Optimized Threshold Implementations: Minimizing the Latency of Secure Cryptographic Accelerators

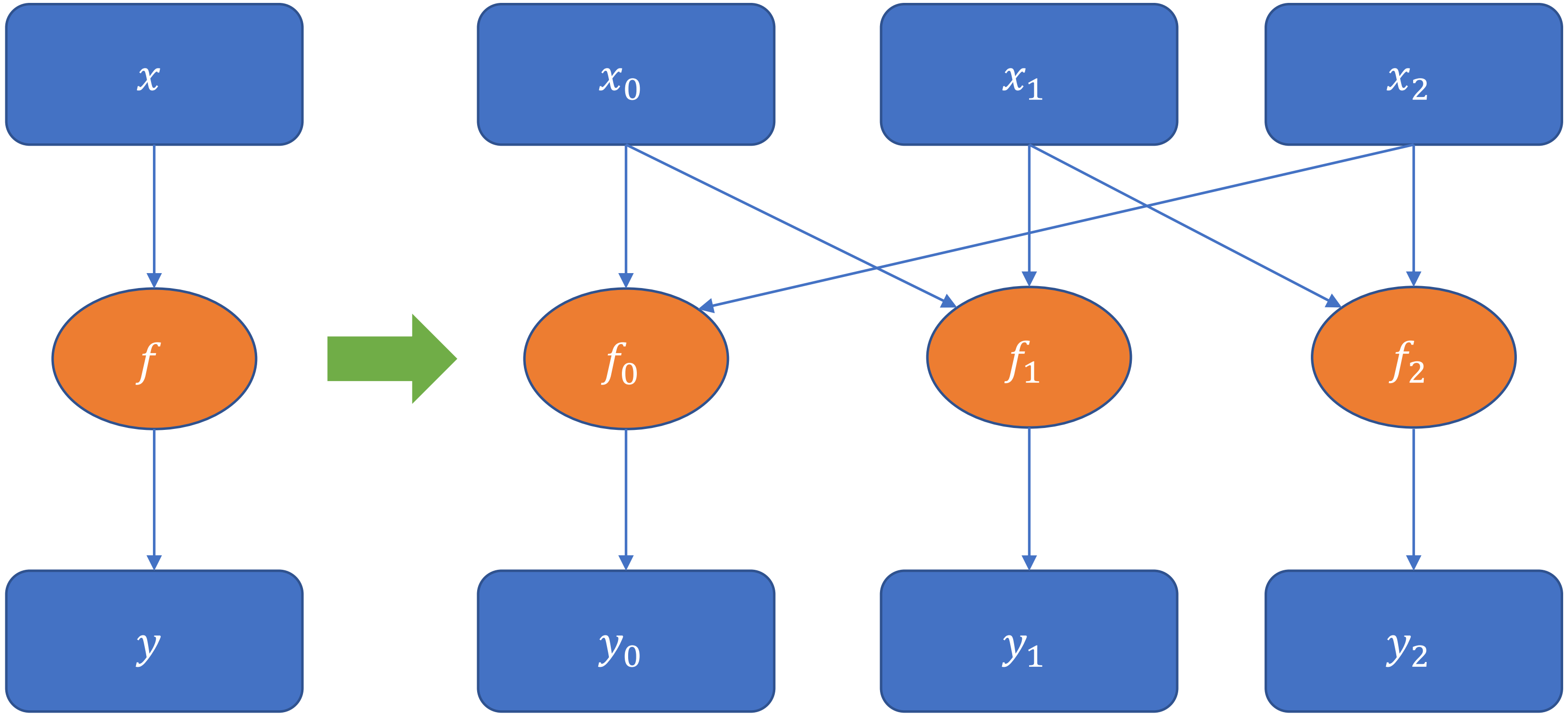
Dušan Božilov, Miroslav Knežević, Ventzislav Nikov

November 13th, 2019

COSIC

Threshold Implementations (TI)

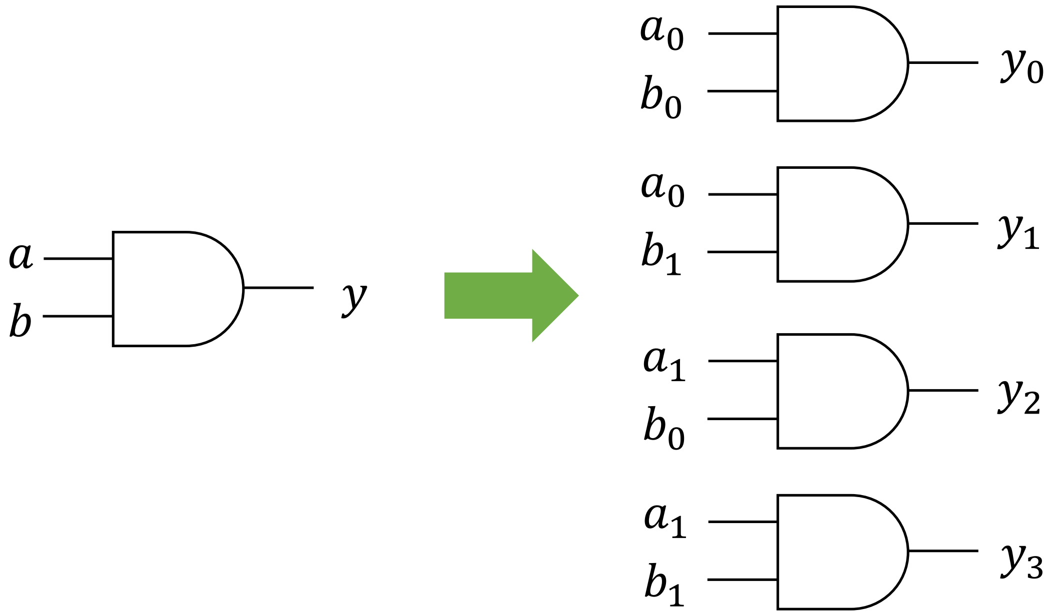
- Boolean masking scheme
- Glitch resistant
- Three key properties
  - Correctness
  - Non-completeness
  - Uniformity
- Two variants
  - td + 1
  - **d + 1**



d+1 TI

- Number of input shares is always d+1, where d is security order
- Number of output shares depends on the algebraic degree t as well, and is lower bound by $(d+1)^t$

$$y = ab$$
$$y = (a_0 + a_1)(b_0 + b_1)$$
$$y_0 = a_0 b_0$$
$$y_1 = a_0 b_1$$
$$y_2 = a_1 b_0$$
$$y_3 = a_1 b_1$$



TI properties

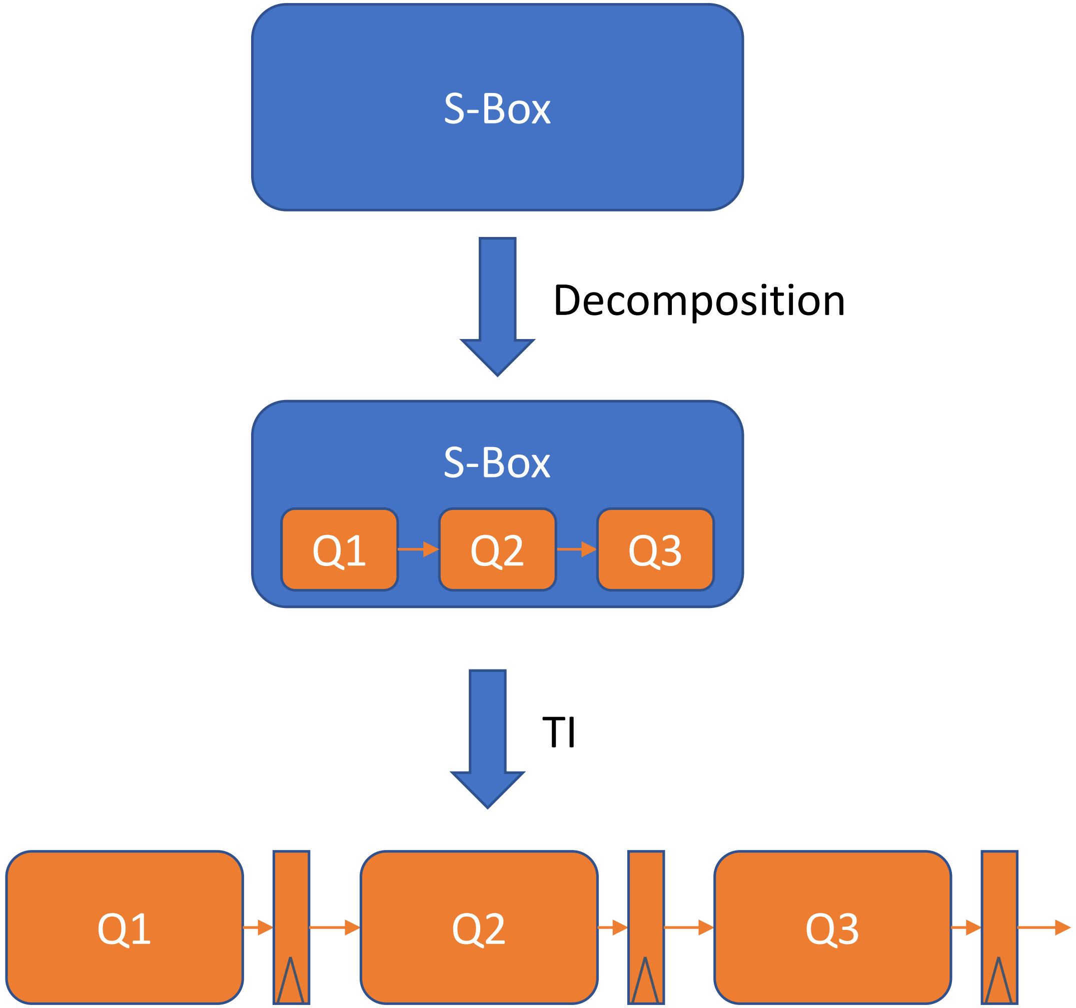
- TI should preserve the functionality of the operation we are trying to protect (correctness)
- Any input share may appear only once in any given output share

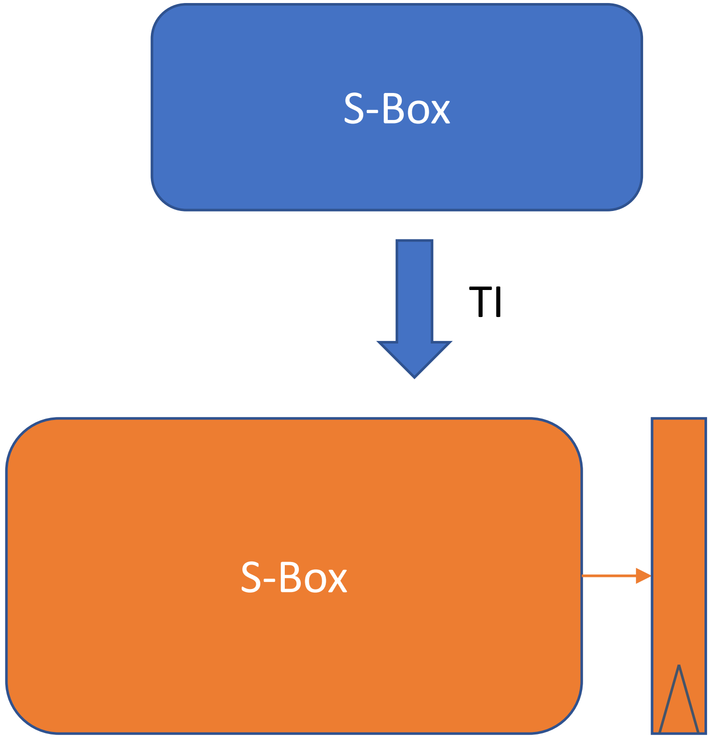
$$y_2 = a_0 b_1 + a_0 \quad ✓$$

$$y_3 = a_0 b_2 + a\_1 c\_0 \quad ✗$$

- Output should preserve the distribution of the input (Uniformity)
  - Mandates registers between non-linear operations
  - Requires randomness injection at the end of every non-linear operation if the result is compressed afterward

S-Box decomposition





From sharing to table

$$y = ab + c$$

$$\begin{array}{c} (a, \quad b, \quad c) \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & * \\ 1 & 0 & * \\ 1 & 1 & 1 \end{pmatrix} \end{array} \qquad \begin{aligned} y_0 &= a_0 b_0 + c_0 \\ y_1 &= a_0 b_1 \\ y_2 &= a_1 b_0 \\ y_3 &= a_1 b_1 + c_1 \end{aligned}$$

- Rows represent one output share and columns represent input variables
- Values represent allowed input share in the output share of a given variable
- Number of variables is the number of columns in the table

From table to sharing

$$y = ab + ac + bc$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ * & 0 & 1 \\ * & 1 & 0 \end{pmatrix} \quad \begin{array}{l} y_0 = a_0 b_0 + a_0 c_0 + b_0 c_0 \\ y_1 = a_0 b_1 + a_0 c_1 + b_1 c_1 \\ y_2 = a_1 b_0 + a_1 c_0 \\ y_3 = a_1 b_1 + a_1 c_1 \\ y_4 = b_0 c_1 \\ y_5 = b_1 c_0 \end{array}$$

- Number of shares is higher than the lower bound of $(d+1)^t = 4$

From table to sharing

- Table implicitly satisfies the non-completeness property
- However, we need to check for correctness
  - For each monomial in the ANF all combinations of its share indices are present

$$y = ab + ac + bc + abc$$

$$\begin{array}{c} (a \quad b \quad c) \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}
\qquad
\begin{array}{cc} (a \quad b) & \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ \\ \\ \end{matrix} \end{array}
\qquad
\begin{array}{cc} (a \quad c) & \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ \\ \\ \end{matrix} \end{array}
\qquad
\begin{array}{cc} (b \quad c) & \\ \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} & \begin{matrix} 1 \\ 4 \\ \\ \\ 2 \\ 3 \end{matrix} \end{array}
\qquad
\begin{array}{cc} (a \quad b \quad c) & \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} \end{array}$$

Table of a n-bit function of degree t

- Table is optimal if it has the minimum number of rows while still satisfying correctness property
- A table D can be used to share any n-bit function of degree t iff every monomial of t input variables can be shared correctly
  - For any chosen t columns from D all input share combinations are present
- Optimal sharing is not unique, hence multiple optimal tables exist
- Two tables D_1 and D_2 are conjugate if there they are both optimal but they contain no same row between the two of them

Optimal sharing of a 2-bit function of degree 1 for any order d

- Number of rows is $d + 1$
- Trivial solution where i-th row is equal to (i, i)
- We can create $d + 1$ conjugate table by rotating the index in the second column

$$D_0 = \begin{array}{ccc} (a \quad b) & (a) & (b) \\ \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} \end{array} \qquad D_1 = \begin{array}{ccc} (a \quad b) & (a) & (b) \\ \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} & \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \end{array} \qquad D_2 = \begin{array}{ccc} (a \quad b) & (a) & (b) \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \\ 2 & 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} & \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \end{array}$$

Optimal sharing of n-bit functions of degree n-1 for any order d

- Start from optimal conjugate $d+1$ tables for $n=2$ of degree 1
- Given $d+1$ optimal conjugate tables with n columns for functions of degree $n-1$ construct $d+1$ optimal conjugate tables with $n+1$ columns for functions of degree n
- Start from $d+1$ optimal and conjugate tables $D_0, \dots, D_d$ with n columns and $(d+1)^{n-1}$ rows
- Obtain tables $T_0, \dots, T_d$ with $n+1$ columns and $(d+1)^n$ rows
  - For T_j append a column to D_i where each value is equal to $i+j \; mod \; (d+1)$ and add them as new rows in T_j

Example for $n = 3$ and $d = 2$

$$D_0 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix}, D_1 = \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{pmatrix}, \mathrm{D}_2 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix}$$

$$T_0 = \begin{pmatrix} & 0 \\ D_0 & 0 \\ & 0 \\ & 1 \\ D_1 & 1 \\ & 1 \\ & 2 \\ D_2 & 2 \\ & 2 \end{pmatrix}, T_1 = \begin{pmatrix} & 1 \\ D_0 & 1 \\ & 1 \\ & 2 \\ D_1 & 2 \\ & 2 \\ & 0 \\ D_2 & 0 \\ & 0 \end{pmatrix}, \mathrm{T}_2 = \begin{pmatrix} & 2 \\ D_0 & 2 \\ & 2 \\ & 0 \\ D_1 & 0 \\ & 0 \\ & 1 \\ D_2 & 1 \\ & 1 \end{pmatrix}$$

$$\begin{matrix} \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \\ 2 & 1 \end{pmatrix} \end{matrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 2 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 1 \\ 2 & 0 & 1 \\ 0 & 2 & 2 \\ 1 & 0 & 2 \\ 2 & 1 & 2 \end{pmatrix} \quad \begin{matrix} \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \\ 2 & 1 \end{pmatrix} \end{matrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 0 & 2 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \\ 2 & 1 & 0 \end{pmatrix} \quad \begin{matrix} \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \\ 2 & 1 \end{pmatrix} \end{matrix} \rightarrow \begin{pmatrix} 0 & 0 & 2 \\ 1 & 1 & 2 \\ 2 & 2 & 2 \\ 0 & 1 & 0 \\ 1 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 1 \\ 2 & 1 & 1 \end{pmatrix}$$

Application to PRINCE cipher

- We have applied our sharing construction to TI of PRINCE cipher
- S-Box is of degree 3 with 4-bit input
- First and second order implementation
- Compared to the previously known PRINCE TI where S-Box decomposition is used

DOM-like remasking of first order TI PRINCE

- Obtained shares have complementary domains that can use the same randomness

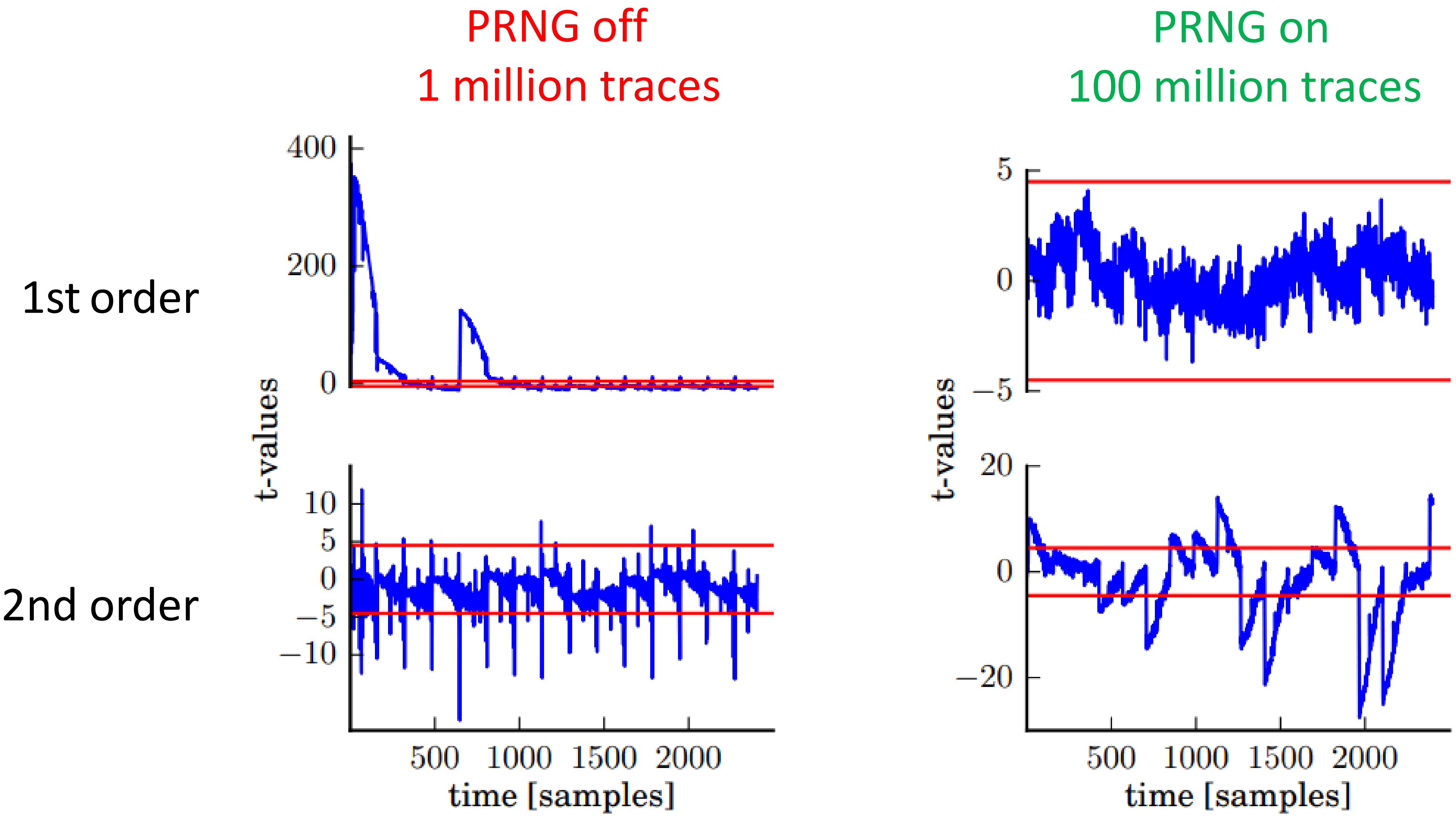
$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{array}{c} +0 \\ +R_1 \\ +R_2 \\ +R_3 \\ +R_3 \\ +R_2 \\ +R_1 \\ +0 \end{array}$$

Results

- We clearly outperform the previous PRINCE TI implementation with respect to latency
- First order implementation consumes less energy despite higher power consumption

PRINCE	Area @10 MHz (GE)	Power @10 MHz (uW)	Energy @10 MHz (pJ)	Rand/ Cycle (bits)	Clock # (cycle)	f_{max} (MHz)	Latency @ f_{max} (ns)
Unprotected	3589	59	71	0	12	393	30.5
[14] 1^{st} $(td+1)$ with S-box decomp.	9484	66	264	0	40	432	92.6
1^{st} $(d+1)$ w/o S-box decomp.	11596	100	241	48	24	376	63.8
2^{nd} $(d+1)$ w/o S-box decomp.	32444	374	898	1728	24	385	62.4

TVLA of first order implementation



Future work

- Explore other cases where degree of the n-bit function is n-2 or smaller
- Application to other use cases
- Remasking optimization considerations

Thank you!
Questions