Security: Fault-Robustness



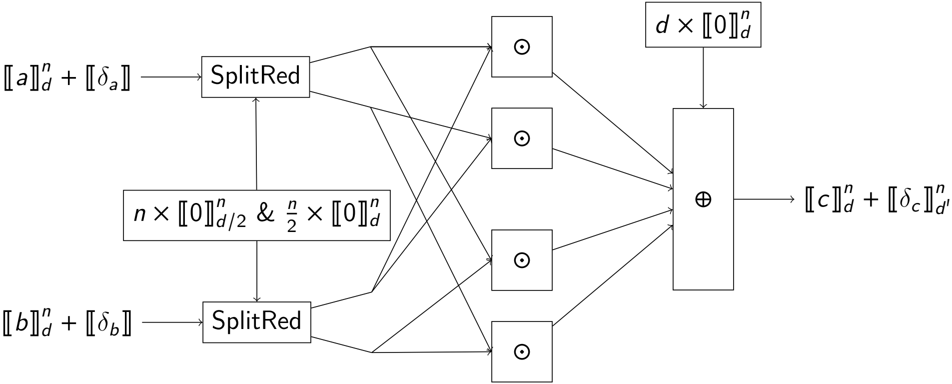
e faults result in $[\![\delta_c]\!]_{d'}^n$

Security: Fault-Robustness



e faults result in $[\![\delta_c]\!]^n_{d'}$ with $d' > d$

Security: Fault-Robustness



e faults result in $[\![\delta_c]\!]_{d'}^n$ with $d' > d$ or $[\![\delta_c]\!]_{d'}^n = (0, \ldots, 0)$

Security: Combined Resilience

Security: Combined Resilience

(a) *e*-Fault Robustness

Security: Combined Resilience

(a) e-Fault Robustness
(b) d-Probing Security (or $d/2$ probes in each gadget)

Security: Combined Resilience

(a) e-Fault Robustness
(b) d-Probing Security (or $d/2$ probes in each gadget)
(c) Fault Invariance

Security: Combined Resilience

(a) e-Fault Robustness

(b) d-Probing Security (or $d/2$ probes in each gadget)

(c) Fault Invariance

- Each intermediate value can be written as a function $f(x_0, x_1, \dots)$ with input values $(x_0, x_1, \dots)$

Security: Combined Resilience

(a) e-Fault Robustness

(b) d-Probing Security (or $d/2$ probes in each gadget)

(c) Fault Invariance

- Each intermediate value can be written as a function $f(x_0, x_1, \dots)$ with input values $(x_0, x_1, \dots)$
- Let f' be the corresponding value in the faulted gadget

Security: Combined Resilience

(a) *e*-Fault Robustness

(b) *d*-Probing Security (or $d/2$ probes in each gadget)

(c) Fault Invariance

- Each intermediate value can be written as a function $f(x_0, x_1, \dots)$ with input values $(x_0, x_1, \dots)$
- Let f' be the corresponding value in the faulted gadget
- There are functions χ, χ_0, χ_1 s.t. $f' = \chi(f(\chi_0(x_0), \chi_1(x_1), \dots))$

Security: Combined Resilience

(a) e-Fault Robustness

(b) d-Probing Security (or $d/2$ probes in each gadget)

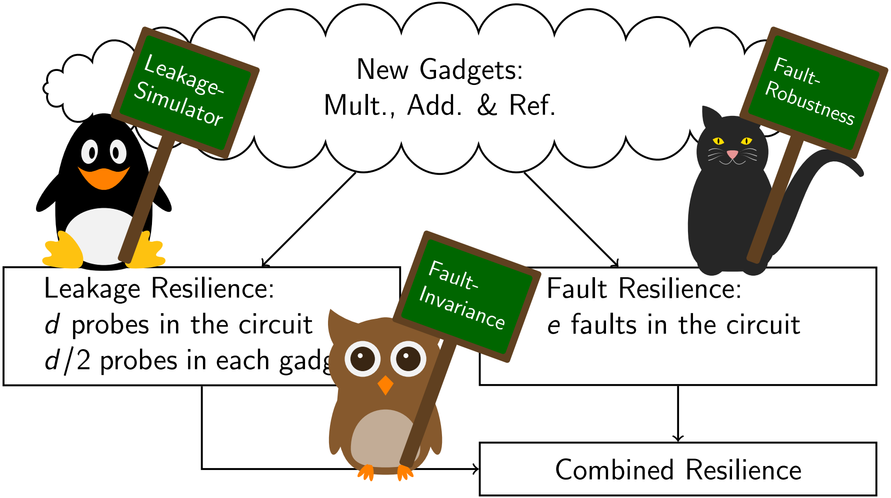
(c) Fault Invariance

- Each intermediate value can be written as a function $f(x_0, x_1, \dots)$ with input values $(x_0, x_1, \dots)$
- Let f' be the corresponding value in the faulted gadget
- There are functions χ, χ_0, χ_1 s.t. $f' = \chi(f(\chi_0(x_0), \chi_1(x_1), \dots))$

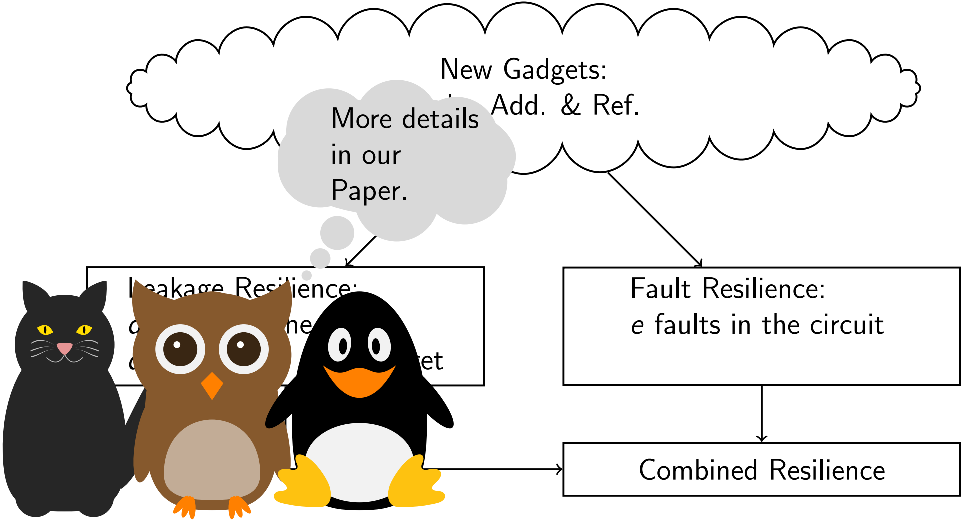
(a)+(b)+(c)= Combined Resilience

Conclusion

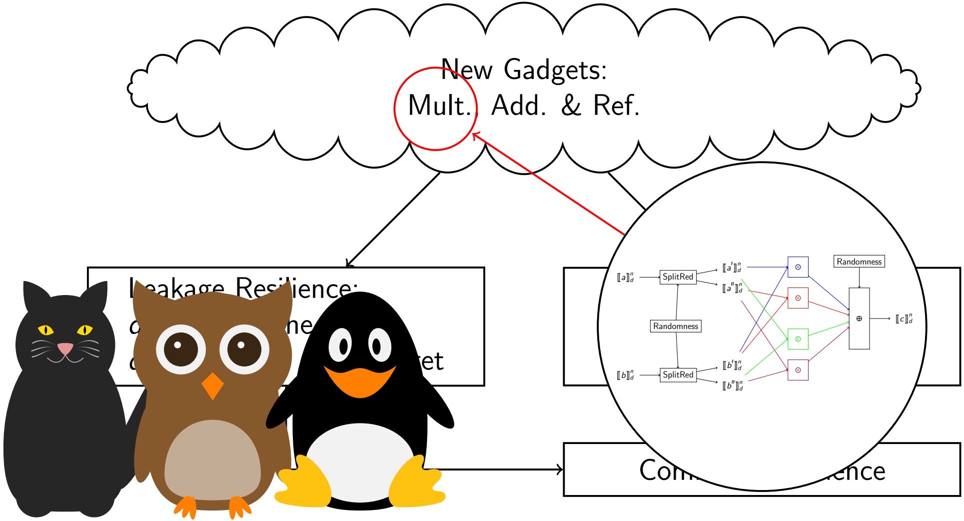
Conclusion


New Gadgets:
Mult., Add. & Ref.
Leakage-Simulator
Fault-Robustness
Fault-Invariance
Leakage Resilience:
d probes in the circuit
$d/2$ probes in each gadg
Fault Resilience:
e faults in the circuit
Combined Resilience

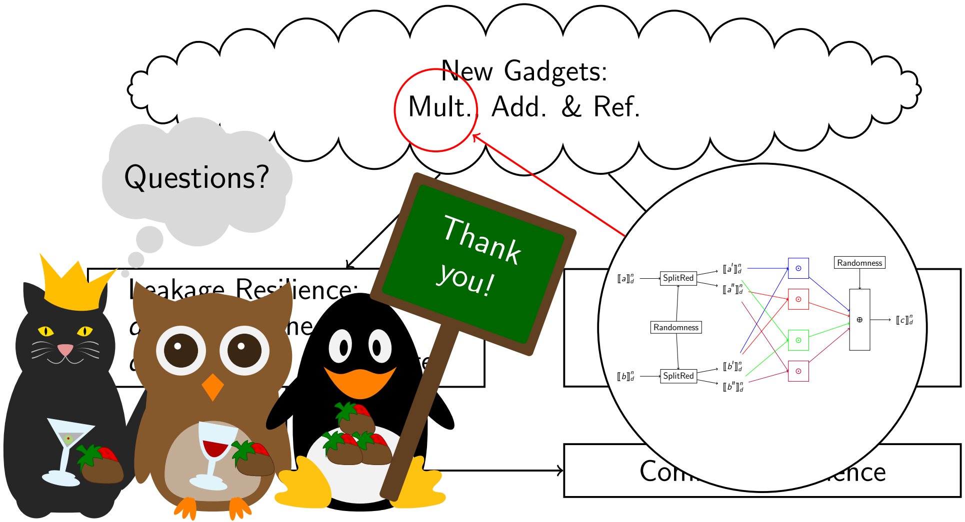
Conclusion


New Gadgets:
Add. & Ref.
More details in our Paper.
Fault Resilience:
e faults in the circuit
Combined Resilience

Conclusion


New Gadgets:
Mult. Add. & Ref.
Leakage Resilience:
Randomness
SplitRed
Randomness
SplitRed

Conclusion


New Gadgets:
Mult. Add. & Ref.
Questions?
Thank you!
SplitRed
Randomness
SplitRed
Randomness