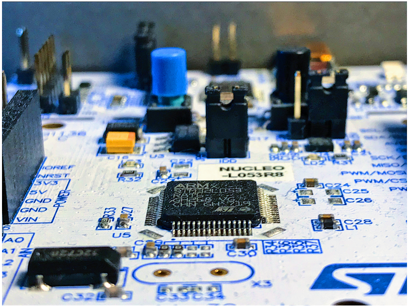
Extending Glitch-Free Multiparty Protocols to Resist Fault Injection Attacks

Okan Seker, Abraham Fernandez-Rubio, Thomas Eisenbarth and Rainer Steinwandt

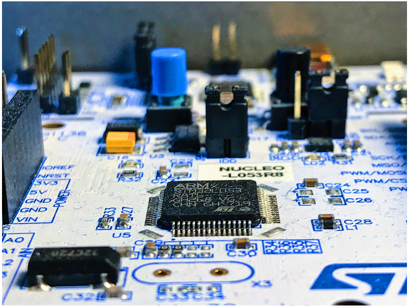
September 9, 2018
CHES 2018 - Amsterdam

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR IT SECURITY

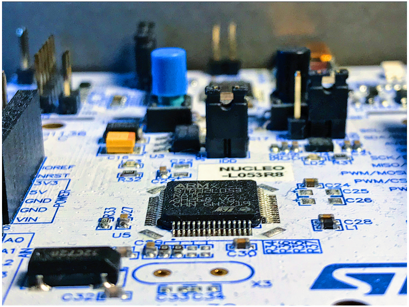
Physical Attacks on Embedded Systems



Physical Attacks on Embedded Systems

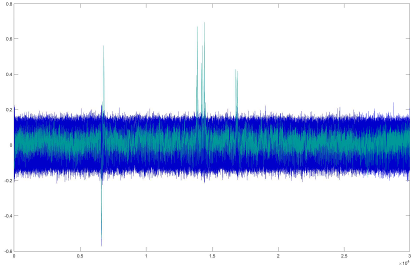


Physical Attacks on Embedded Systems



- Side Channel Attacks,
- Fault Injection,
- Probing,
- Glitches,. . .

Physical Attacks on Embedded Devices



- Side Channel Attacks,
- Fault Injection,
- Probing,
- Glitches,. . .

How to Protect Implementations?

Side Channel Contermeasures:

- Private Circuits
- Boolean & Polynomial Masking
- Threshold Implementations

Fault Injection Countermeasures

- Redundancy in time and space
- Error detection
- Infective computation

How to Protect Implementations?

Side Channel Contermeasures:

- Private Circuits
- Boolean & Polynomial Masking
- Threshold Implementations

Fault Injection Countermeasures

- Redundancy in time and space
- Error detection
- Infective computation

Combined Countermeasures

- Private Circuits II [IPSW06],
- ParTI [SMG16],
- CAPA [RMB^{+}17].

Table of Contents

Shamir's Secret Sharing [Sha79]

1. $F(x) = f_0 + f_1 x + \ldots + f_d x^d$,
2. Evaluating $F(x)$ for n nonzero public points $(\alpha_0, \ldots, \alpha_{n-1})$,
3. *Secret shares* of f_0 is : $\mathcal{F} = (F(\alpha_0), \ldots, F(\alpha_{n-1}))$ or $\mathcal{F} = (F_0, \ldots, F_{n-1})$.

Shamir's Secret Sharing [Sha79]

1. $F(x) = f_0 + f_1 x + \ldots + f_d x^d$,
2. Evaluating $F(x)$ for n nonzero public points $(\alpha_0, \ldots, \alpha_{n-1})$,
3. *Secret shares* of f_0 is : $\mathcal{F} = (F(\alpha_0), \ldots, F(\alpha_{n-1}))$ or $\mathcal{F} = (F_0, \ldots, F_{n-1})$.

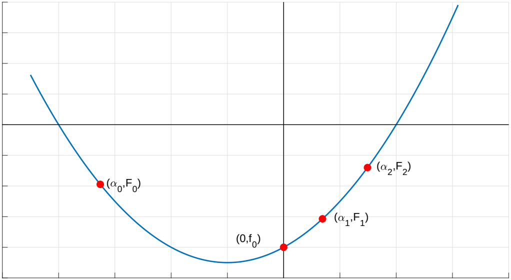
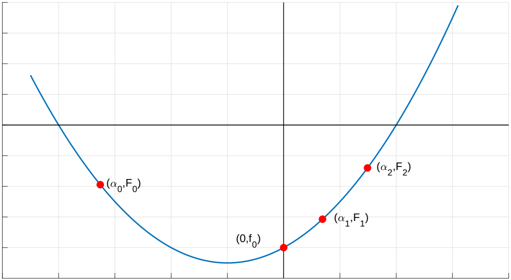


Figure: Shamir's Secret Sharing.

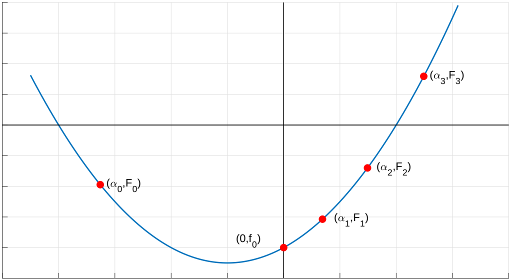
The Secret Reconstruction:

$$F(x) = f_0 + f_1 x + f_2 x^2 \iff \{F_0, F_1, F_2\}$$



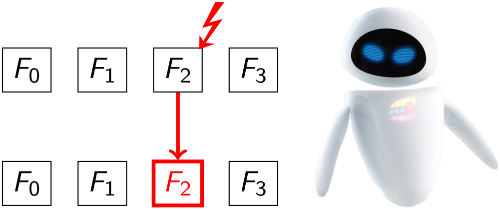
The Secret Reconstruction:

$$F(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 \Longleftrightarrow \{F_0, F_1, F_2, F_3\} \text{ s.t. } f_3 = 0.$$



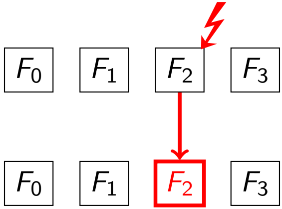
The Secret Reconstruction:

$$\{F_0, F_1, F_2, F_3\} \Longrightarrow f_3 = 0.$$



The Secret Reconstruction:

$$\{F_0, F_1, F_2, F_3\} \Longrightarrow f_3 = 0.$$

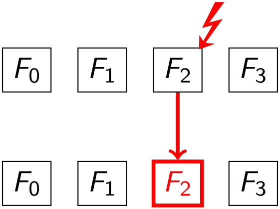


Error Detection:

- The Effect of of FI: $\{F_0, F_1, F_2, F_3\} \Longrightarrow f_3 \neq 0.$

The Secret Reconstruction:

$$\{F_0, F_1, F_2, F_3\} \Longrightarrow f_3 = 0.$$



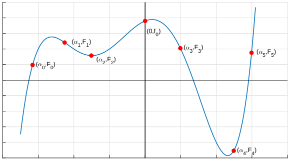
Error Detection:

- The Effect of of FI: $\{F_0, F_1, F_2, F_3\} \Longrightarrow f_3 \neq 0$.
- $\{F_0, \ldots, F_{n-1}\} \Longrightarrow f_{d+1} = \ldots = f_{n-1} = 0$.
- *Error detection terms*: $f_{d+1}, f_{d+2}, \ldots, f_{n-1}$.

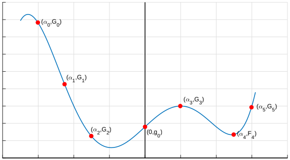
SMC Operations

Secret States:

Shares of f_0 as $(F_i)_{0\leq i<n}$ and shares of g_0 as $(G_i)_{0\leq i<n}$.



$\oplus$

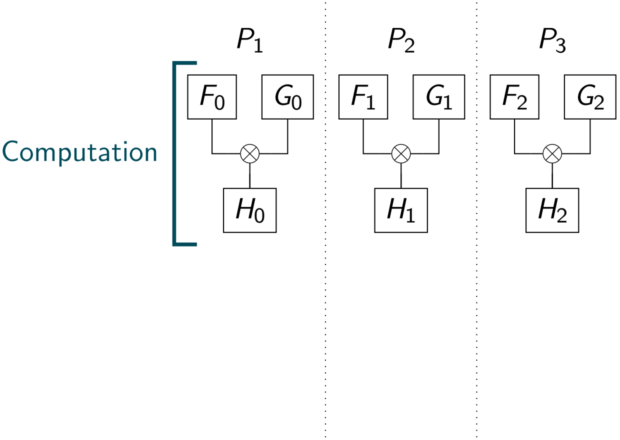


Addition of two Shares:

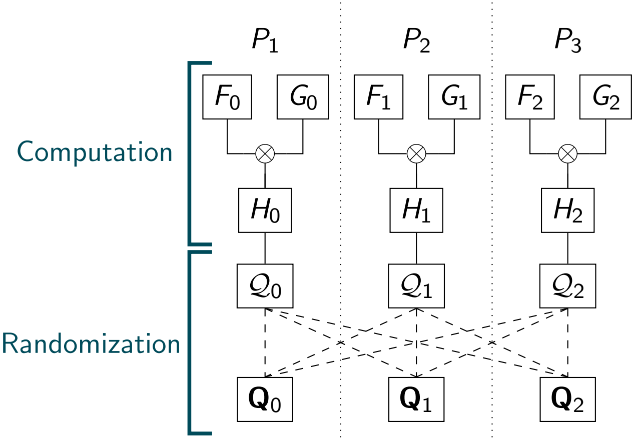
$$[F_i \oplus G_i]_{0\leq i<n}$$

- Affine transformation of a secret $\mathrm{L}(f_0)$.
- Efficient squaring operation $f_0^{2^k}$.

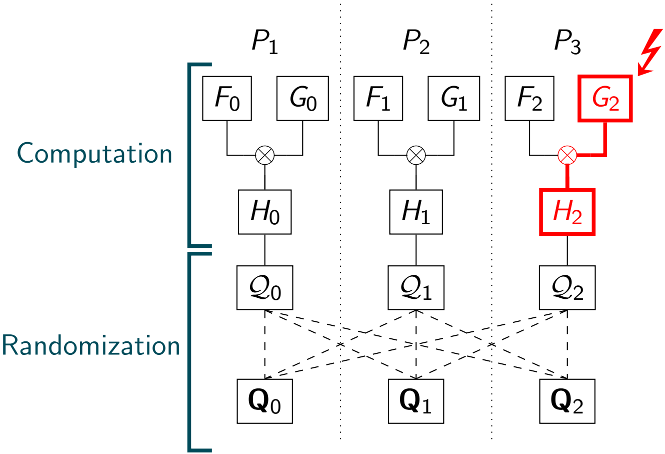
Multiplication of Two Secrets [GRR98]



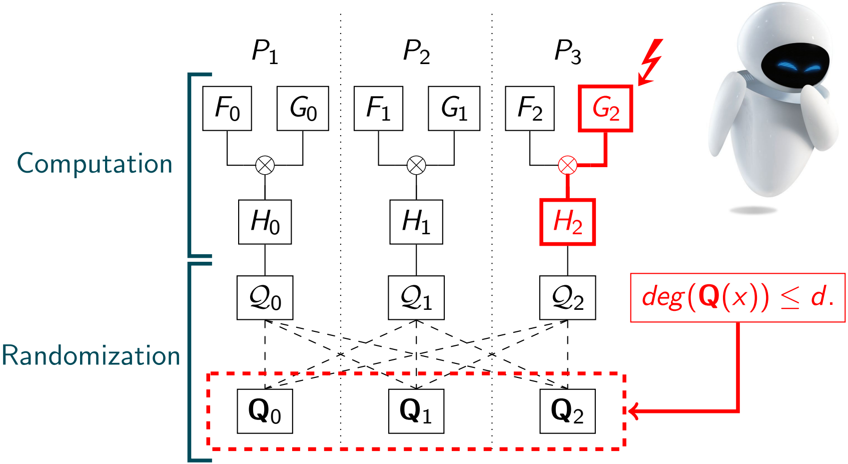
Multiplication of Two Secrets [GRR98]



Multiplication of Two Secrets Under FA



Multiplication of Two Secrets Under FA



How to Protect Implementations?

How to Protect Implementations?

- Error Detection Only:

$$deg(\texttt{Output}) > d.$$

How to Protect Implementations?

- Error Detection Only:

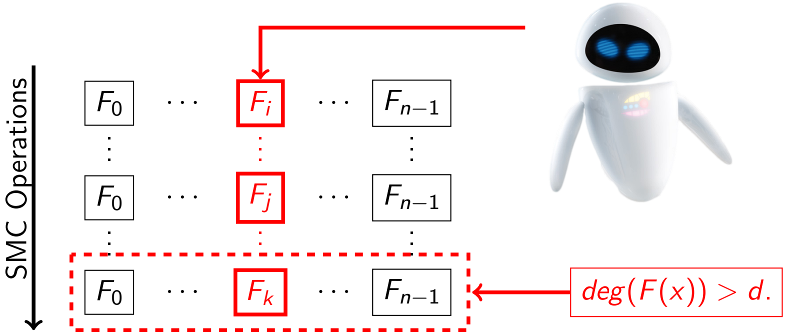
$$deg(\texttt{Output}) > d.$$

- Fault Detection Without Leaking Information:

$$\{F_0, \ldots, F_{n-1}\}$$

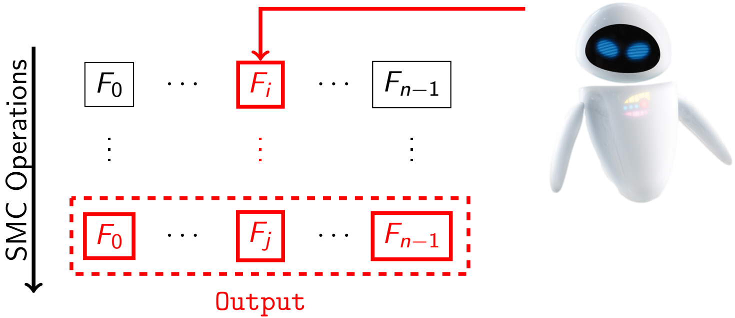
How to Protect Implementations?

- Error-Preserving Computation.



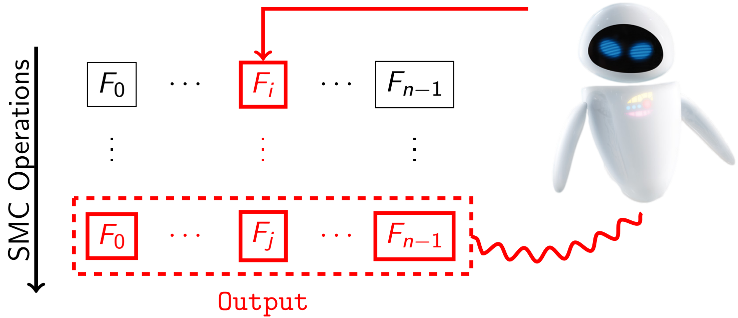
How to Protect Implementations?

- Infective Computation.



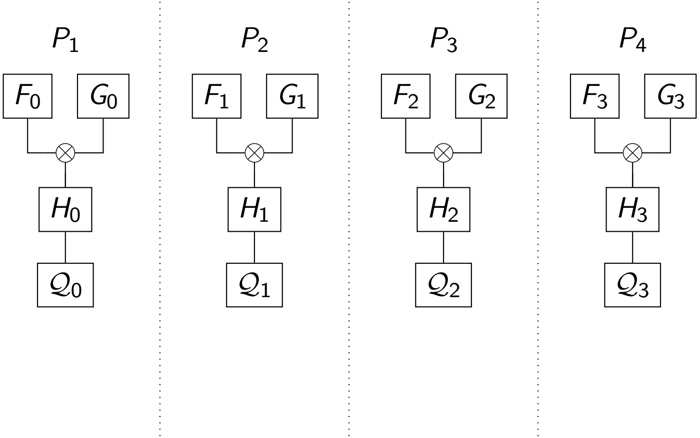
How to Protect Implementations?

- Infective Computation.



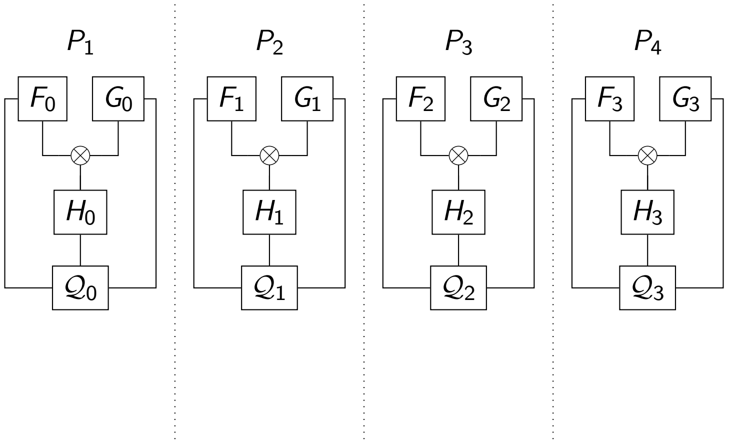
Secret States with $n > 2d + \varepsilon$

- Shares of f_0 as $(F_i)_{0 \leq i < n}$ and shares of g_0 as $(G_i)_{0 \leq i < n}$.
- *Error detection terms*: $(f_i, g_i)_{d < i < n}$ & $(h_j)_{2d < j < n}$.



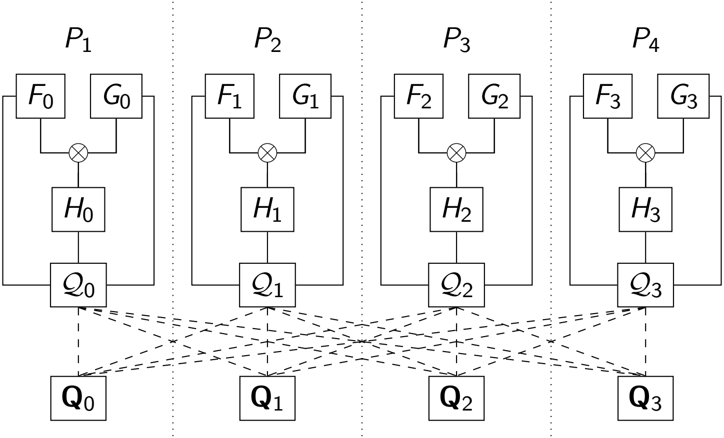
Secret States with $n > 2d + \varepsilon$

- Shares of f_0 as $(F_i)_{0\leq i<n}$ and shares of g_0 as $(G_i)_{0\leq i<n}$.
- *Error detection terms*: $(f_i, g_i)_{d<i<n}$ & $(h_j)_{2d<j<n}$.



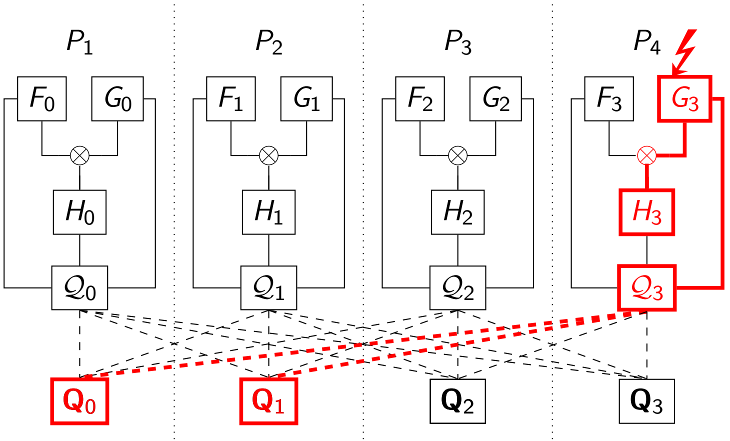
Secret States with $n > 2d + \varepsilon$

- Shares of f_0 as $(F_i)_{0 \leq i < n}$ and shares of g_0 as $(G_i)_{0 \leq i < n}$.
- *Error detection terms*: $(f_i, g_i)_{d < i < n}$ & $(h_j)_{2d < j < n}$.



Secret States with $n > 2d + \varepsilon$

- Shares of f_0 as $(F_i)_{0 \leq i < n}$ and shares of g_0 as $(G_i)_{0 \leq i < n}$.
- *Error detection terms*: $(f_i, g_i)_{d < i < n}$ & $(h_j)_{2d < j < n}$.



Propogation of Error Detection Terms

1. The update of $\mathcal{Q}_i$ and the utilization of *error detection terms*:
   - $\mathcal{Q}_i(\alpha_j) \leftarrow \mathcal{Q}_i(\alpha_j) \oplus \mathtt{E}_{i,j}$ for $j = 0, \ldots, n-1$.

$\mathsf{E}_{i,j} \leftarrow$ A share of error detection term of H or $F \oplus G$

Propogation of Error Detection Terms

1. The update of $\mathcal{Q}_i$ and the utilization of *error detection terms*:
    - $\mathcal{Q}_i(\alpha_j) \leftarrow \mathcal{Q}_i(\alpha_j) \oplus \mathtt{E}_{i,j}$ for $j = 0, \ldots, n-1$.

$$\mathsf{E}_{i,j} \leftarrow \text{A share of error detection term of } H \text{ or } F \oplus G$$

2. Fault propagation within $\mathbf{Q}_i$.

$$\mathbf{Q}_i \leftarrow \sum_{j=0}^{n-1} \lambda_i^0 \mathcal{Q}_{j,i}$$

Propogation of Error Detection Terms

1. The update of $\mathcal{Q}_i$ and the utilization of *error detection terms*:
   - $\mathcal{Q}_i(\alpha_j) \leftarrow \mathcal{Q}_i(\alpha_j) \oplus \mathtt{E}_{i,j}$ for $j = 0, \ldots, n-1$.

$$\mathrm{E}_{i,j} \leftarrow \text{A share of error detection term of } H \text{ or } F \oplus G$$

2. Fault propagation within $\mathbf{Q}_i$.

$$\mathbf{Q}_i \leftarrow \sum_{j=0}^{n-1} \lambda_i^0 \mathcal{Q}_{j,i} = \begin{cases} \mathbf{Q}_i \oplus h_{n-i-1} & \text{if } 0 \leq i < \varepsilon \\ \mathbf{Q}_i \oplus g_{n-i-1} \oplus f_{n-i-1} & \text{if } \varepsilon \leq i < \varepsilon + d \\ \mathbf{Q}_i & \text{if } \varepsilon + d \leq i < n \end{cases}$$

Security in Probing Model

t-SNI Security [CGPZ16]:

The standard way of proving the security against probing attacks.

Security in Probing Model

t-SNI Security [CGPZ16]:

The standard way of proving the security against probing attacks.

t-SNI$_d^n$ Security:

[t probes & $\mathcal{O}$] should be simulatable by I.

- $\mathcal{O}$ with $t + |\mathcal{O}| \leq d$ and $|I| \leq t$.
- d shares are uniformly distributed.

$\rightarrow$ t probes brings no information to the adversary.

Security in Additive Fault Model

Error Propagation:

$$Propagation_{\varepsilon} := Pr\left[deg(\texttt{Ouput}) > d \mid deg(\texttt{Input}) > d\right].$$

- $Propagation_{\varepsilon}(\texttt{Affine}, \texttt{Sqr}) = 1.$
- $Propagation_{\varepsilon}(\texttt{Add}, \texttt{EPMult}) \approx 1.$

The Cost of an EPMult

Table: Number of operations in Gennaro et al. [GRR98] and EPMult.

	[GRR98]			EPMult			Overhead
	step 1	step 2	step 3	step 1	step 2	step 3	
Mul.	n	n^2d	n^2	n	$n^2d + n(\varepsilon + d)$	n^2	$n(\varepsilon + d)$
Add.	-	n^2d	$(n-1)n$	-	$n^2d + n(\varepsilon + 2d)$	$(n-1)n$	$n(\varepsilon + 2d)$
Rand.	-	nd	-	-	nd	-	-

The Cost of an EPMult

Table: Number of operations in Gennaro et al. [GRR98] and EPMult.

	[GRR98]			EPMult			Overhead
	step 1	step 2	step 3	step 1	step 2	step 3	
Mul.	n	n^2d	n^2	n	$n^2d + n(\varepsilon + d)$	n^2	$n(\varepsilon + d)$
Add.	-	n^2d	$(n-1)n$	-	$n^2d + n(\varepsilon + 2d)$	$(n-1)n$	$n(\varepsilon + 2d)$
Rand.	-	nd	-	-	nd	-	-

Calculation of $E_{i,j}$.

*Exp*254

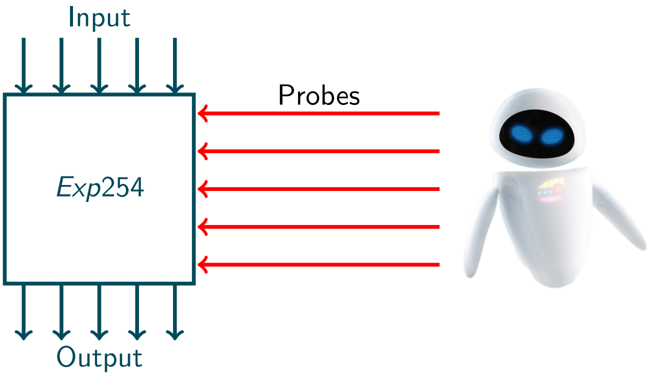
$Sbox(x) = \tau_A \circ Exp254(x)$ where $Exp254(x)$ requires:

- 4 `EPMult`, 3 $\texttt{Sqr}_k$, and 2 `RefreshM`.

*Exp*254

$Sbox(x) = \tau_A \circ Exp254(x)$ where $Exp254(x)$ requires:

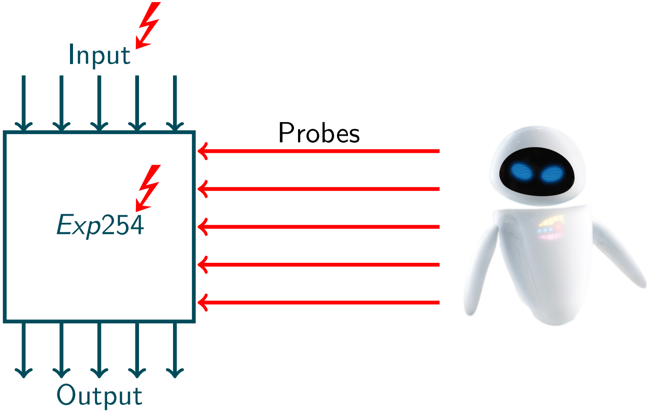
- 4 `EPMult`, 3 $\texttt{Sqr}_k$, and 2 `RefreshM`.



*Exp*254

$Sbox(x) = \tau_A \circ Exp254(x)$ where $Exp254(x)$ requires:

- 4 `EPMult`, 3 $\texttt{Sqr}_k$, and 2 `RefreshM`.



$$Propagation(Exp254) \approx 1 - 2^{-12}$$

The New Multiplication Engine

- Information about the fault remains as a part of the shares.
- The error propagates through the algebraic operations.
- Delay any error detection as late as the final recombination step.

The New Multiplication Engine

- Information about the fault remains as a part of the shares.
- The error propagates through the algebraic operations.
- Delay any error detection as late as the final recombination step.

The Fault Detection and Recombination Gate

- For both fault detection and reconstruction.
- Infective Computation.

Security properties

- ISW probing model.
- t-SNI security of the scheme [RP12].
- Fault detection of our scheme is examined using the notion of *Propagation*.

Security properties

- ISW probing model.
- t-SNI security of the scheme [RP12].
- Fault detection of our scheme is examined using the notion of *Propagation*.

A proof-of-concept C implementation AES-128

- Ultra-low power architecture, the ARM Cortex M0+ core
- full leakage analysis including higher order moments,
- fully constant execution flow with constant memory accesses.

The code has been made publicly available at
https://github.com/vernamlab/Robust-AES.

Thank you!

okan.seker@uni-luebeck.de

its.uni-luebeck.de vernam.wpi.edu

UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR IT SECURITY

WPI

Recombination Operation