Higher-Order Threshold Implementations

Begül Bilgin, Benedikt Gierlichs, Svetla Nikova,
Ventzislav Nikov, and Vincent Rijmen

COSIC

KU LEUVEN

NXP

UNIVERSITEIT TWENTE.

Higher-Order Threshold Implementations

Higher-Order Threshold Implementations

In a nutshell:

Higher-Order Threshold Implementations

In a nutshell:

- A countermeasure against Higher-Order Differential Power Analysis (HO-DPA)

Higher-Order Threshold Implementations

In a nutshell:

- A countermeasure against Higher-Order Differential Power Analysis (HO-DPA)
- Ideas from secret sharing and multi-party computation

Higher-Order Threshold Implementations

In a nutshell:

- A countermeasure against Higher-Order Differential Power Analysis (HO-DPA)
- Ideas from secret sharing and multi-party computation
- Can be applied to any algorithm

Higher-Order Threshold Implementations

In a nutshell:

- A countermeasure against Higher-Order Differential Power Analysis (HO-DPA)
- Ideas from secret sharing and multi-party computation
- Can be applied to any algorithm
- Independent of technology, library, etc.

Higher-Order Threshold Implementations

In a nutshell:

- A countermeasure against Higher-Order Differential Power Analysis (HO-DPA)
- Ideas from secret sharing and multi-party computation
- Can be applied to any algorithm
- Independent of technology, library, etc.
- Efficient

Higher-Order Threshold Implementations

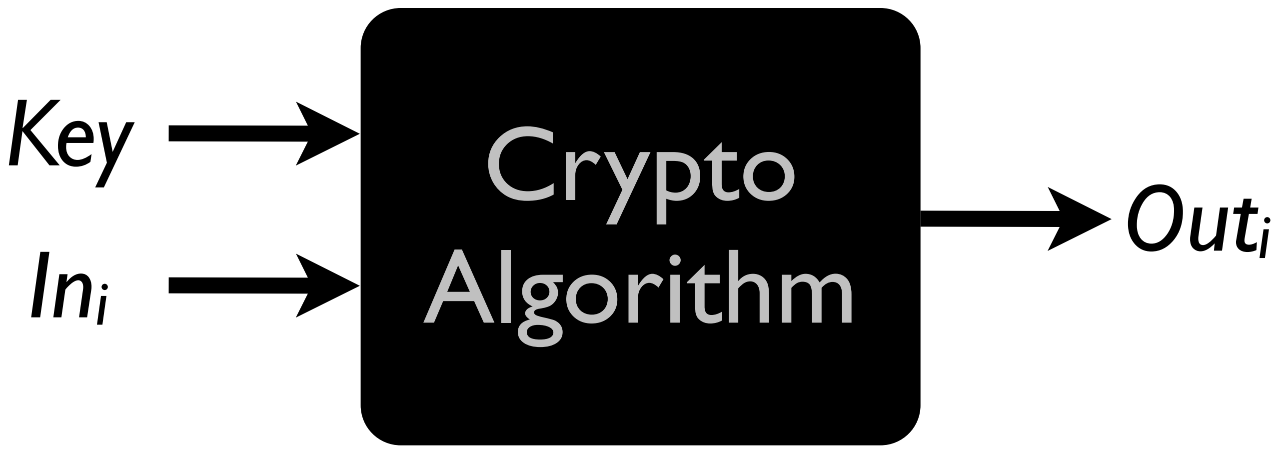
In a nutshell:

- A countermeasure against Higher-Order Differential Power Analysis (HO-DPA)
- Ideas from secret sharing and multi-party computation
- Can be applied to any algorithm
- Independent of technology, library, etc.
- Efficient
- Application to KATAN-32

Background

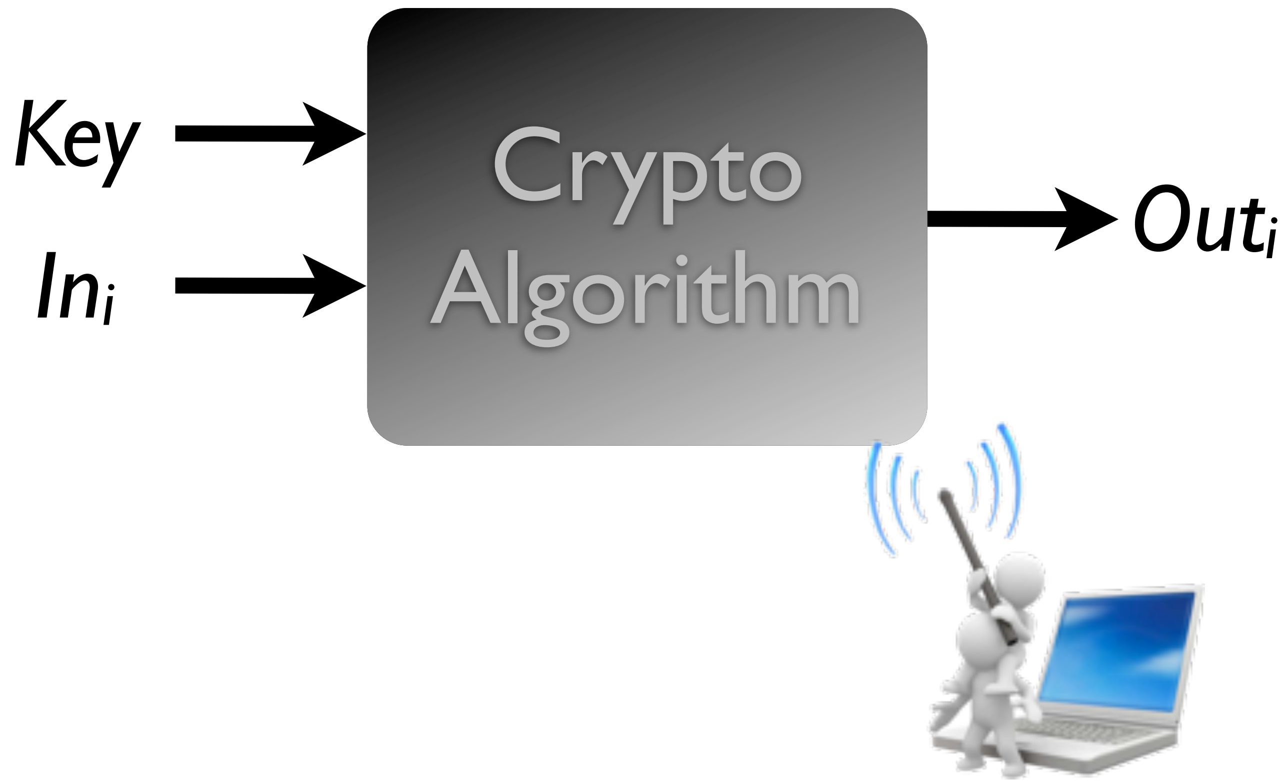
Differential Power Analysis
&
Its Countermeasures

Background - DPA

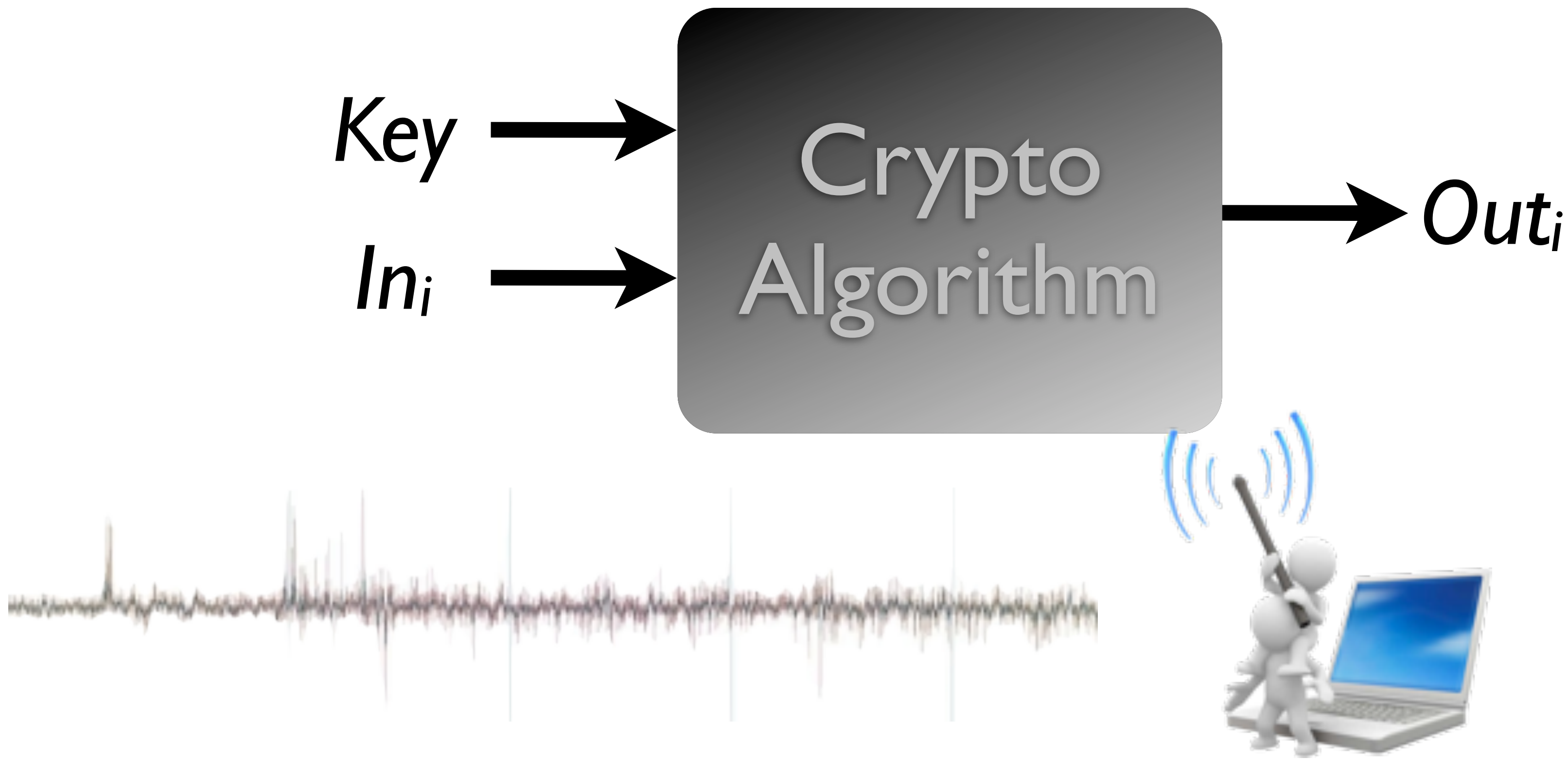


Background - DPA

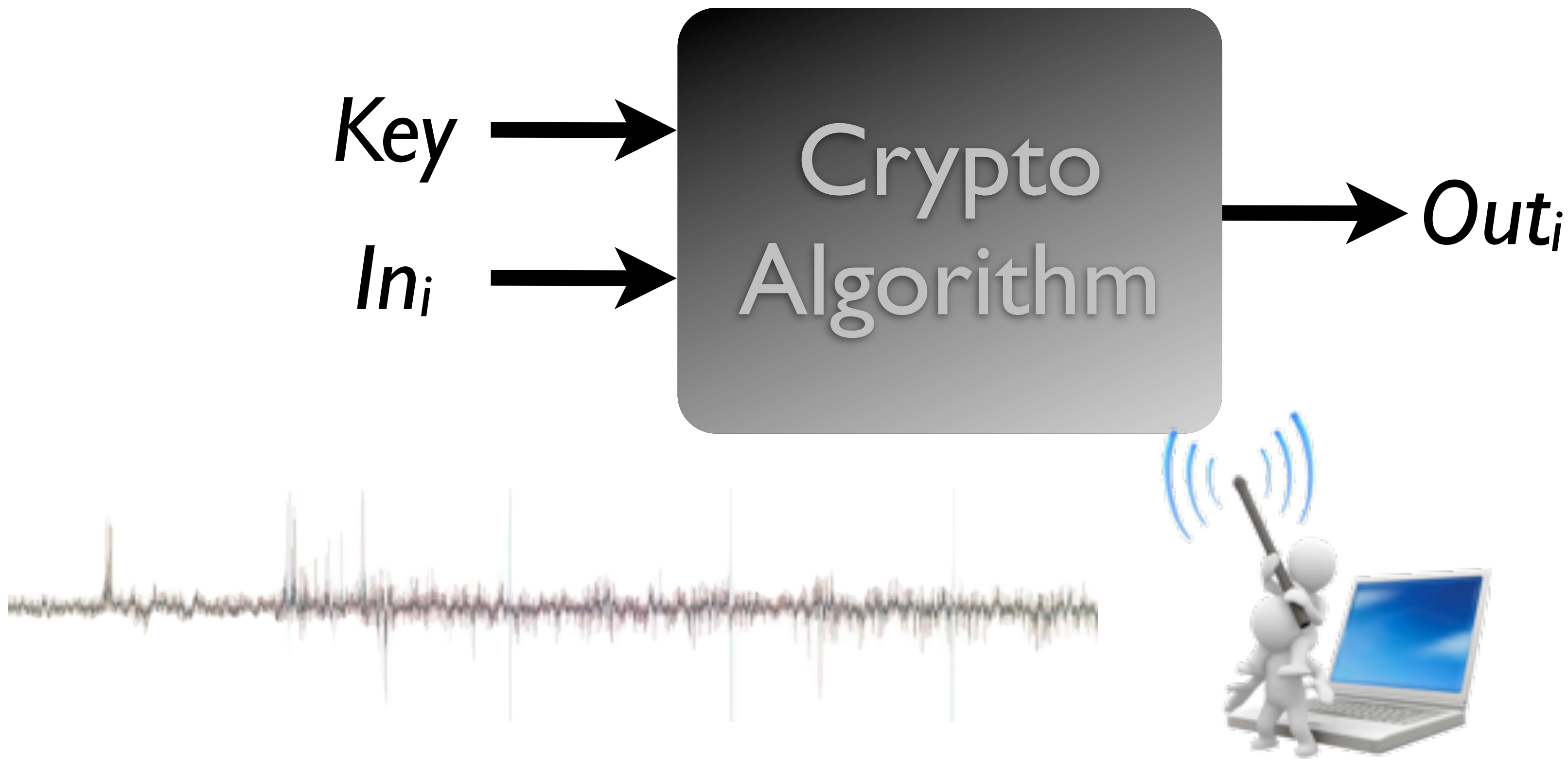
Background - DPA



Background - DPA

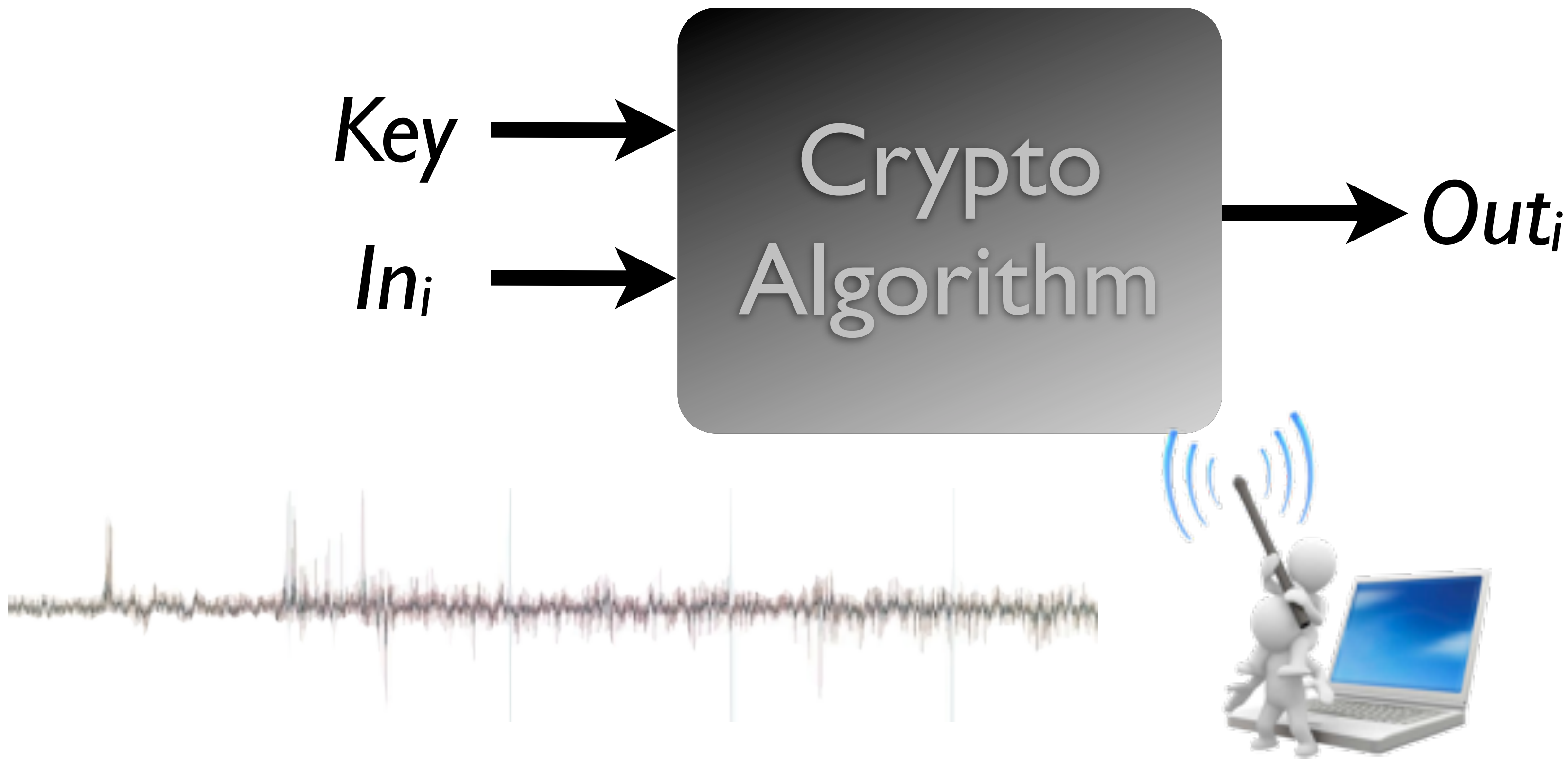


Background - DPA



DPA focuses on finding the secret using the relation between

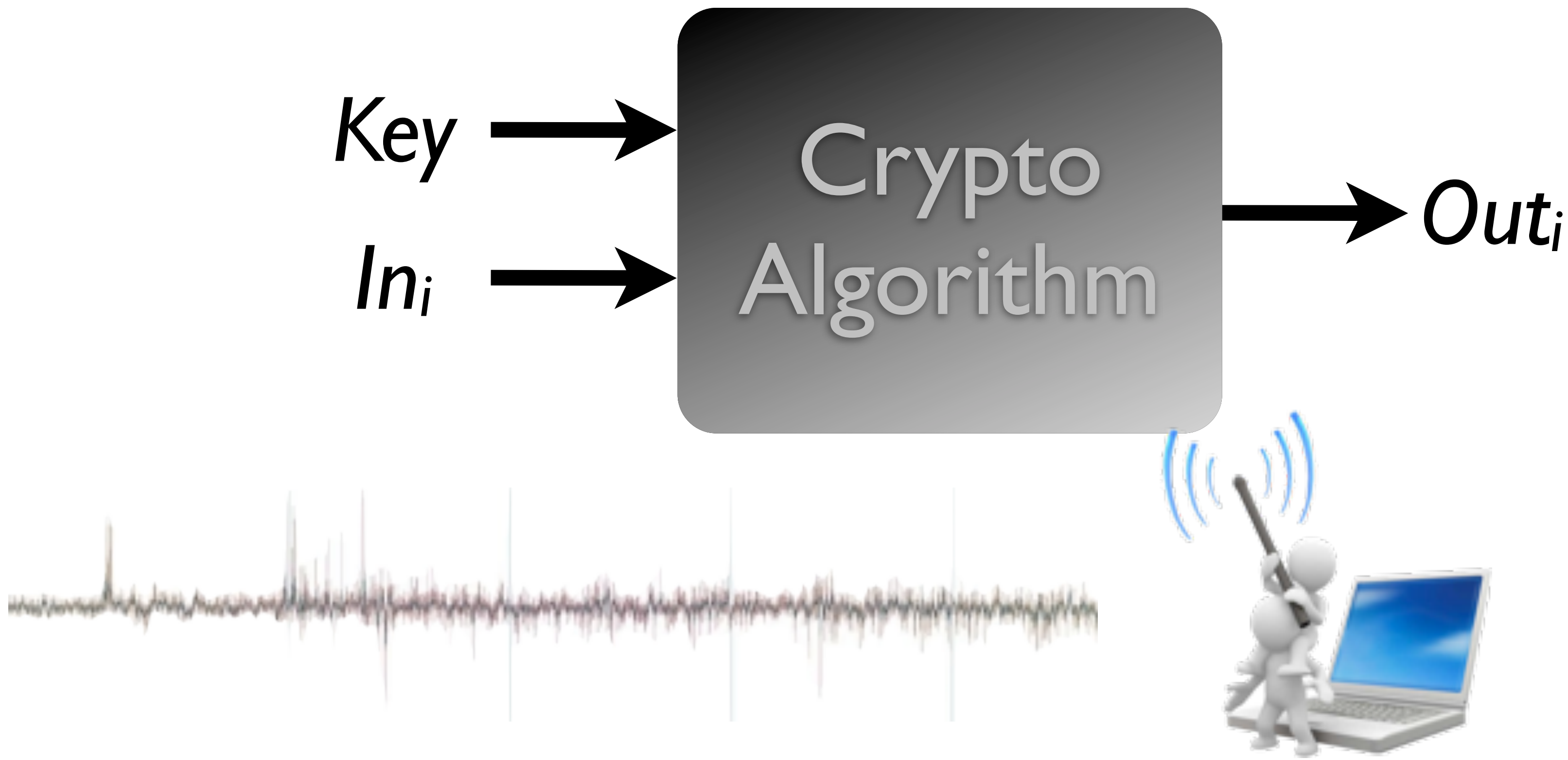
Background - DPA



DPA focuses on finding the secret using the relation between

- Instantaneous power consumption

Background - DPA



DPA focuses on finding the secret using the relation between

- Instantaneous power consumption
- And intermediate results of the algorithm

Background - DPA Countermeasures

Background - DPA Countermeasures

Countermeasure can be

Background - DPA Countermeasures

Countermeasure can be

- Limiting the usage of key

Background - DPA Countermeasures

Countermeasure can be

- Limiting the usage of key
- Decreasing the SNR

Background - DPA Countermeasures

Countermeasure can be

- Limiting the usage of key
- Decreasing the SNR
- Breaking the relation between the trace and (In_i, Out_i)

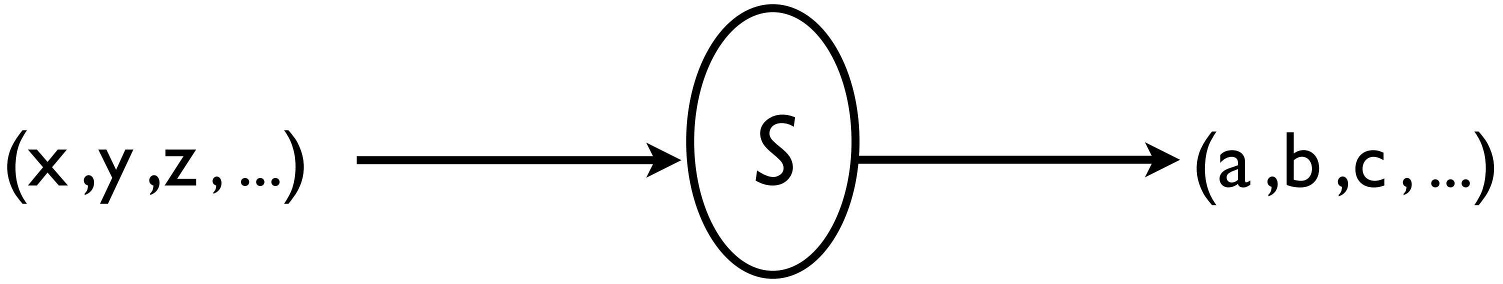
Background - DPA Countermeasures

Countermeasure can be

- Limiting the usage of key
- Decreasing the SNR
- Breaking the relation between the trace and (In_i, Out_i)

  e.g. Masking

Background - Boolean Masking



Background - Boolean Masking

Background - Boolean Masking

$(x_1, y_1, z_1, \ldots)$

Background - Boolean Masking

$$(x_1, y_1, z_1, \ldots)$$

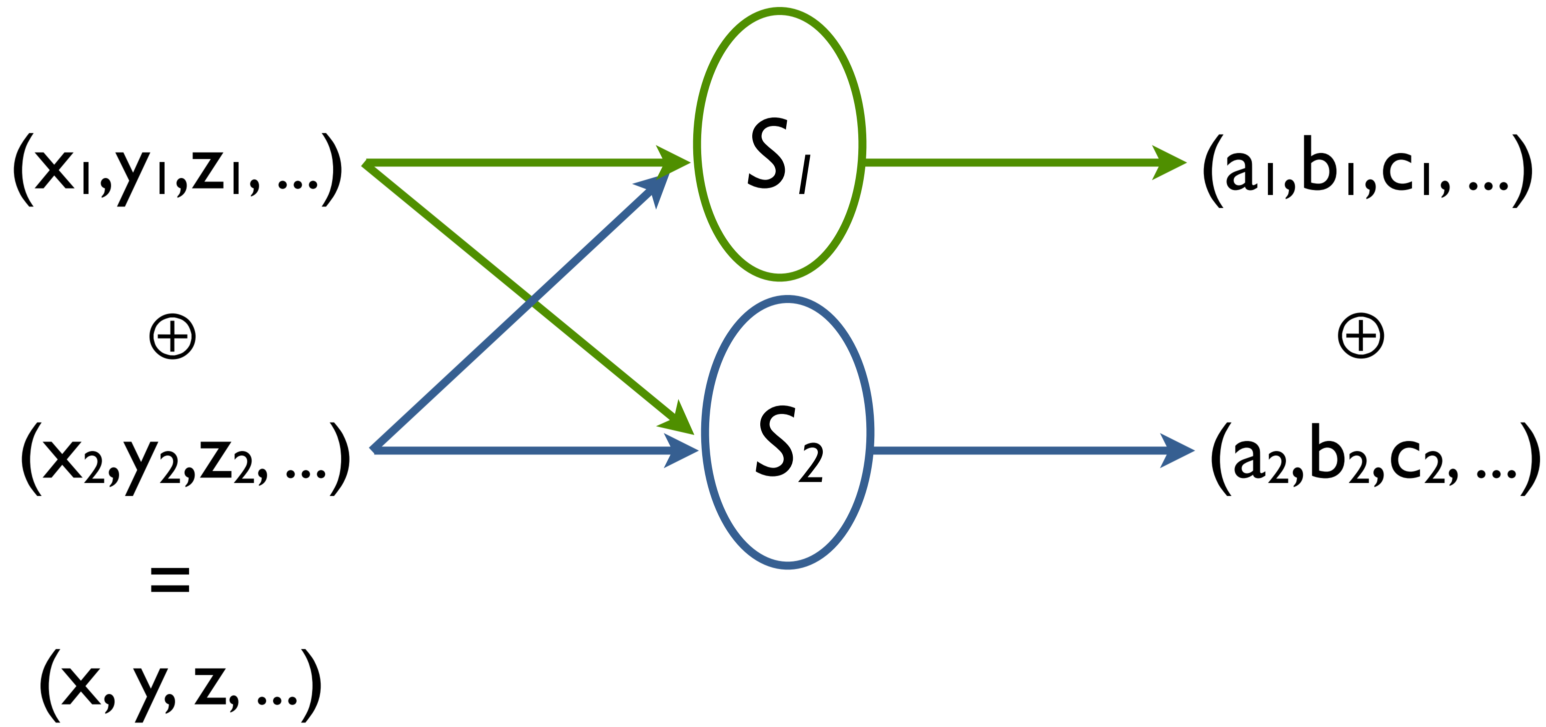
$$\oplus$$

$$(x_2, y_2, z_2, \ldots)$$

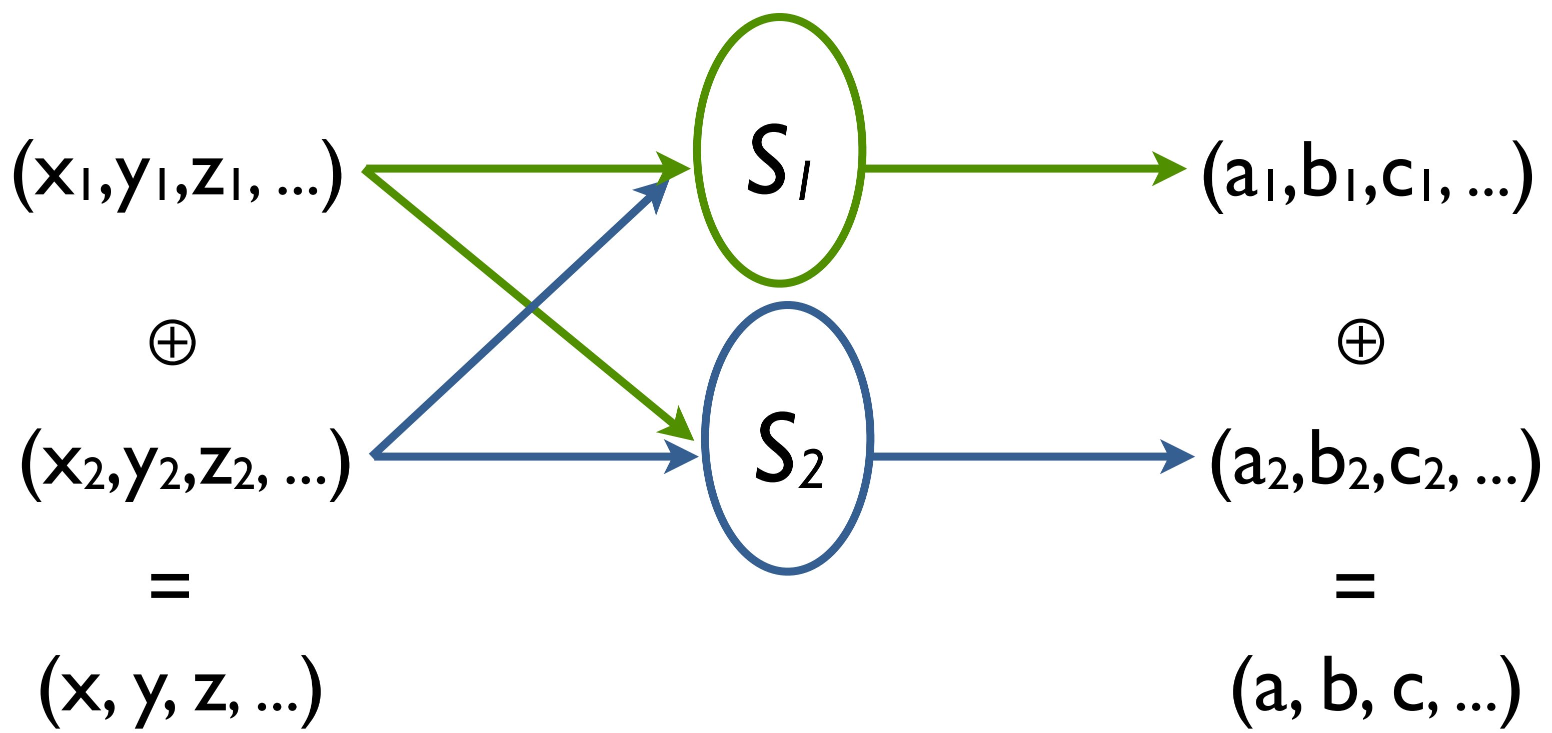
$$=$$

$$(x, y, z, \ldots)$$

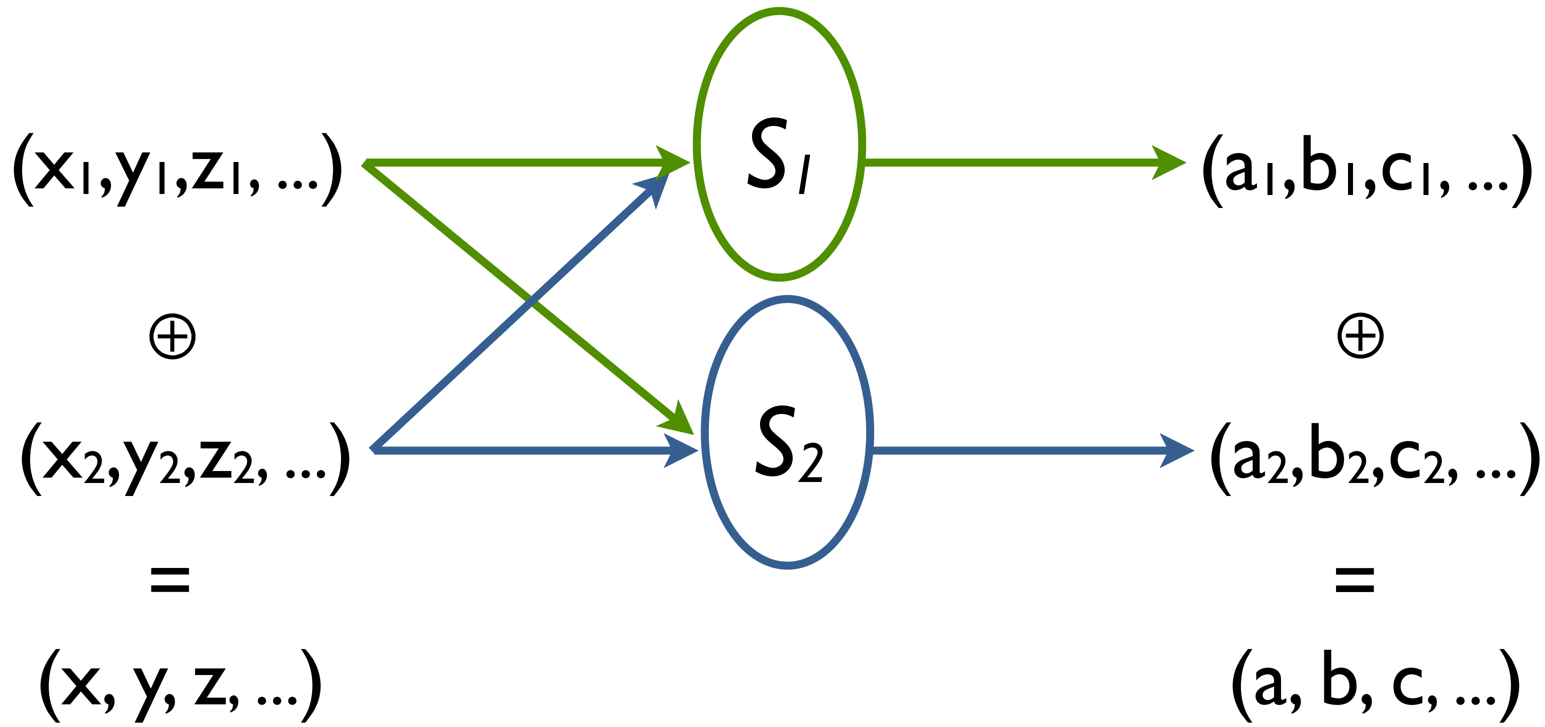
Background - Boolean Masking



Background - Boolean Masking



Background - Boolean Masking



Random input/output shares ➡ Random intermediate values

Background - Higher-Order DPA

Background - Higher-Order DPA

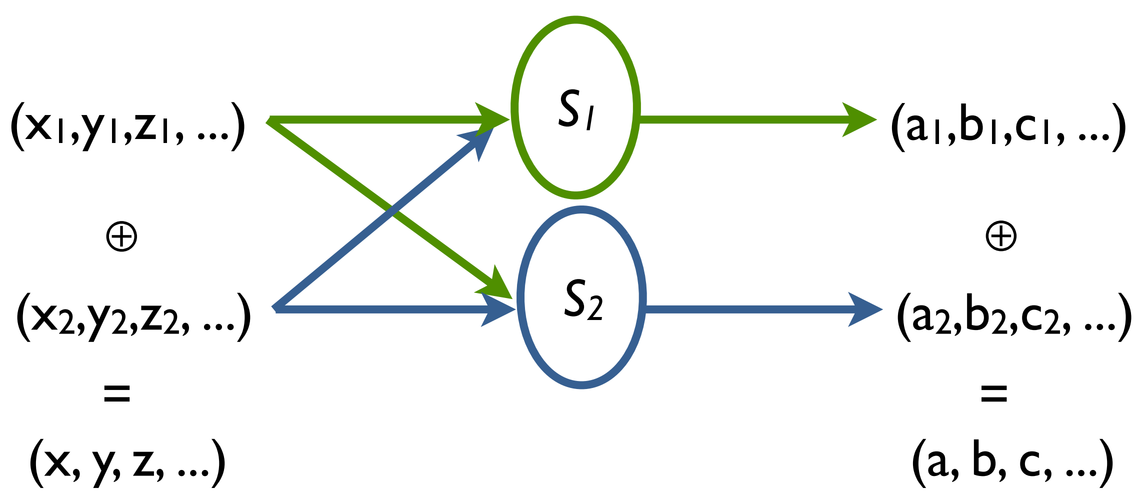
d^{th}-order DPA $\rightleftarrows$ d probing model

Lemma: Attack order in higher-order DPA corresponds to number of wires probed in the circuit (per unmasked bit).

Background - Higher-Order DPA

d^{th}-order DPA ⇄ d probing model

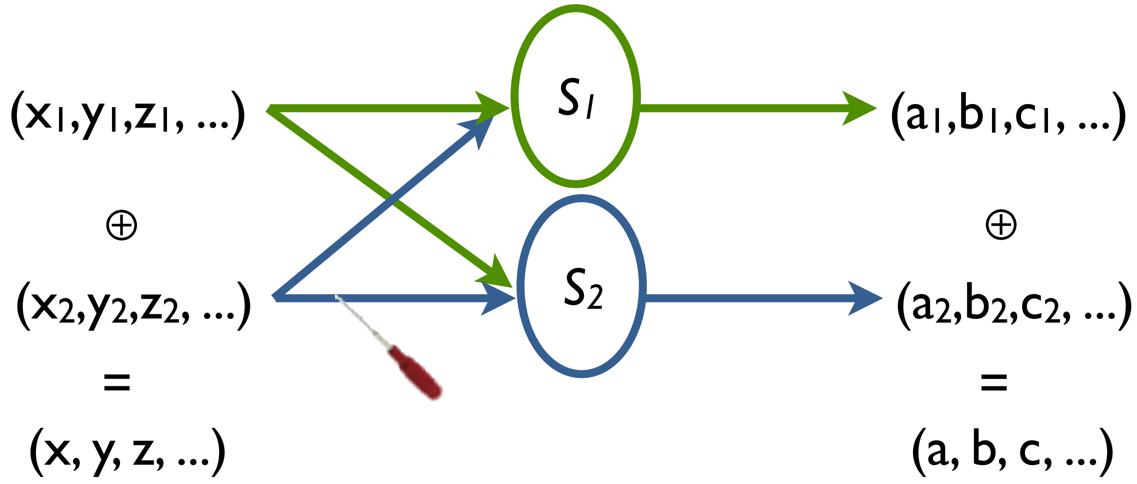
<u>Lemma:</u> Attack order in higher-order DPA corresponds to number of wires probed in the circuit (per unmasked bit).



Background - Higher-Order DPA

d^{th}-order DPA ⇄ d probing model

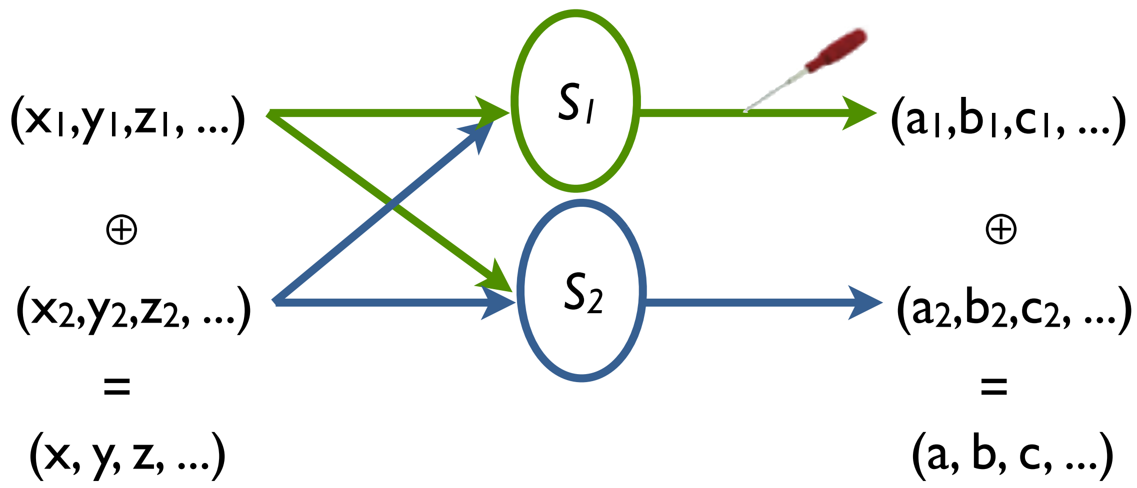
Lemma: Attack order in higher-order DPA corresponds to number of wires probed in the circuit (per unmasked bit).



Background - Higher-Order DPA

d^{th}-order DPA $\rightleftarrows$ d probing model

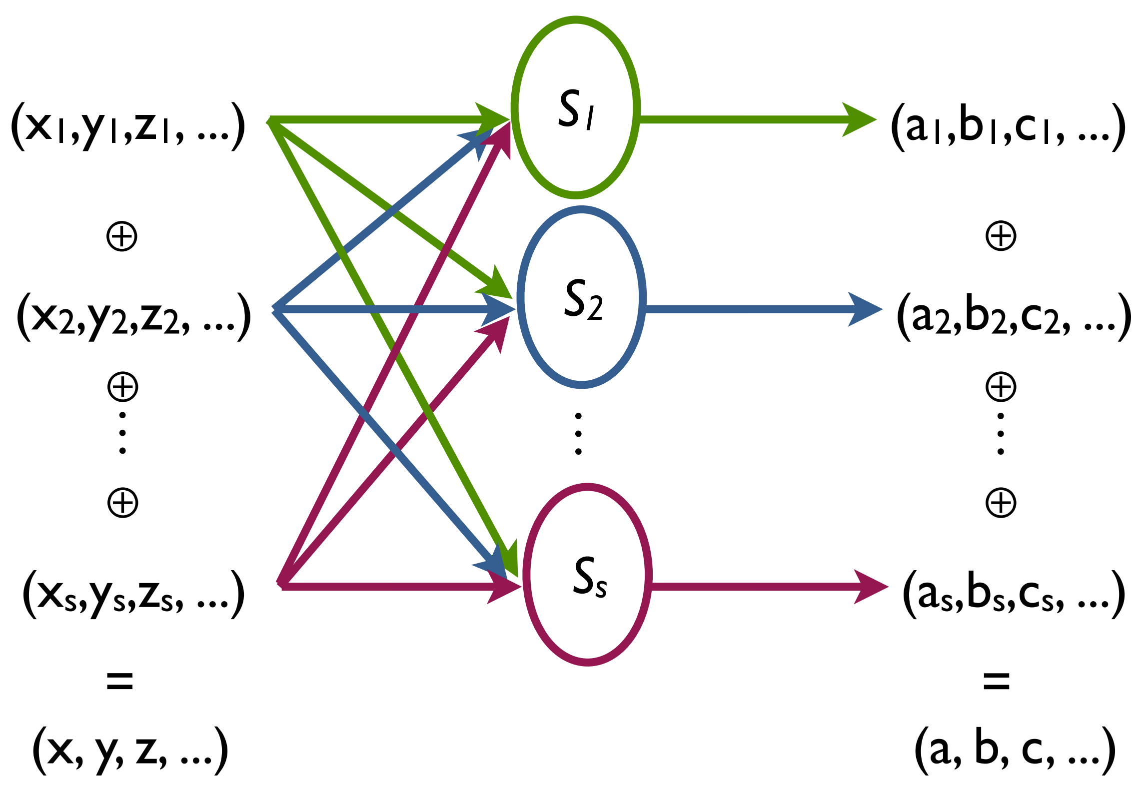
<u>Lemma:</u> Attack order in higher-order DPA corresponds to number of wires probed in the circuit (per unmasked bit).



Background - Higher-Order DPA

d^{th}-order DPA ⇄ d probing model

<u>Lemma:</u> Attack order in higher-order DPA corresponds to number of wires probed in the circuit (per unmasked bit).



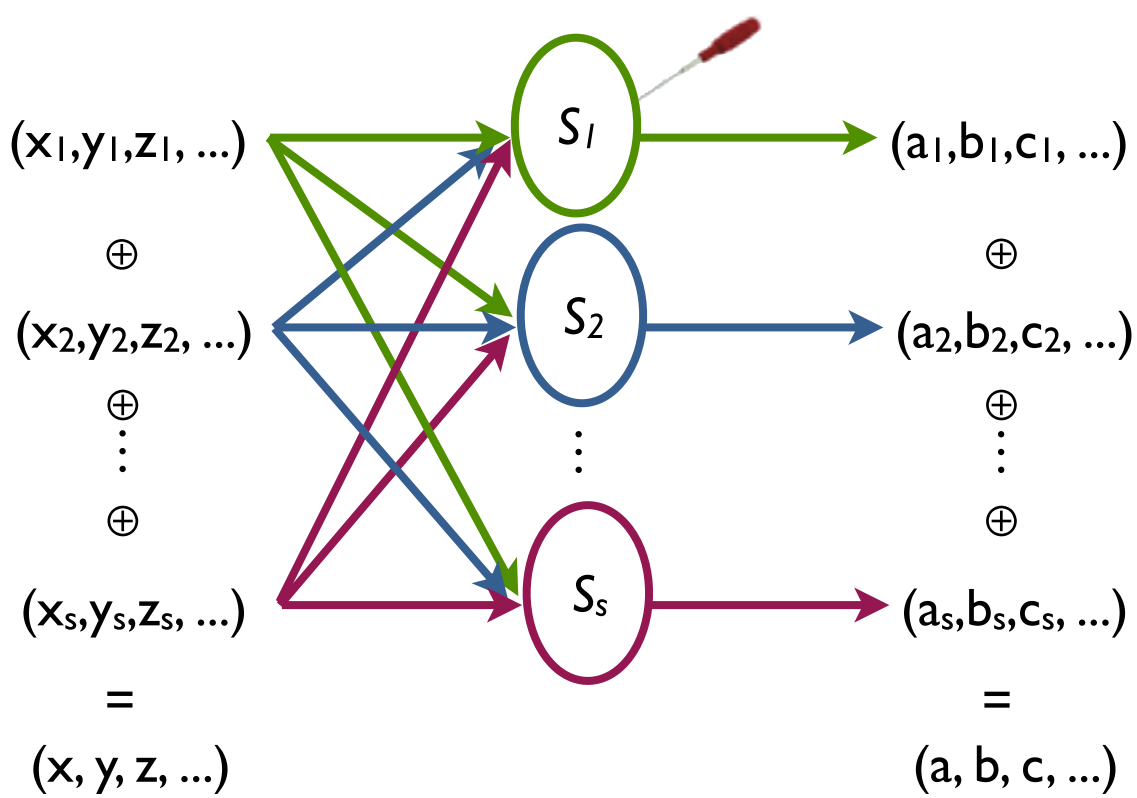
Boolean Masking:

- #shares > d

Background - Higher-Order DPA

d^{th}-order DPA $\rightleftarrows$ d probing model

<u>Lemma:</u> Attack order in higher-order DPA corresponds to number of wires probed in the circuit (per unmasked bit).



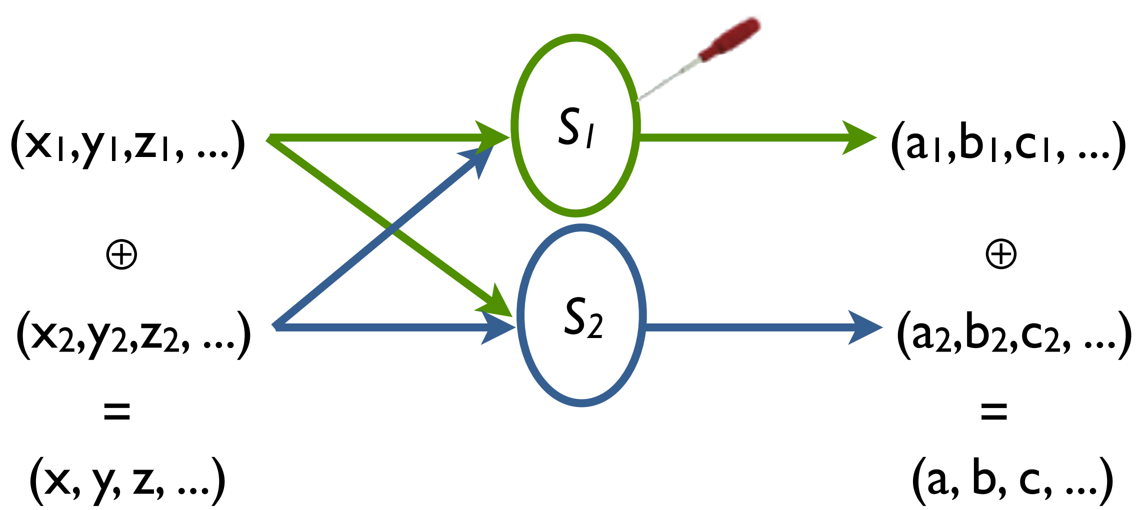
Boolean Masking:

- #shares > d
- glitches reduce the security

Background - Higher-Order DPA

d^{th}-order DPA ⇄ d probing model

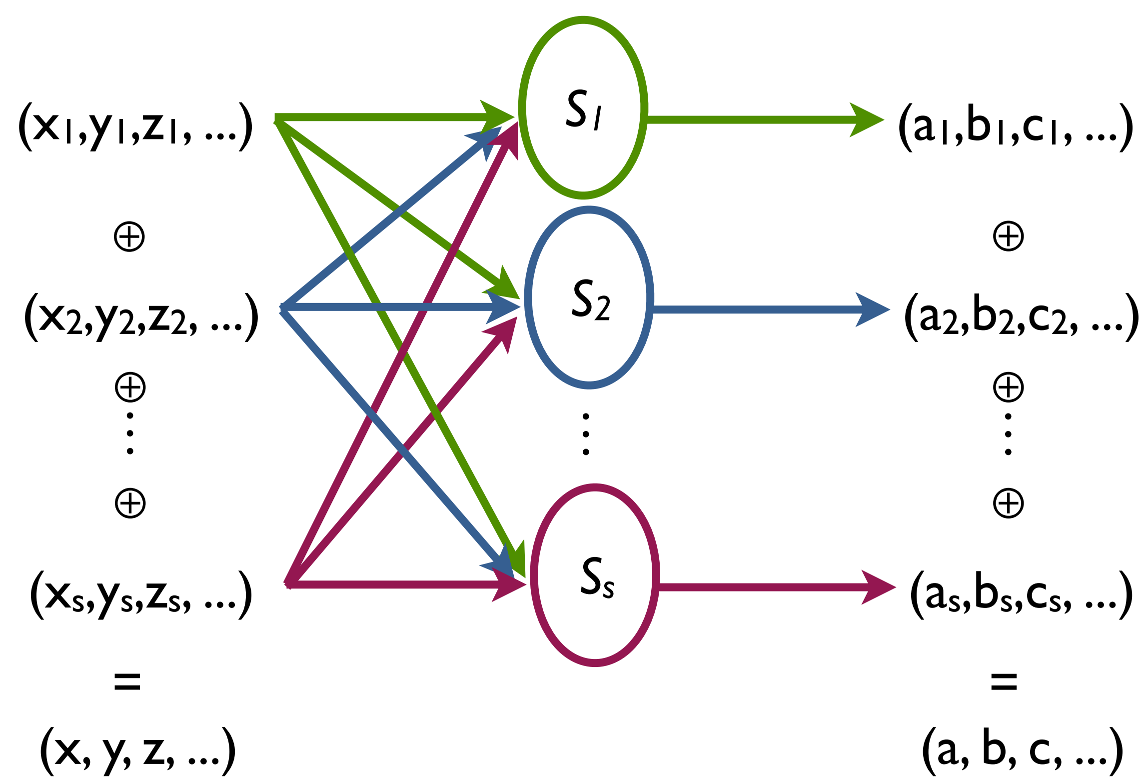
<u>Lemma:</u> Attack order in higher-order DPA corresponds to number of wires probed in the circuit (per unmasked bit).



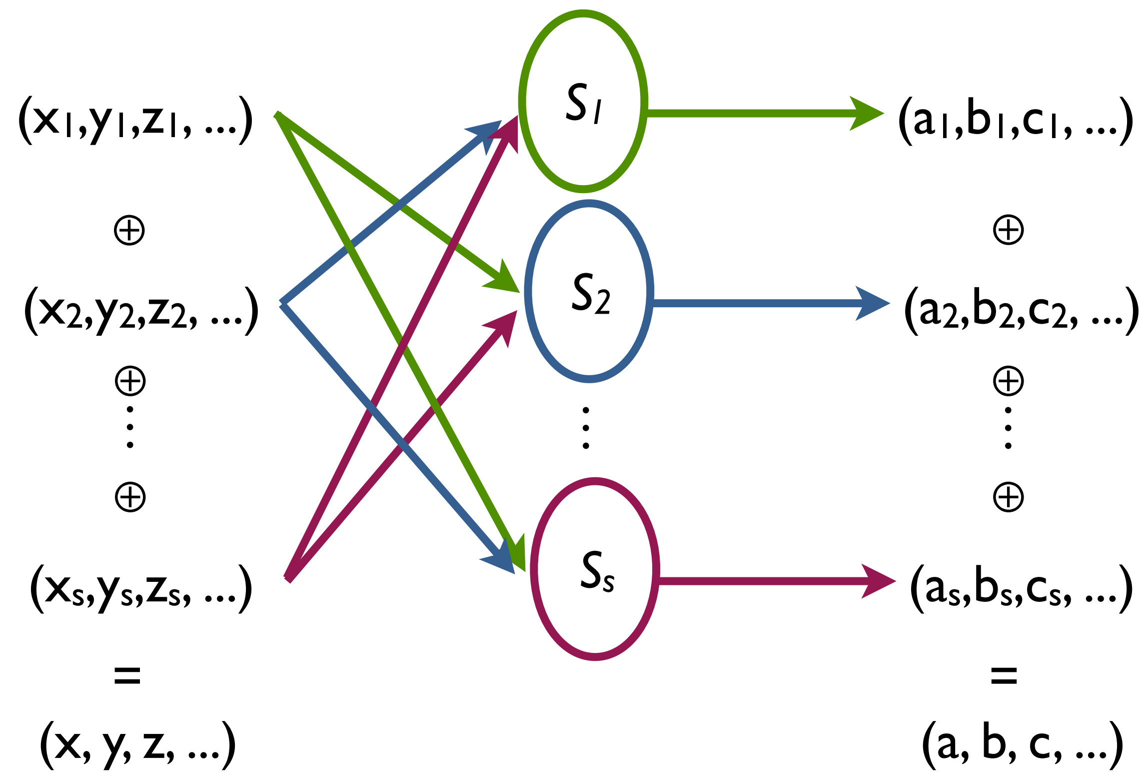
Boolean Masking:

- #shares > d
- glitches reduce the security

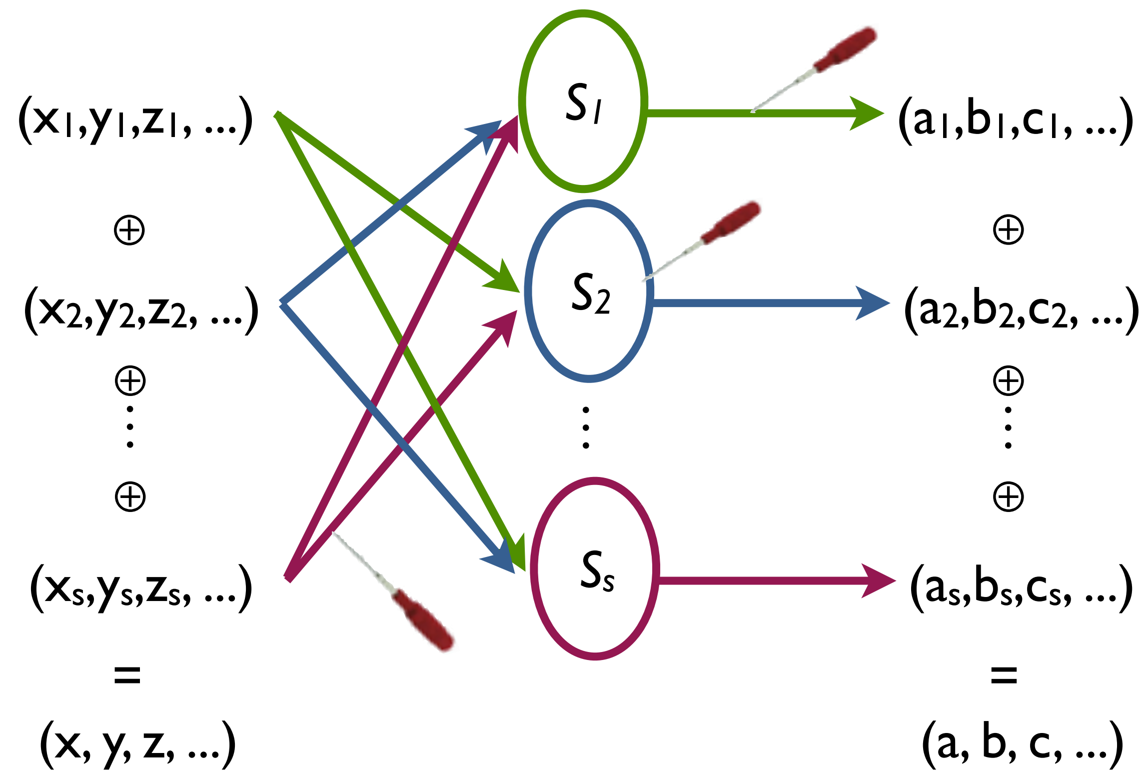
Background - Countermeasures



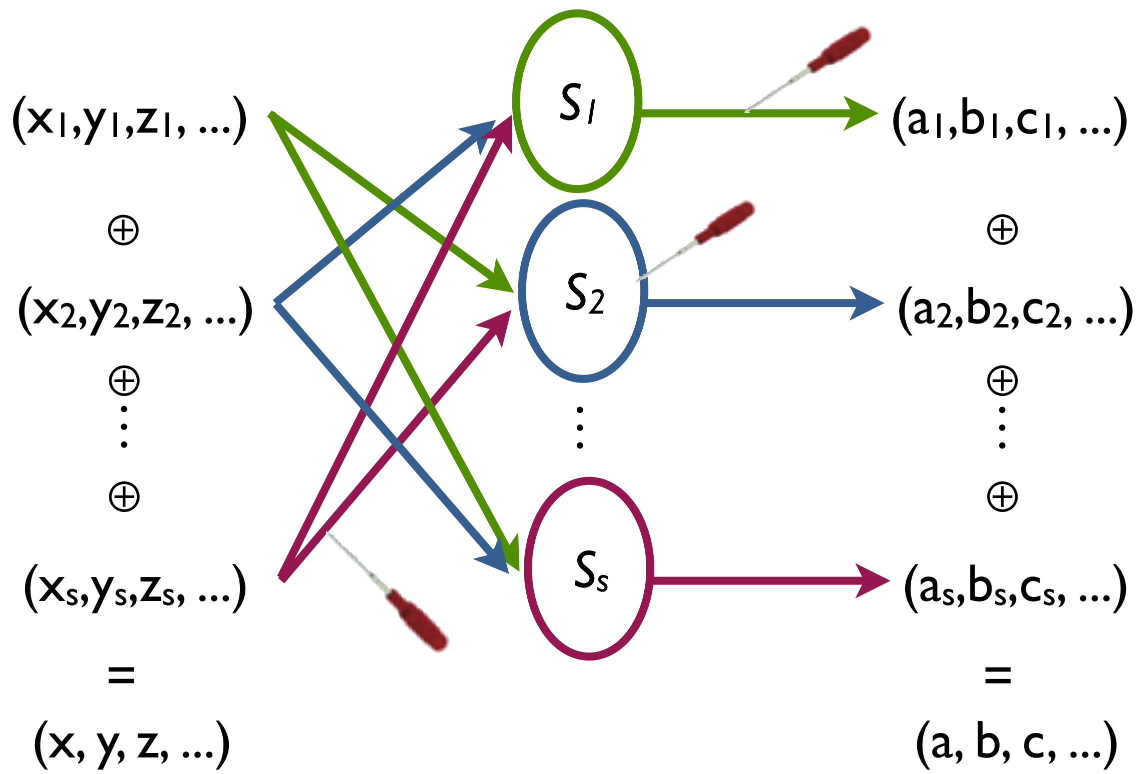
Background - Countermeasures



Background - Countermeasures

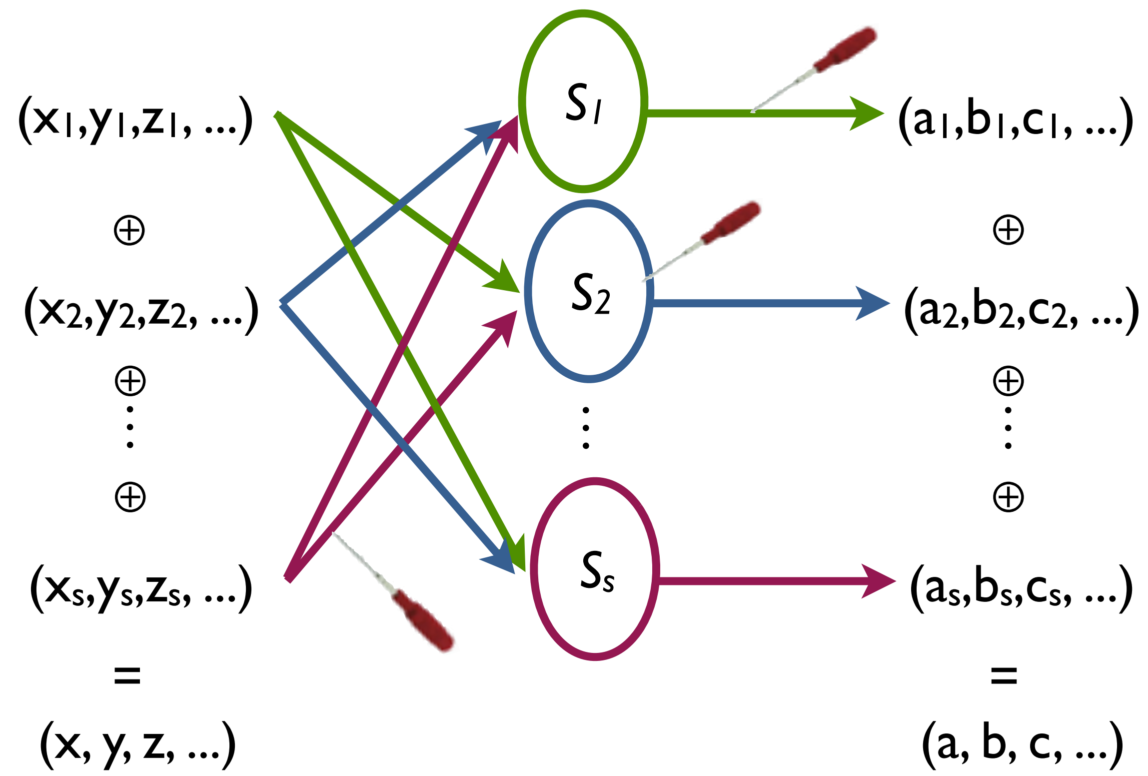


Background - Countermeasures



Such as:

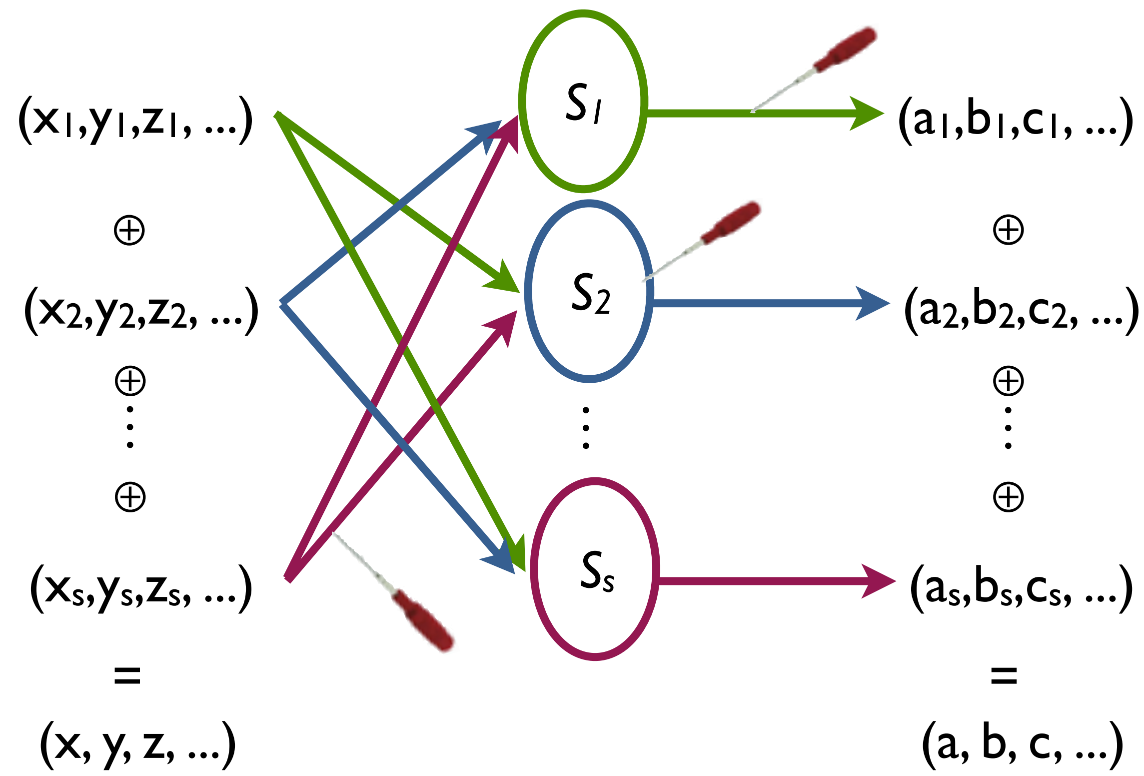
Background - Countermeasures



Such as:

- (*d=1*) Threshold Implementations (ICICS'06)

Background - Countermeasures

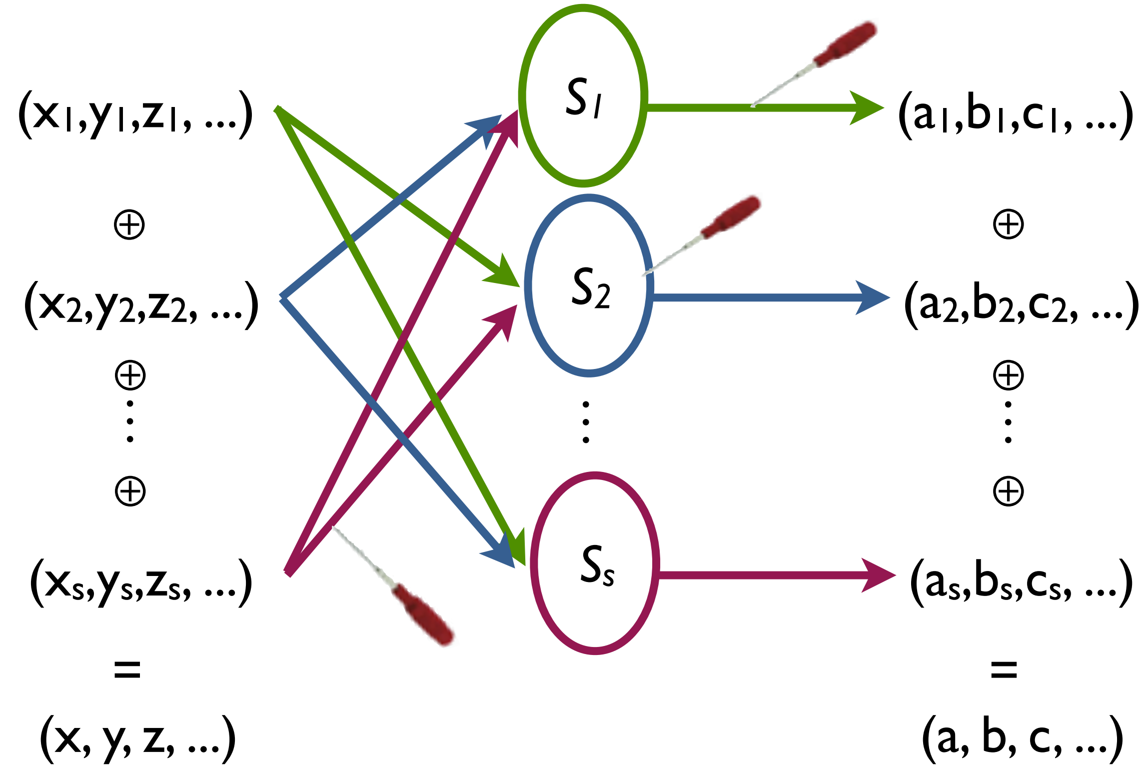


Such as:

- (*d=1*) Threshold Implementations (ICICS'06)

  #shares > t (algebraic degree of S = t)

Background - Countermeasures



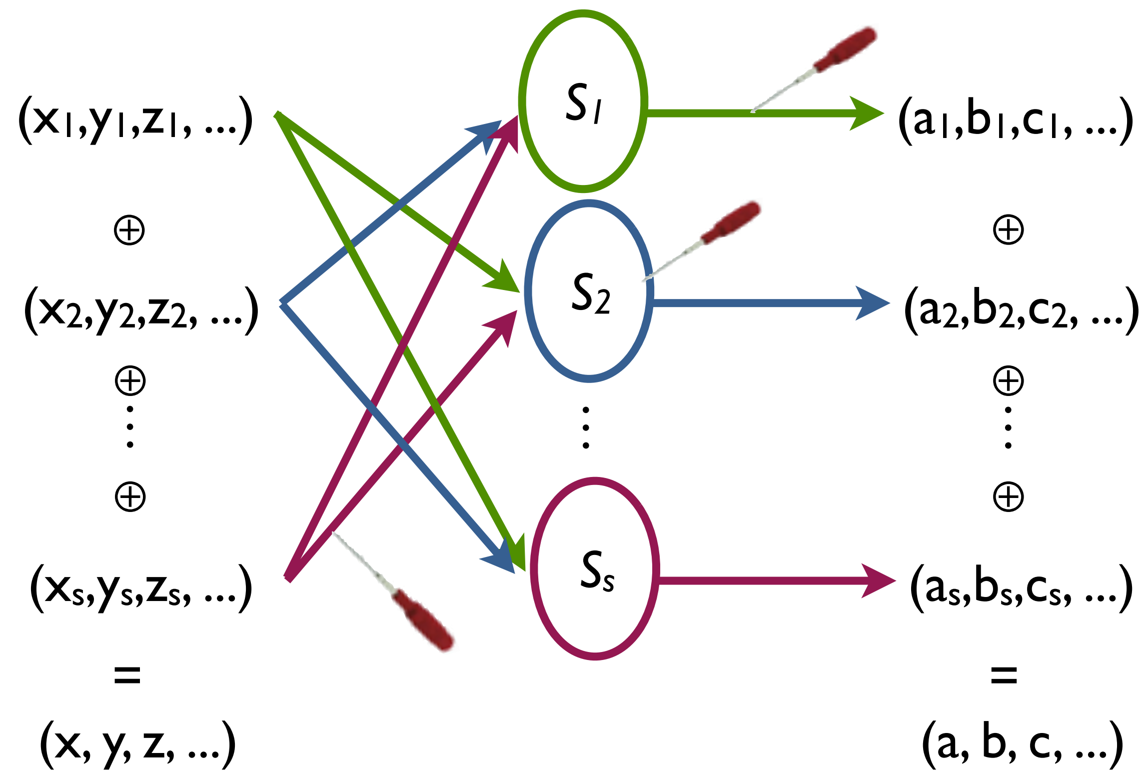
Such as:

- (*d=1*) Threshold Implementations (ICICS'06)

  #shares > t (algebraic degree of S = t)

- Prouff & Roche (CHES'11) ~ BGW scheme

Background - Countermeasures



Such as:

- (*d=1*) Threshold Implementations (ICICS'06)

  #shares > t (algebraic degree of S = t)

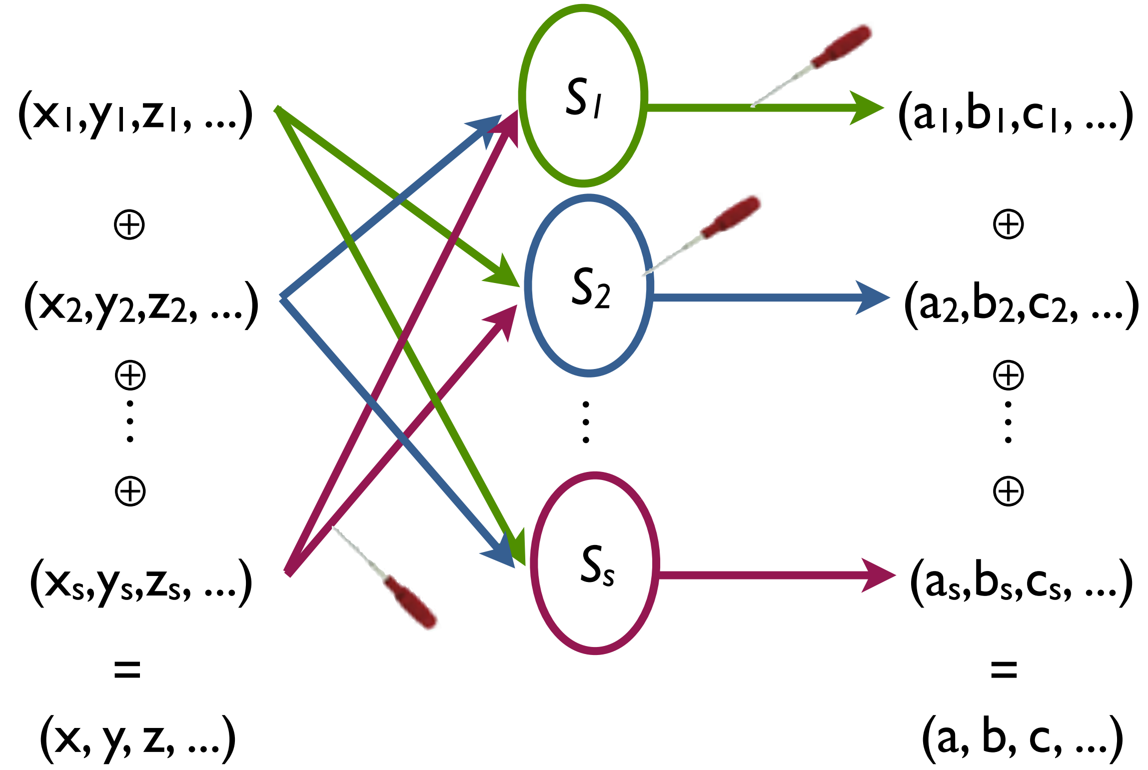
- Prouff & Roche (CHES'11) ~ BGW scheme

  #shares > 2d

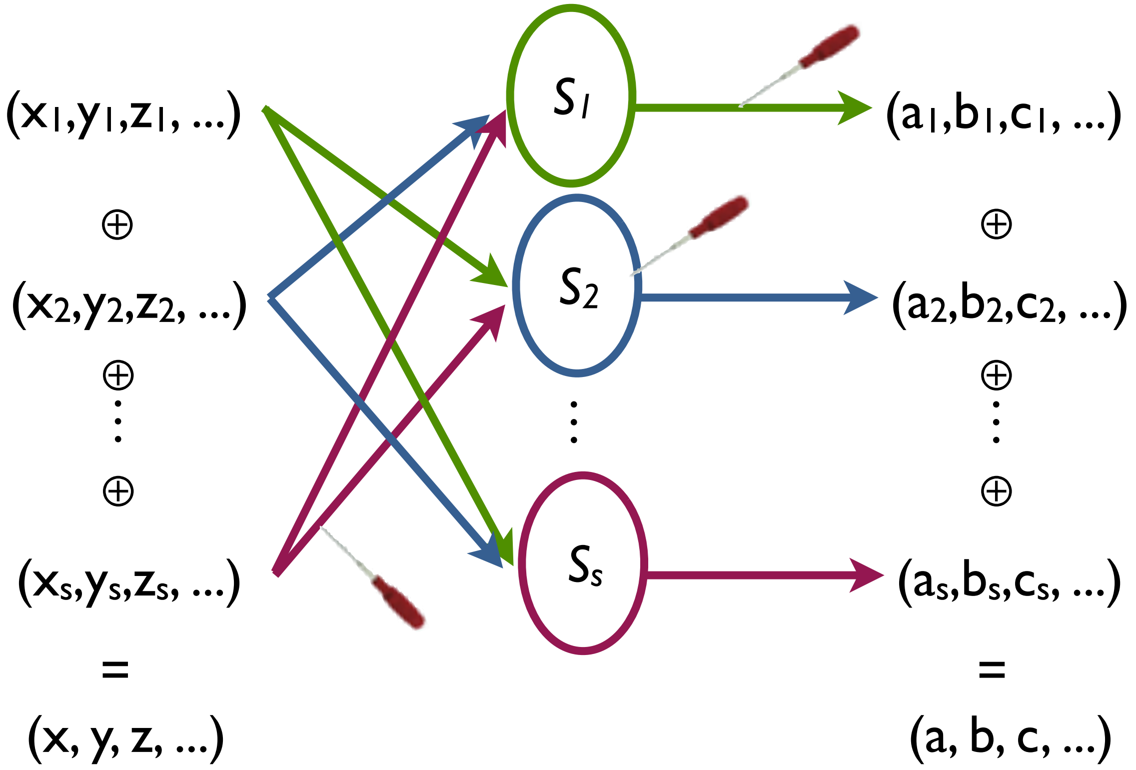
Higher-Order Threshold Implementations

Properties
&
Requirements

Threshold Implementations (*d*=*1*)



Threshold Implementations (*d*=*1*)

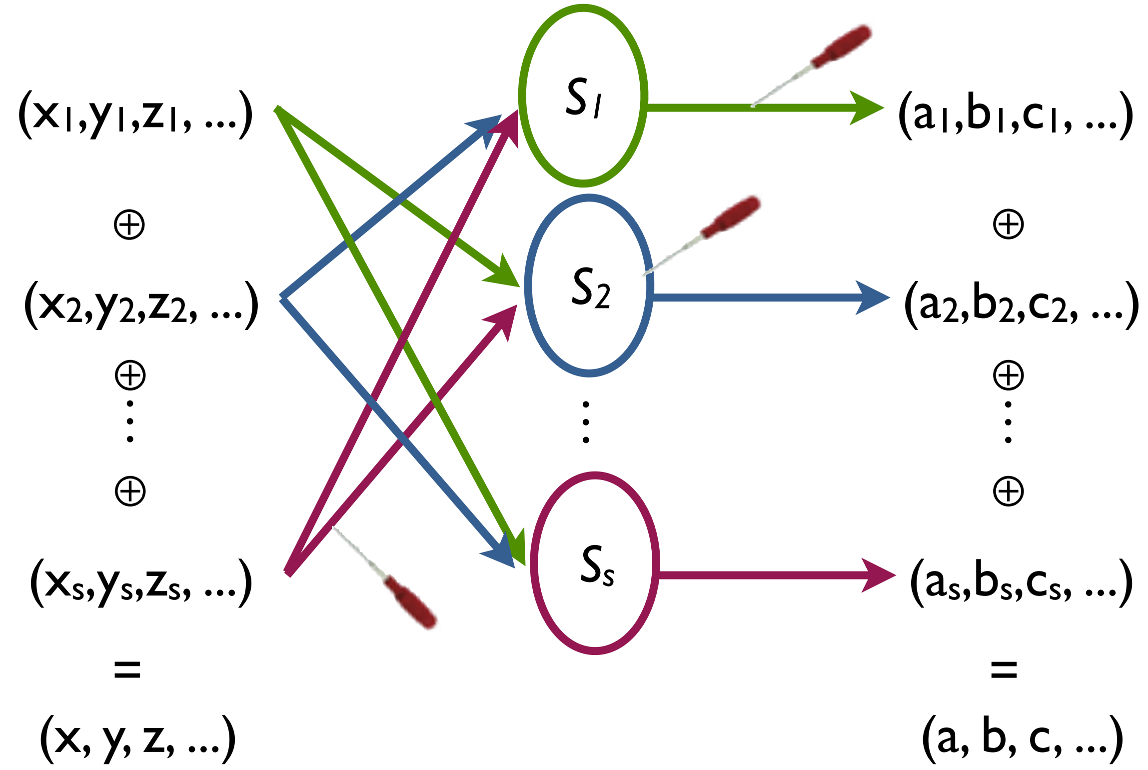


Uniform input masking

Correctness

Non-completeness: Every function is independent of at least one input share.

Higher-Order Threshold Implementations

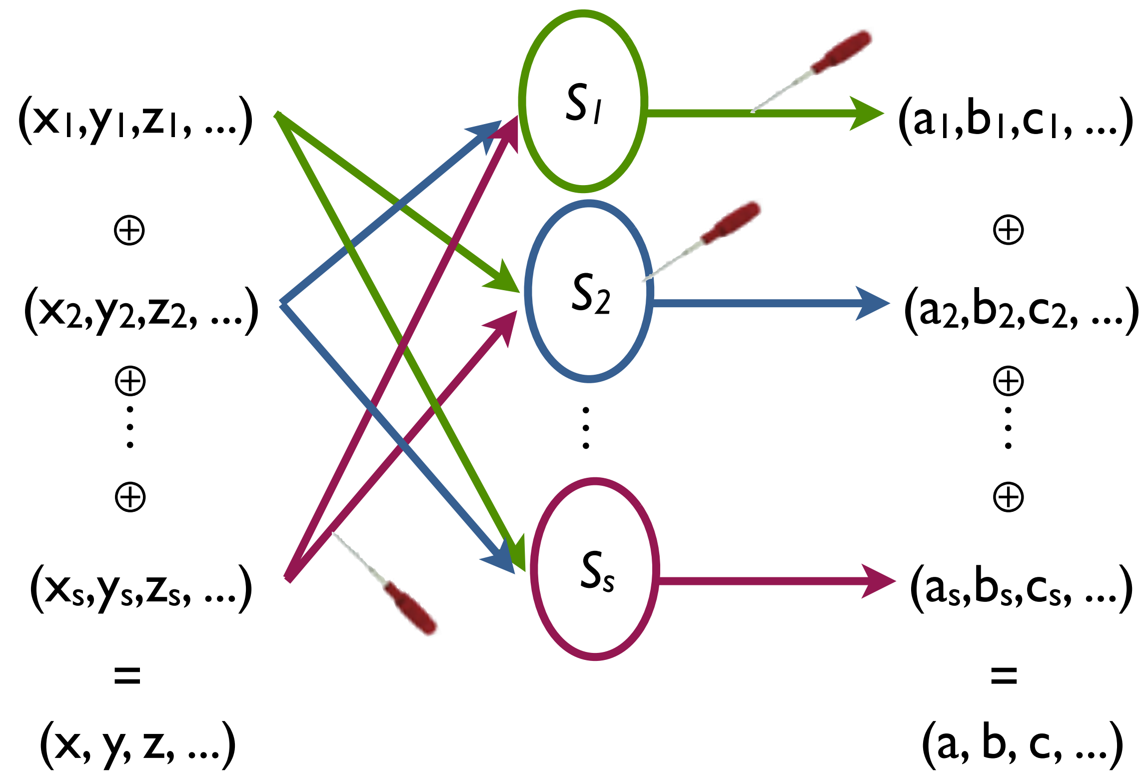


Uniform input masking

Correctness

dth-order non-completeness: Combination of up to *d* functions is independent of at least one input share.

Higher-Order Threshold Implementations



Uniform input masking

Correctness

d^{th}-order non-completeness: Combination of up to *d* functions is independent of at least one input share.

How many shares are necessary?

Higher-Order Threshold Implementations

Higher-Order Threshold Implementations

Linear functions

Higher-Order Threshold Implementations

Linear functions

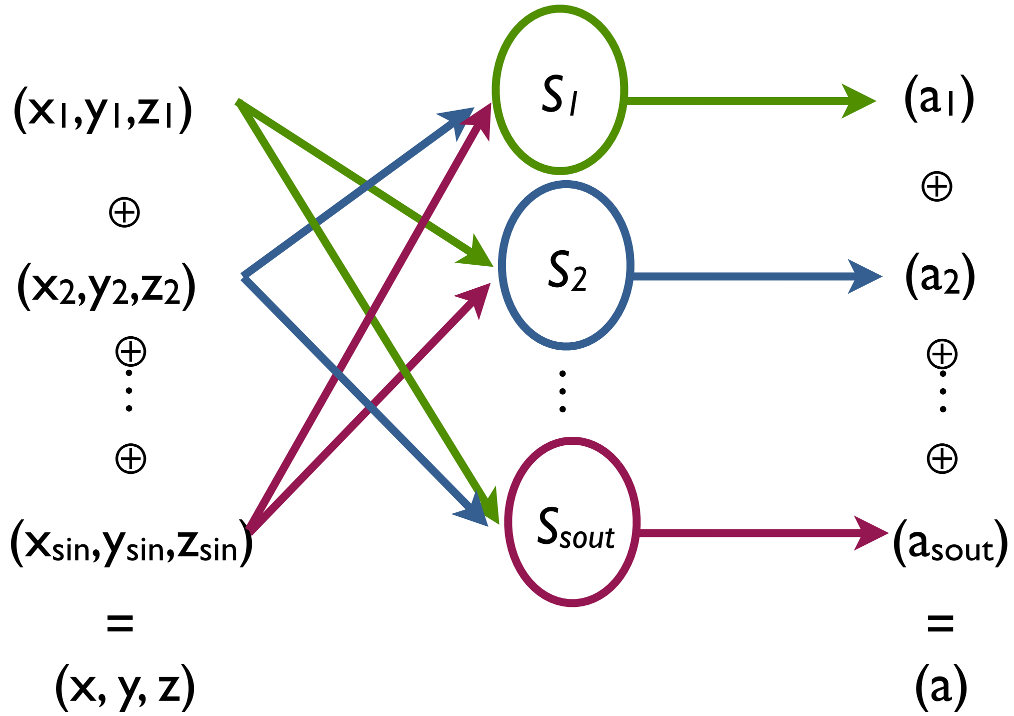
- $S(x) = S(x_1) \oplus S(x_2) \oplus \ldots \oplus S(x_s)$

Higher-Order Threshold Implementations

Linear functions

- $S(x) = S(x_1) \oplus S(x_2) \oplus \ldots \oplus S(x_s)$
- $\#shares\ (s) > d$

Higher-Order Threshold Implementations



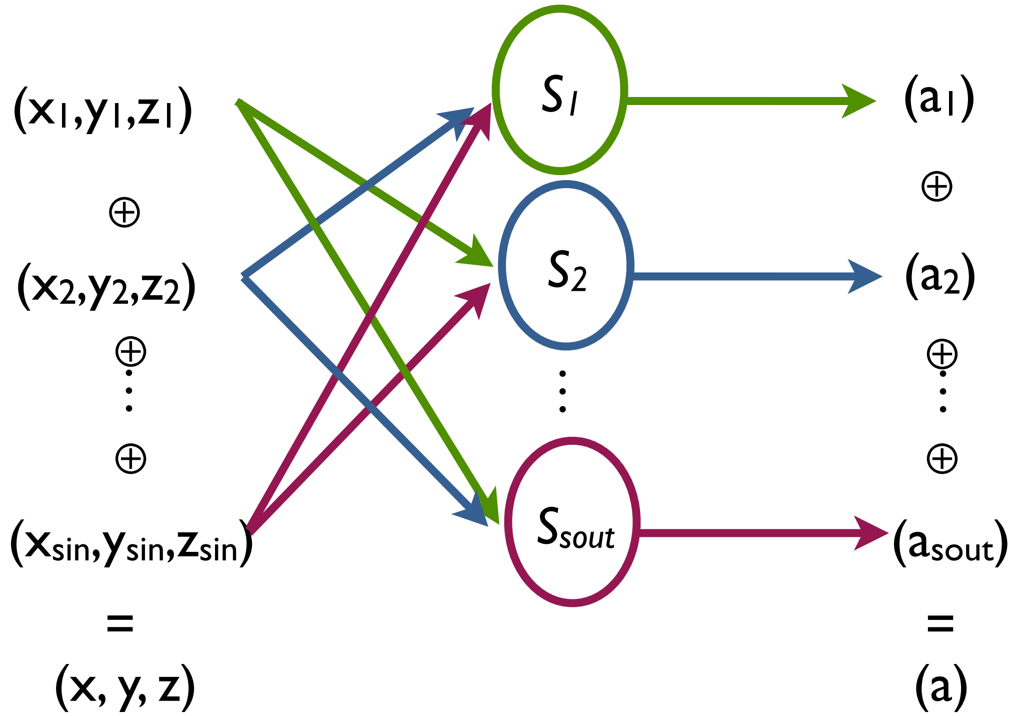
Linear functions

- $S(x) = S(x_1) \oplus S(x_2) \oplus \ldots \oplus S(x_s)$
- *#shares* $(s) > d$

Nonlinear functions (a = $S(x,y,z) = xy+z$)

- More challenging

Higher-Order Threshold Implementations



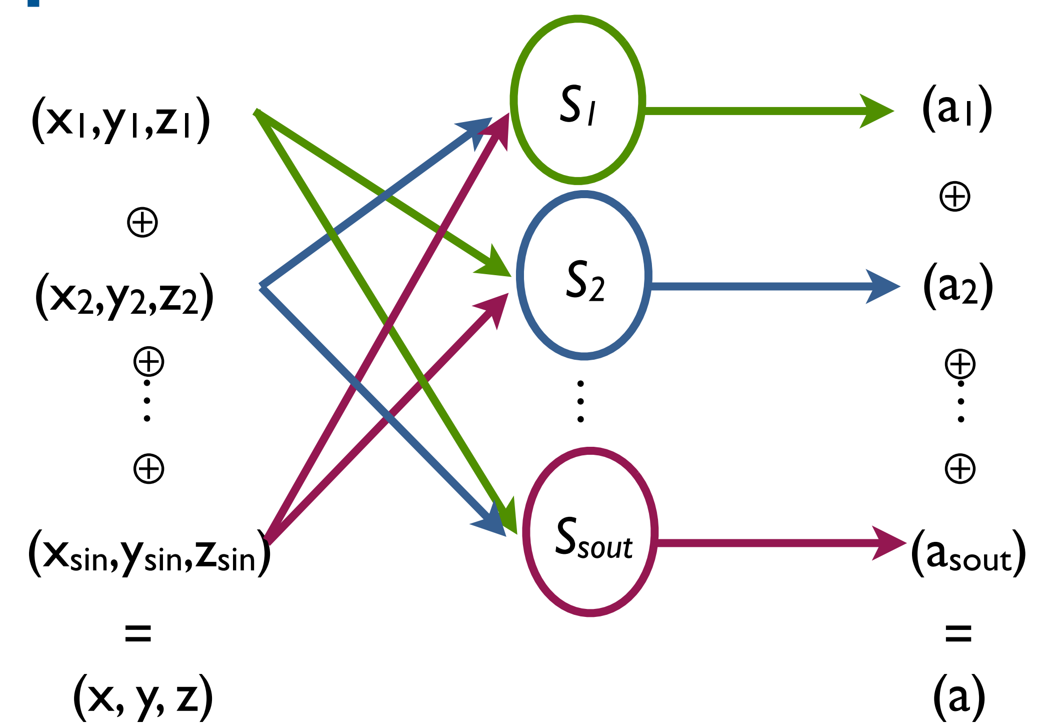
Linear functions

- $S(x) = S(x_1) \oplus S(x_2) \oplus \ldots \oplus S(x_s)$
- *#shares (s) > d*

Nonlinear functions (a = $S(x,y,z) = xy+z$)

- More challenging
- $s_{in} \geq td+1$ and $s_{out} \geq \binom{s_{in}}{t}$ *(algebraic degree of S = t)*

Higher-Order Threshold Implementations



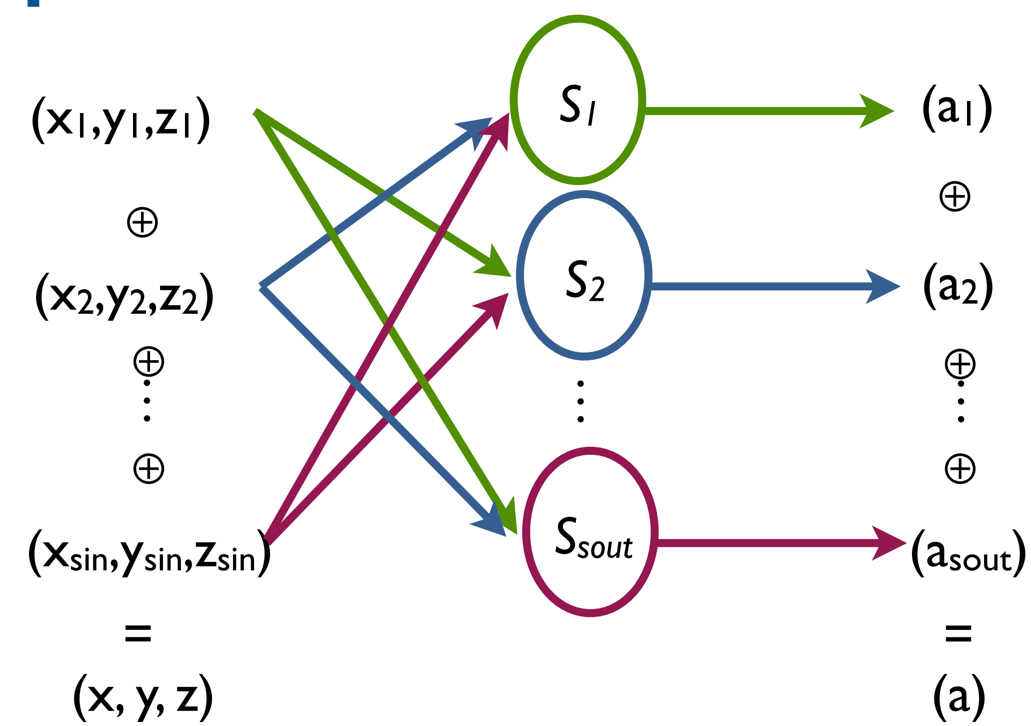
Linear functions

- $S(x) = S(x_1) \oplus S(x_2) \oplus \ldots \oplus S(x_s)$
- *#shares* $(s) > d$

Nonlinear functions (a = $S(x,y,z) = xy+z$)

- More challenging
- $s_{in} \geq td+1$ and $s_{out} \geq \binom{s_{in}}{t}$ *(algebraic degree of* $S = t$*)*
- First-order $s_{in} \geq 3$ input $s_{out} \geq 3$ output shares

Higher-Order Threshold Implementations



Linear functions

- $S(x) = S(x_1) \oplus S(x_2) \oplus \ldots \oplus S(x_s)$
- *#shares (s) > d*

Nonlinear functions (a = $S(x,y,z) = xy+z$)

- More challenging
- $s_{in} \geq td+1$ and $s_{out} \geq \binom{s_{in}}{t}$ *(algebraic degree of S = t)*
- First-order $s_{in} \geq 3$ input $s_{out} \geq 3$ output shares
- Second-order $s_{in} \geq 5$ input and $s_{out} \geq 10$ output shares

Higher-Order Threshold Implementations

Higher-Order Threshold Implementations

Two issues to solve in the system:

Higher-Order Threshold Implementations

Two issues to solve in the system:

1. Increase of the number of shares when d>1

Higher-Order Threshold Implementations

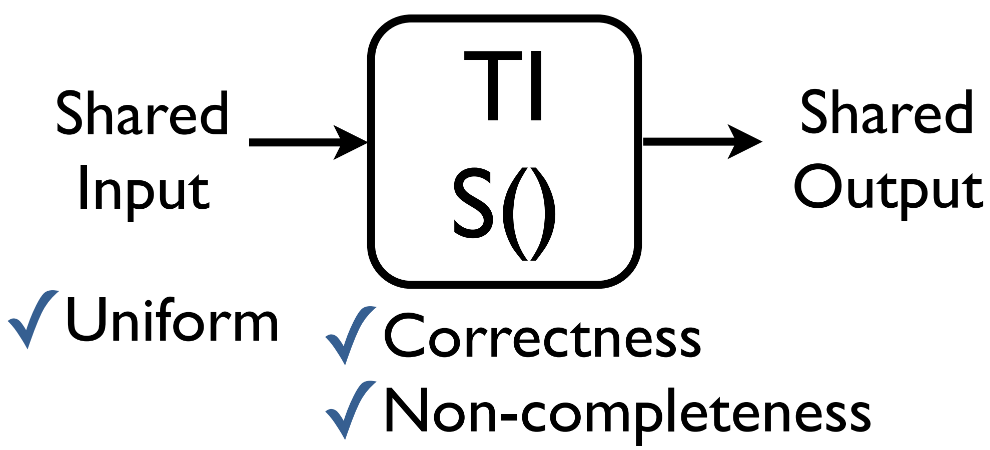
Two issues to solve in the system:

1. Increase of the number of shares when d>1
2. Input to the next nonlinear function must be uniform

Higher-Order Threshold Implementations

Two issues to solve in the system:

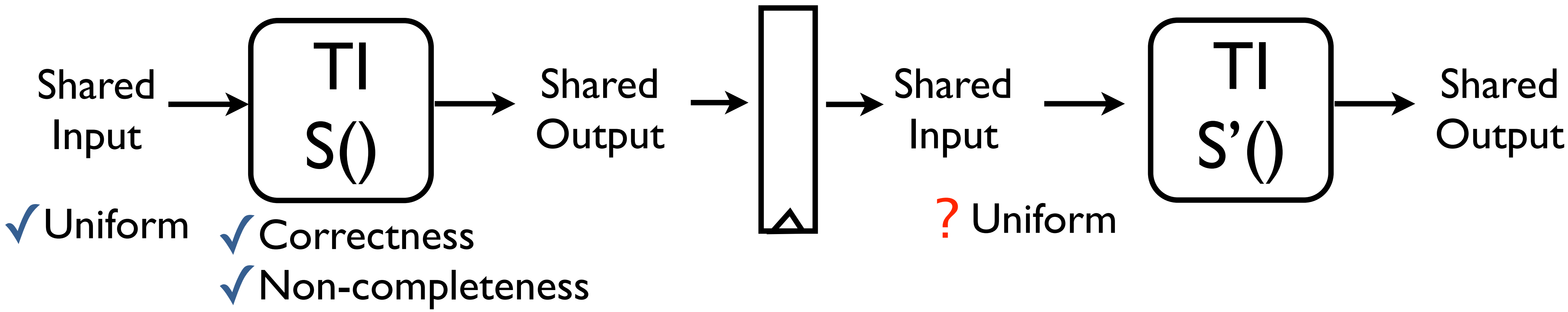
1. Increase of the number of shares when d>1
2. Input to the next nonlinear function must be uniform



Higher-Order Threshold Implementations

Two issues to solve in the system:

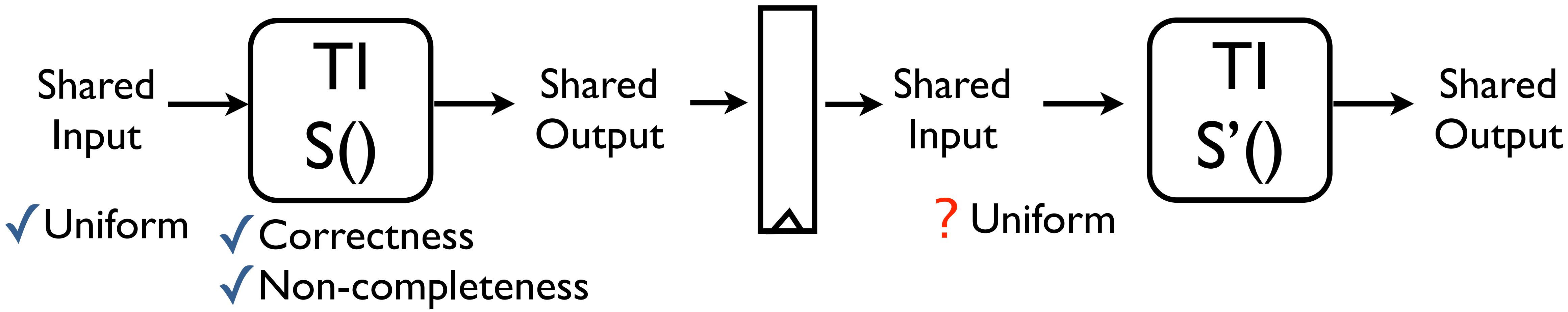
1. Increase of the number of shares when d>1
2. Input to the next nonlinear function must be uniform



Higher-Order Threshold Implementations

Two issues to solve in the system:

1. Increase of the number of shares when d>1
2. Input to the next nonlinear function must be uniform

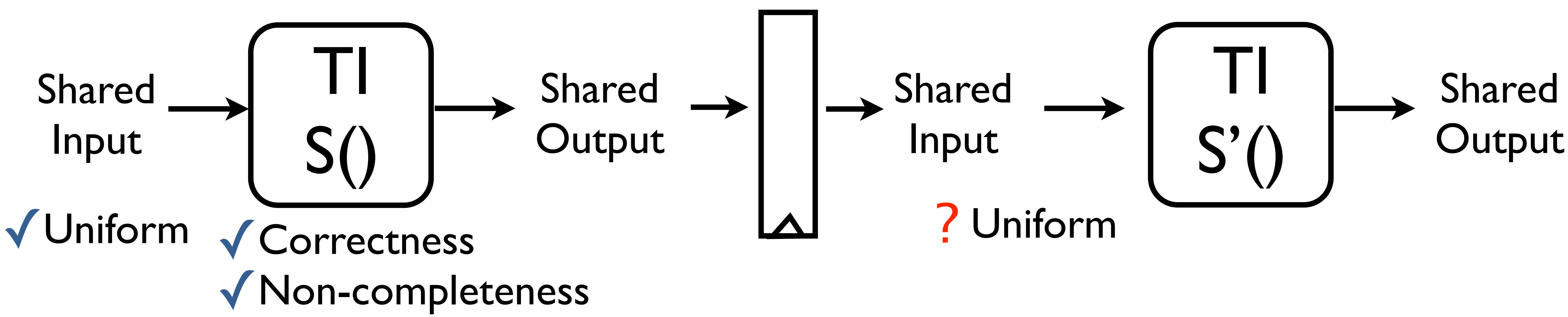


One solution: XOR some of the output shares

Higher-Order Threshold Implementations

Two issues to solve in the system:

1. Increase of the number of shares when d>1
2. Input to the next nonlinear function must be uniform



One solution: XOR some of the output shares

In our paper:

- Uniform HO-TI of an AND/XOR gate
- Uniform second-order TI of quadratic 4-bit permutations

Higher-Order Threshold Implementations

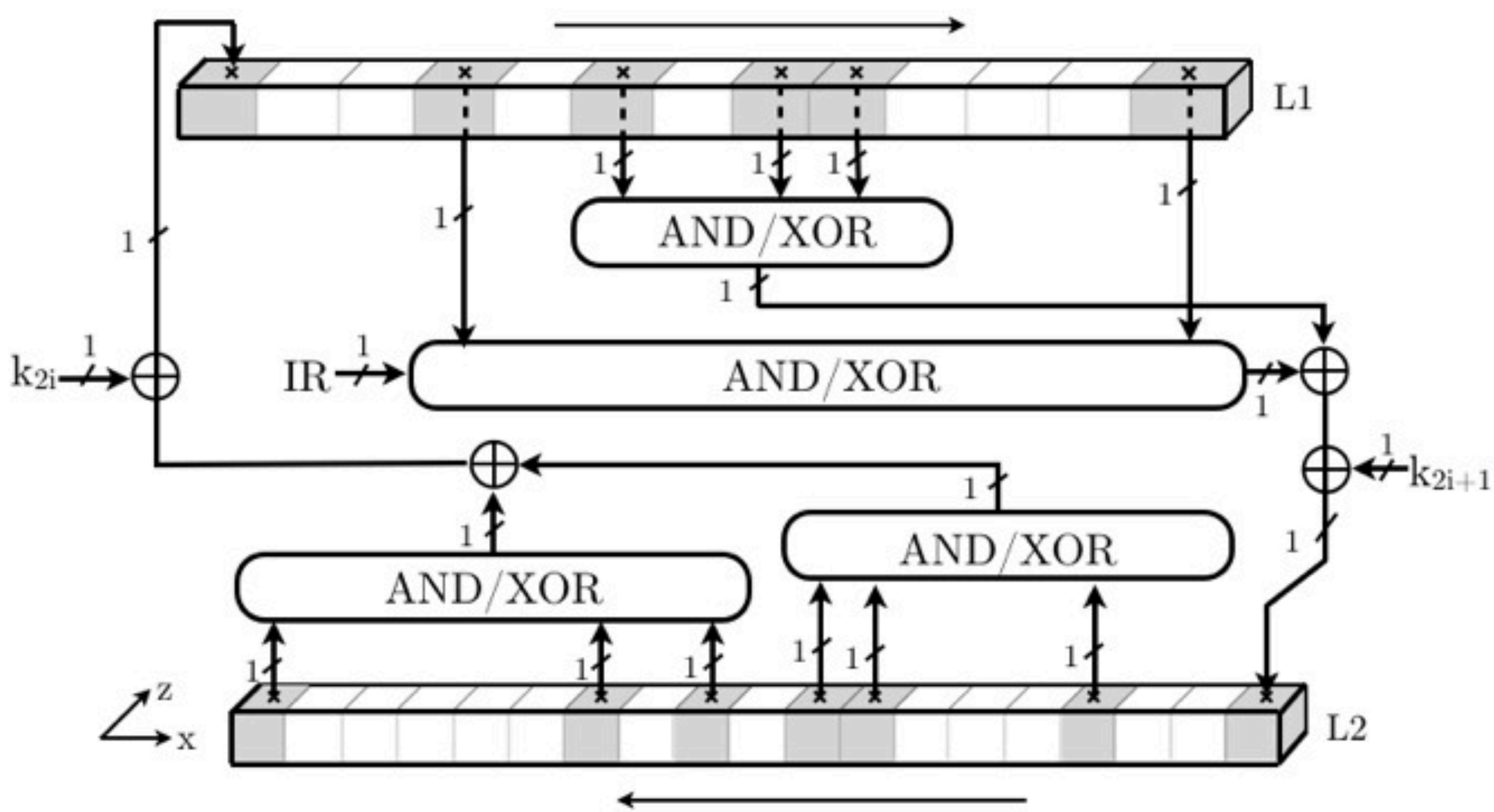
Application to a cryptographic algorithm
&
Testing

Higher-Order Threshold Implementations

Second-order TI of KATAN-32
&
Leakage Detection Tests on SASEBO-G

KATAN-32

- 254-round block cipher
- 32-bit plain/cipher-text and 80-bit key
- Round keys are generated by an LFSR



Linear: $s \geq d+1$
Nonlinear: $s_{in} \geq td+1$ and $s_{out} \geq \binom{s_{in}}{t}$

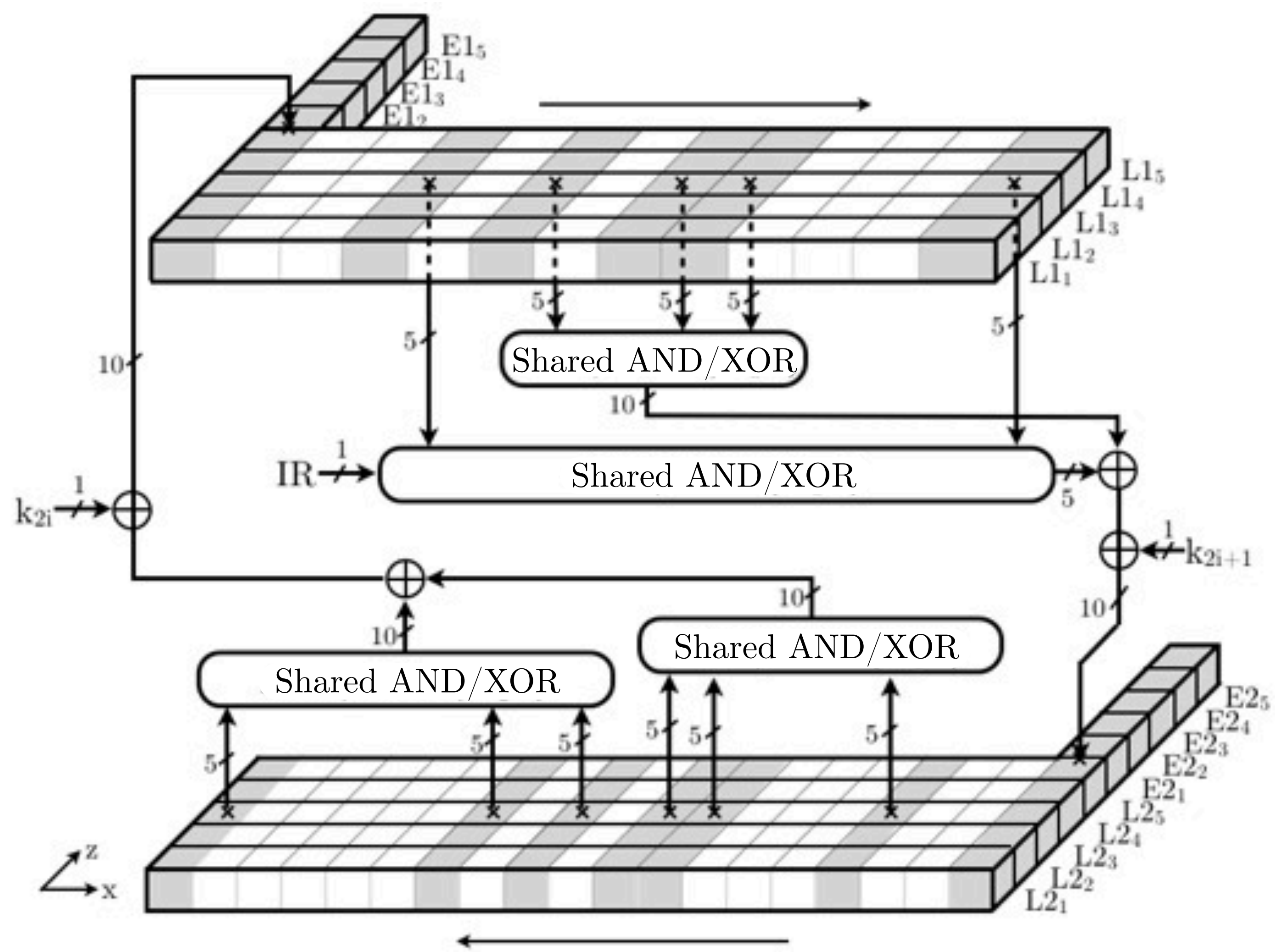
HO-TI of KATAN-32

	# of shares		
	Linear	Nonlinear	
		s_{in}	s_{out}
Unprotected	1	1	1
First-Order TI	3	3	3
Second-Order TI	5	5	10
Third-Order TI	7	7	21

Linear: $s \geq d+1$
Nonlinear: $s_{in} \geq td+1$ and $s_{out} \geq \binom{s_{in}}{t}$

HO-TI of KATAN-32

	# of shares		
	Linear	Nonlinear	
		s_{in}	s_{out}
Unprotected	1	1	1
First-Order TI	3	3	3
Second-Order TI	5	5	10
Third-Order TI	7	7	21

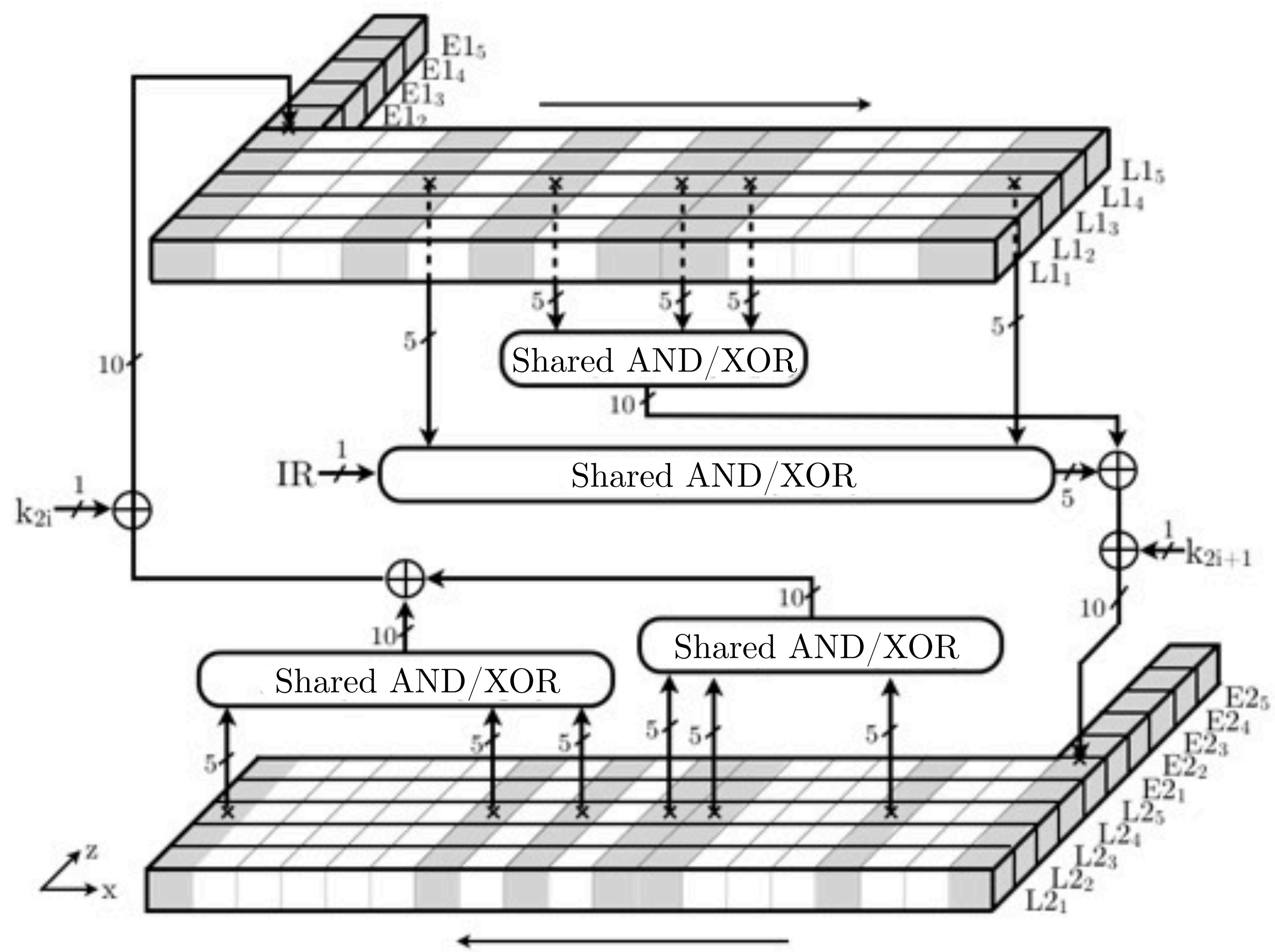


Linear: $s \geq d+1$

Nonlinear: $s_{in} \geq td+1$ and $s_{out} \geq \binom{s_{in}}{t}$

HO-TI of KATAN-32

	# of shares			Area (GE) Faraday Standard Cell Library FSA0A C Generic Core
	Linear	Nonlinear		
		s_{in}	s_{out}	
Unprotected	1	1	1	1002
First-Order TI	3	3	3	1720
Second-Order TI	5	5	10	2556
Third-Order TI	7	7	21	3539



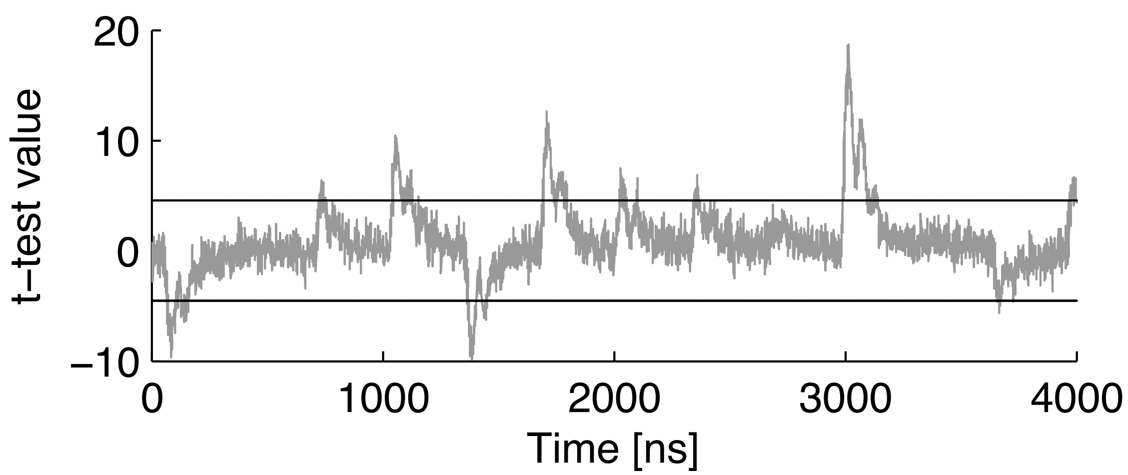
HO-TI of KATAN-32 - Analysis

Fix vs. random leakage detection test
RNG is OFF to test the setup

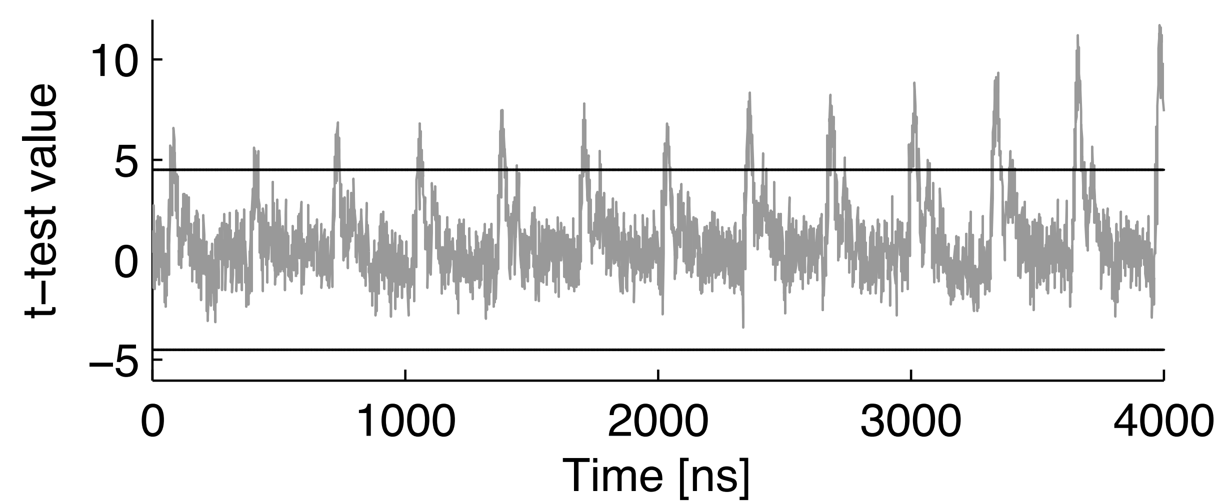
HO-TI of KATAN-32 - Analysis

Fix vs. random leakage detection test
RNG is OFF to test the setup

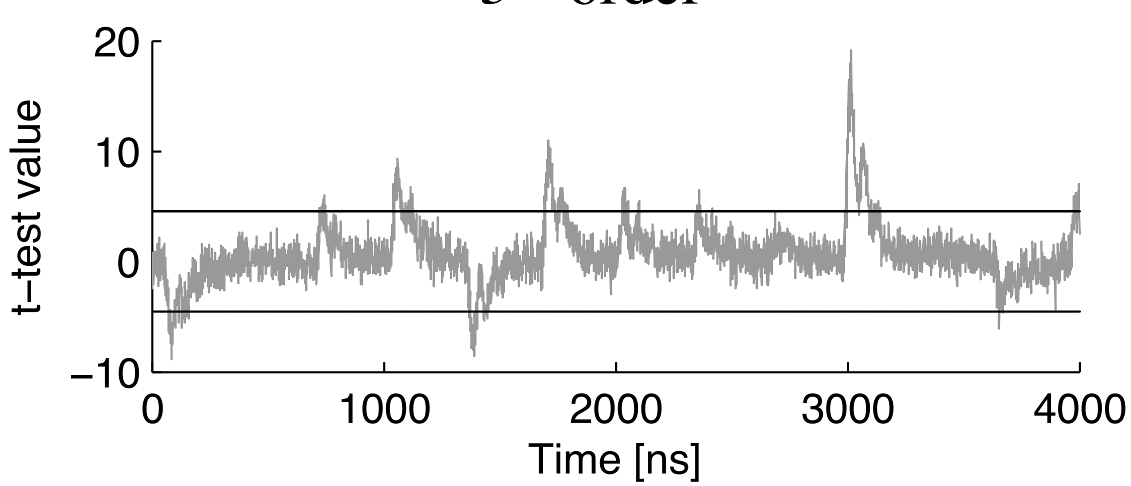
1st-order



2nd-order



3rd-order



1000 traces

HO-TI of KATAN-32 - Analysis

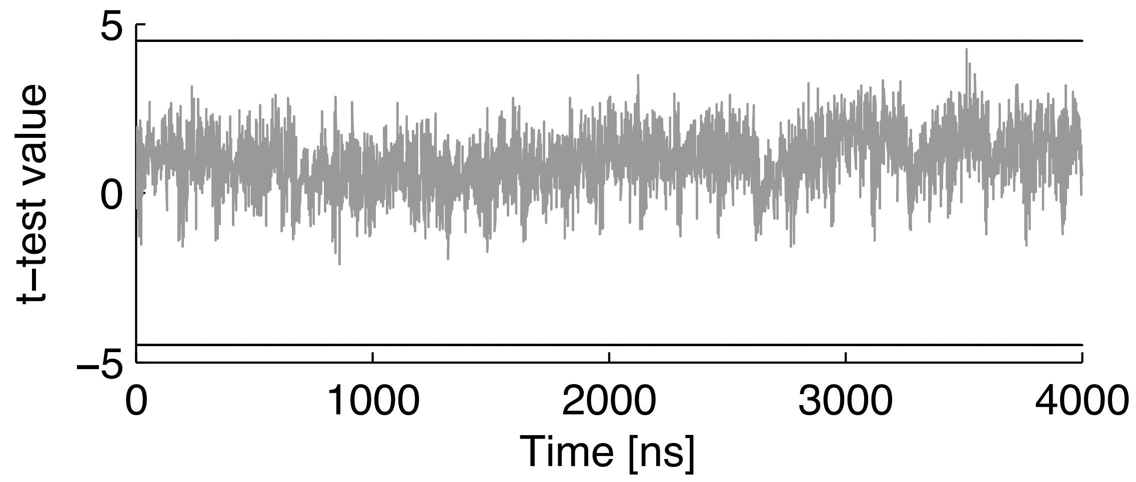
Fix vs. random leakage detection test
RNG is ON

HO-TI of KATAN-32 - Analysis

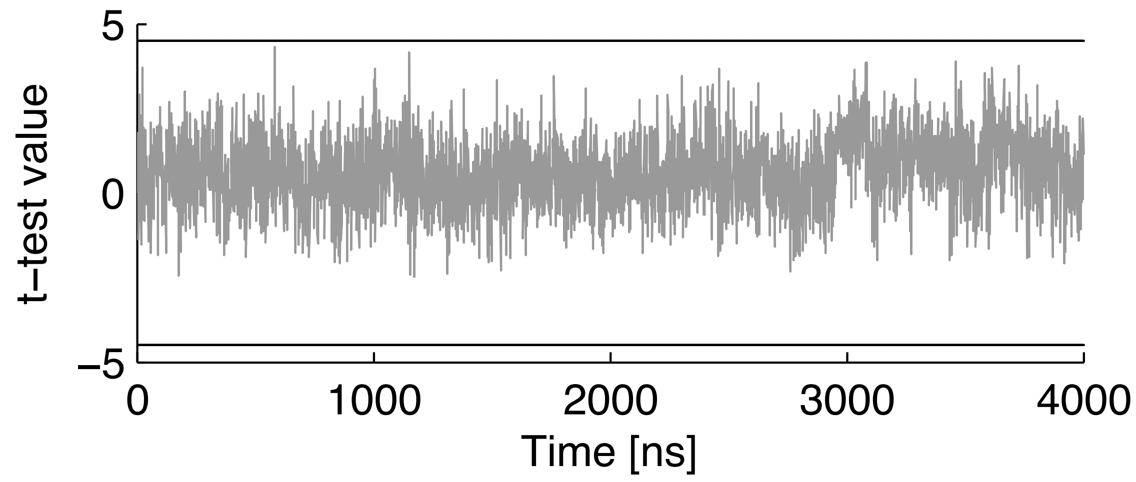
Fix vs. random leakage detection test

RNG is ON

1st-order



2nd-order

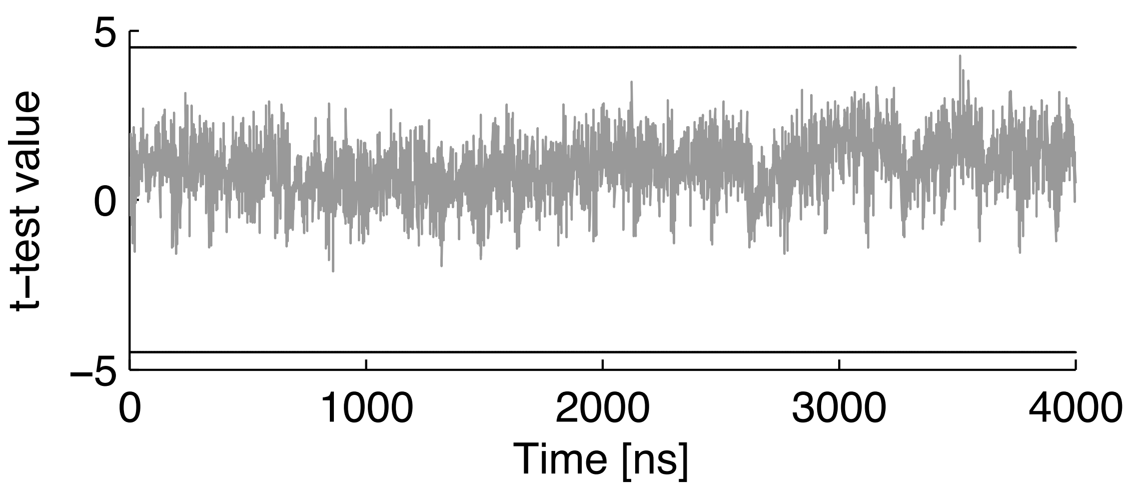


HO-TI of KATAN-32 - Analysis

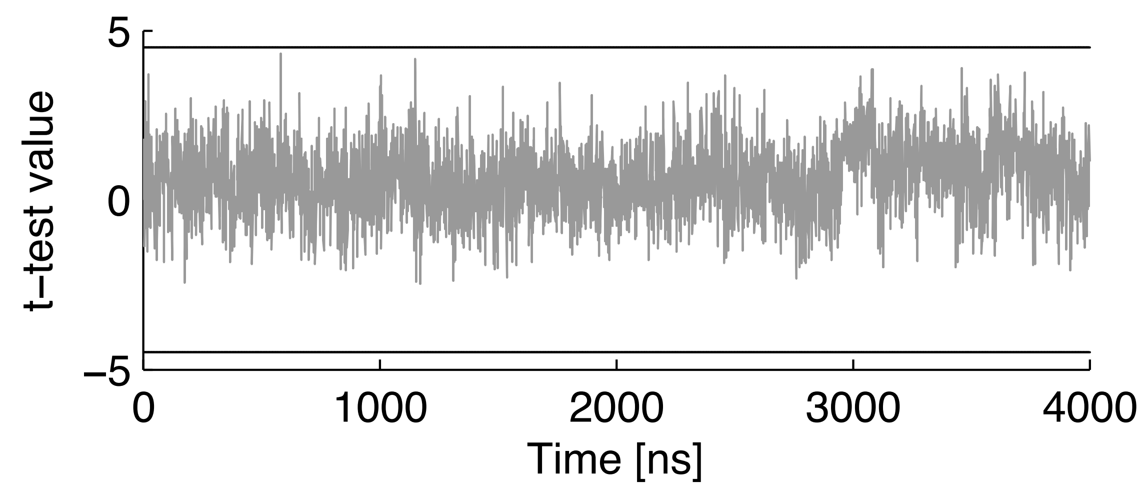
Fix vs. random leakage detection test

RNG is ON

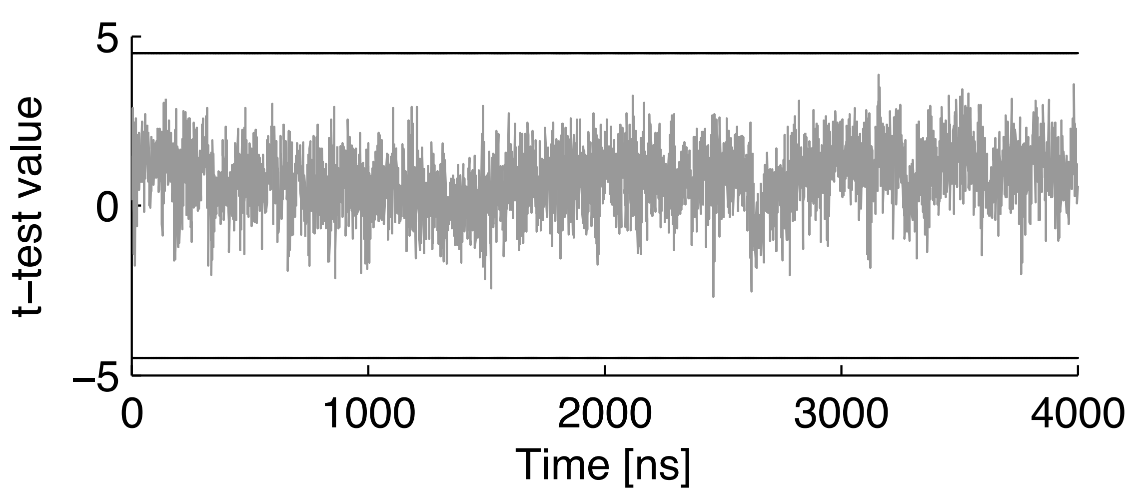
1st-order



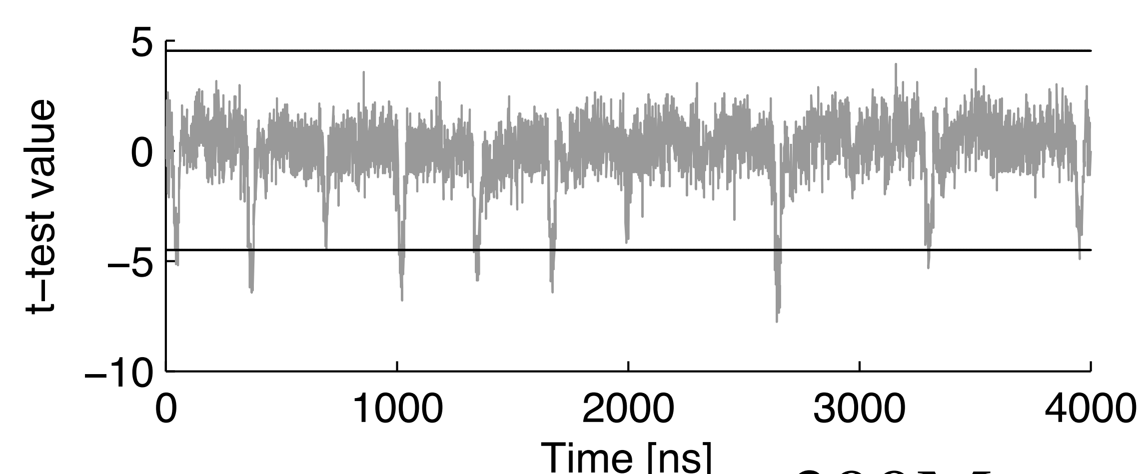
2nd-order



3rd-order



5th-order



300M traces

Conclusion

- Countermeasure against HO-DPA
- Efficient TIs of KATAN-32
- Confirmed the claimed security using leakage detection tests
- Methods for second-order TI of quadratic 4-bit permutations

Thank You!