RSAConference2019

Constructing TI-Friendly Substitution Boxes using Shift-Invariant Permutations

Si Gao, Arnab Roy, and Elisabeth Oswald

Outline

- Introduction
- Design Philosophy
- Sbox Constructions
- Implementations
- Summary

Outline

- Introduction
- Design Philosophy
- Sbox Constructions
- Implementations
- Summary

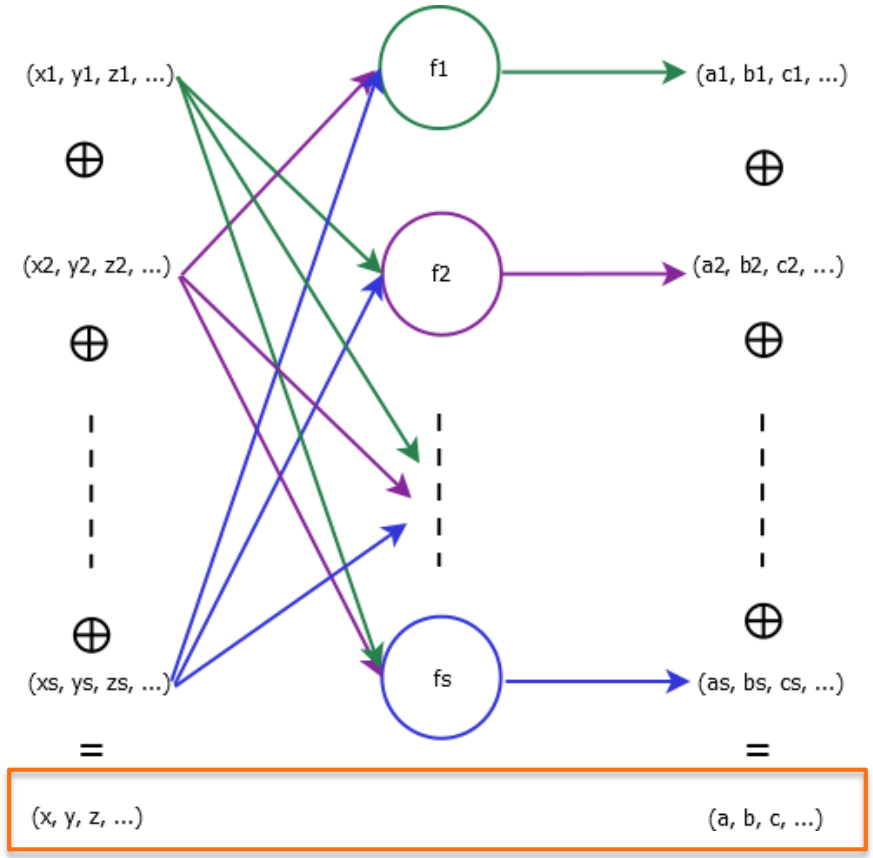
Block Ciphers & Side Channel Protection

- Component design (Sbox, diffusion layer etc.)
  - Strength against cryptanalysis
  - Implementation cost

- Side channel protection
  - Not a metric, yet draw attention
    - Bit-sliced masking: reducing the number of “AND2”
    - Threshold implementation (TI): “TI-friendly Sboxes”

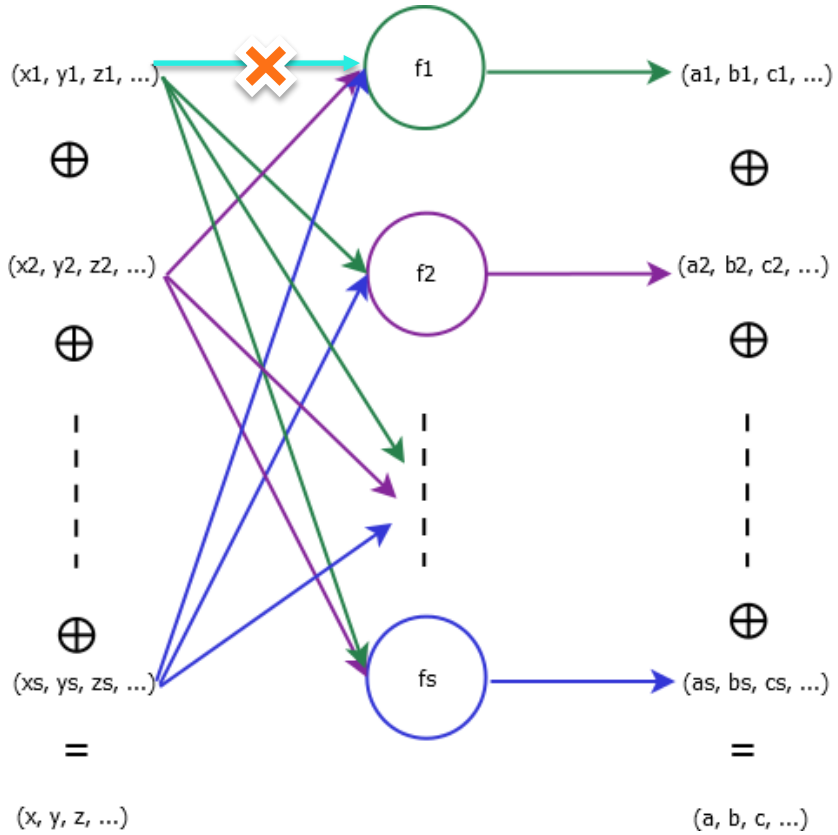
Threshold Implementation

- A countermeasure that based on the MPC concept
  - Goal: cope with hardware glitches
  - Requirements
    - Correct
    - Assigning each term (eg. a_i) to one of the "parties" (eg. f_i)



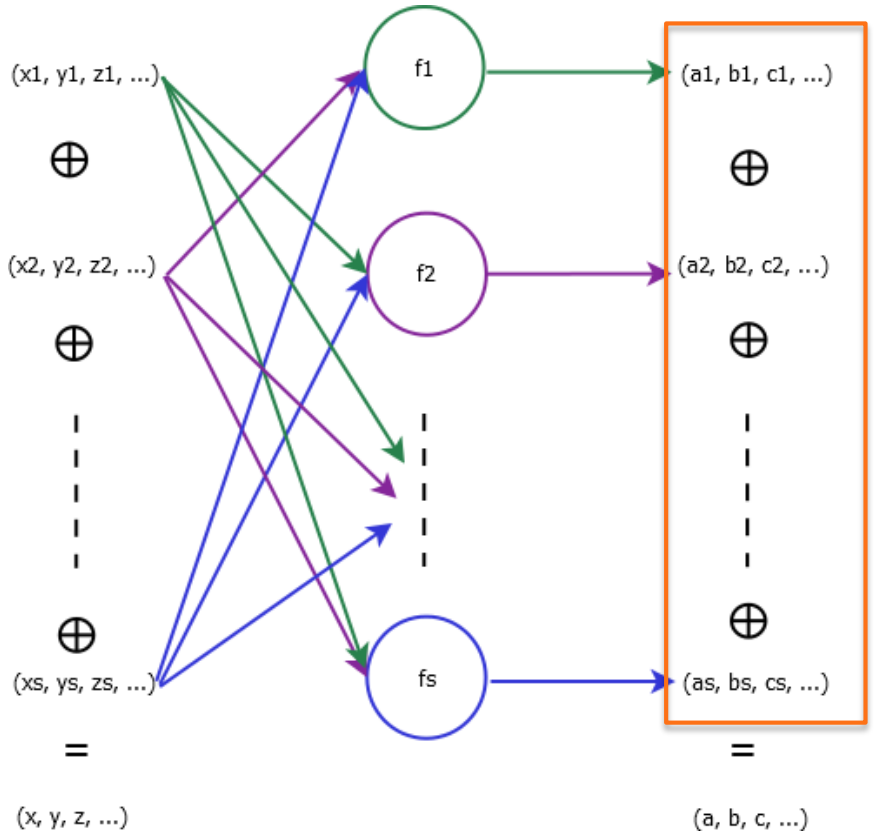
Threshold Implementation

- A countermeasure that based on the MPC concept
  - Goal: cope with hardware glitches
  - Requirements
    - Non-complete
    - Ensure deg(f)<s, so that every term uses at most s-1 shares



Threshold Implementation

- A countermeasure that based on the MPC concept
  - Goal: cope with hardware glitches
  - Requirements
    - Uniform
    - Add fresh randomness
    - Otherwise, no general constructions



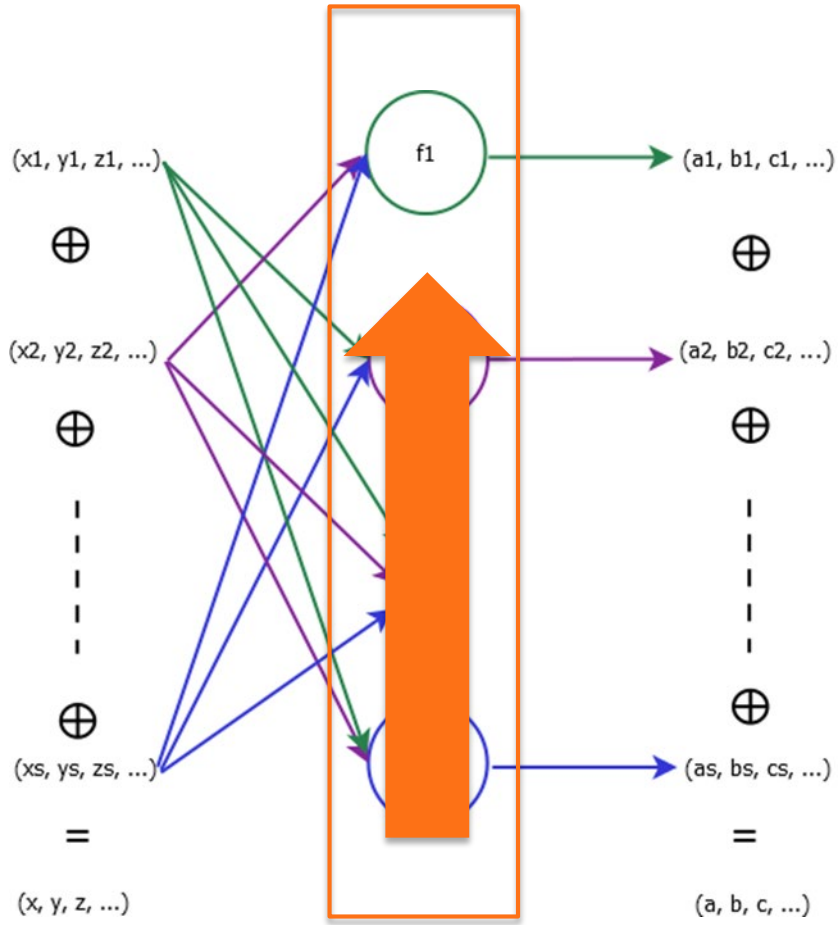
Should be a valid share!

TI-friendly Designs

- Diffusion Layer
  - For an *s*-share scheme, simply performing linear operation on each share
- Sbox
  - Lower degree functions/decomposition
    - Idea: less shares (*s*) save time/area
    - All 3*3/4*4 Sboxes up to affine equivalence [CHES 12]
    - 5-bit/ some 6-bit quadratic permutations [FSE 17, BFA18]
  - 8-bit Sbox constructions with smaller Sboxes
    - Feistel/SPN/MISTY [CHES 16]
    - Other structures [SITB 17]

Implementation perspective

- Hardware
  - Main target
    - Glitches etc.
  - Serial TI
    - If f is "intrinsically" uniform, all "parties" $(f_1,\ldots,f_s)$ share the same functionality
    - Implement only f_1, learn others by shifting shares [COSADE 13, 18]

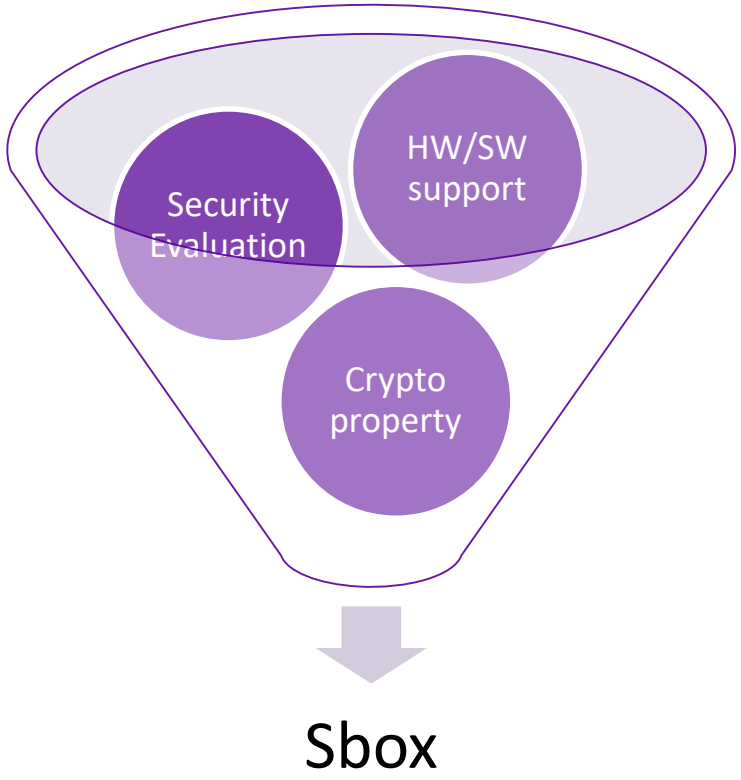


Implementation perspective

- Software
  - Less discussed
    - “Glitches” become less dreadful
    - Lost its competitive advantages in a restricted architecture
  - Why software-TI?
    - Obscure internal operations → unexpected leakage
    - 1st order leakage in global look-up tables [COSADE 18’]
    - Security order reduction in Boolean masking [CARDIS 14]

Goal

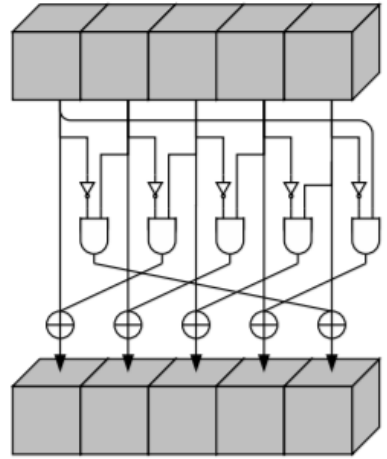
- "Constructing TI-friendly Sbox"
  - Considering SW platforms
  - Realistic implementations & Security Evaluation
  - Meet cryptographic requirements



Outline

- Introduction
- Design Philosophy
- Sbox Constructions
- Implementations
- Summary

Design Philosophy



- "Shift-Invariant"
  - Definition
    - For any rotated shift $\boldsymbol{\tau}$, F satisfies

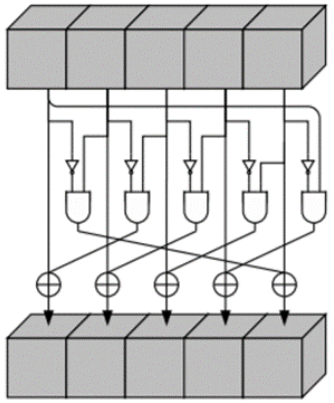
$$F \circ \tau = \tau \circ F$$

    - Originally studied in Joan Daemen's thesis, Chapter 6 [JDA 95]
    - Cellular automaton perspective: 7*7 Sbox [CC 18]
    - Keccak's χ^2

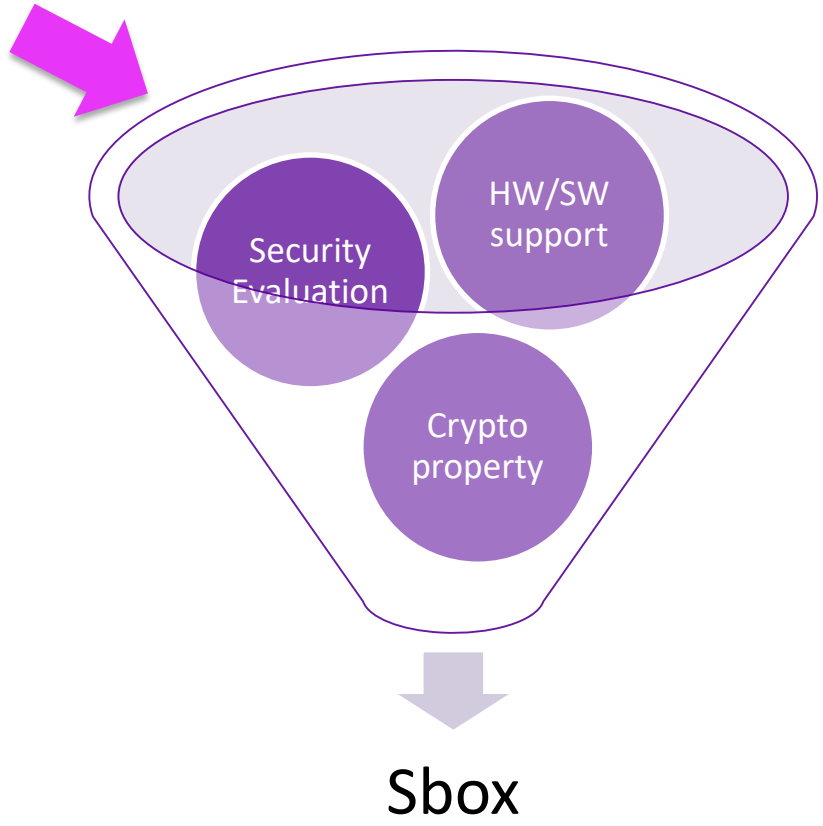
Source: Keccak sponge function family main document, https://keccak.team/obsolete/Keccak-main-1.1.pdf

Design Philosophy

- Why “Shift-Invariant”?
  - Software Implementation
    - Suitable for bit-slicing
    - Fine-grained



Source: Keccak sponge function family main document, https://keccak.team/obsolete/Keccak-main-1.1.pdf

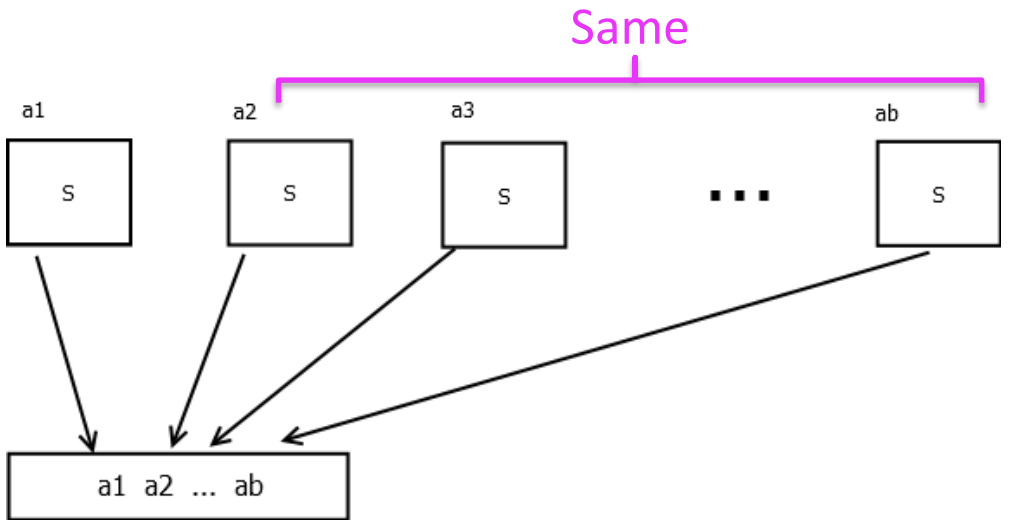


Design Philosophy

- Detour
  - Efficiency in bit-slicing
    - Pack the same bits to one register
    - Best when b>processor's bit width

    Otherwise, "borrow" from other blocks
    - Part of the "slicing cost"
  - Shift-invariant
    - "Easier slicing"

    Eg. a 32-bit shift-invariant function does not take any "slicing" on 32-bit processors



Design Philosophy

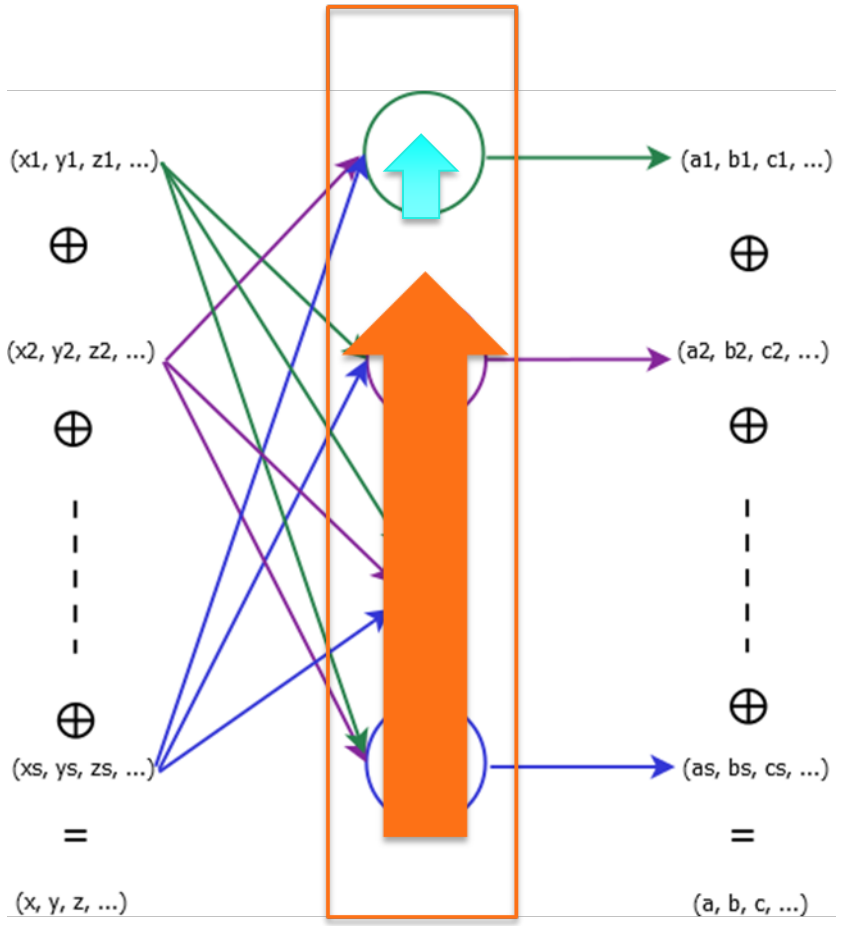
- Why “Shift-Invariant”?
  - Serial TI
    - Provides more trade-off options

    Eg,

    $a_1=f_a(x_1,y_1,z_1,\ldots)$

    $b_1=f_b(x_1,y_1,z_1,\ldots)$

    $=f_a(z_1,y_1,\ldots,x_1)$
    - 1 bit of 1 share v.s. all the bits in 1 share
    - Hardware: smaller footprint
    - Software: easier for bit-slicing



Design Philosophy

- To sum up, in our constructions, we choose
  - Quadratic (deg=2) permutations
    - Quadratic: 3-share TI (less shares -> lower cost)
    - Permutation: Sbox construction without invertible structure
  - Shift-invariant
    - More trade-off options (eg. 1-bit implementation)
  - Uniform TI
    - Shift-invariant on each share
    - Further trade-off options (eg. 1-bit of one share)

Outline

- Introduction
- Design Philosophy
- Sbox Constructions
- Implementations
- Summary

Sbox Constructions

- Quadratic building block search
  - *n=4*
    - Total 2^{11}
    - Permutation 24
    - 3-share TI uniform 24
  - *n=8*
    - Total 2^{37}
    - Permutation 520128
    - 3-share TI uniform 520128

n	All f	Has $x_0 \& c = 0$	Balanced	Permutation	TI Permutation
4	2048	952	392	24	24

Table 1. Shift-invariant quadratic TI permutations: $n = 4$

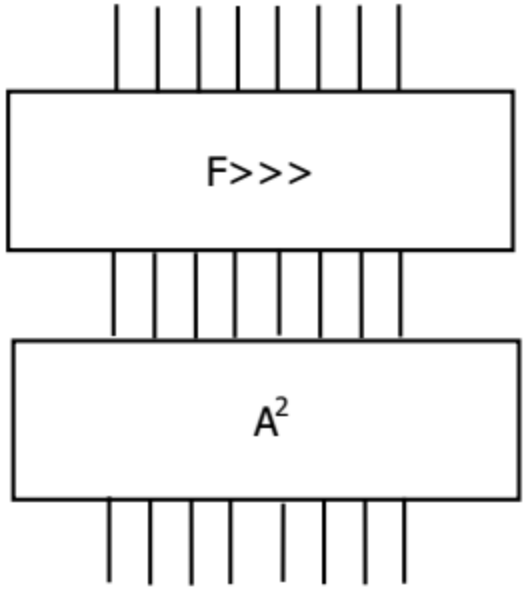
only for n=4 or 8!

n	All f	Has $x_0 \& c = 0$	Balanced	Permutation	TI Permutation
8	2^{37}	68451041152	29986581632	520128	520128

Table 2. Shift-invariant quadratic TI permutations: $n = 8$

Sbox Constructions

- Design Architecture
  - Full range SPN
    - Branches: not perfect for “slicing”
  - Permutation Layer
    - Not shift-invariant
      - F already covers all possible options
      - Security concern [FSE 10]
    - AES Xtime-like operation
      - Rotate with conditional XOR
      - Do it twice for better diffusion



$$\mathbf{A} = \begin{bmatrix} a_{1,1} & 1 & 0 & \dots & 0 \\ a_{2,1} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n-1,1} & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}$$

Sbox Constructions

- n=4
  - Diff.=4, Lin.=8 (a.k.a. “optimal”)
    - 16 options, 2 rounds
    - One instance:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	9	4	A	3	7	8	C	5	B	6	D	E	F

Table 3. Shift-invariant quadratic TI permutation for S4

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	4	8	A	F	C	6	9	1	E	B	D	7	5	3	2

Table 4. Final Sbox for S4

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Sbox Constructions

- n=8
  - Diff.<=8, Lin.<=72
    - 6 options, 3 rounds
    - One instance: diff=8, lin=64, deg=6

$$\mathbf{A} = \begin{bmatrix} 1&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 1&0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \end{bmatrix}$$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	4c	98	db	31	0a	b7	83	62	56	14	2f	6f	2c	07	4b
1	c4	dd	ac	ba	28	46	5e	3f	de	bf	58	36	0e	18	96	8f
2	89	ca	bb	f7	59	6d	75	4e	50	6b	8c	b8	bc	f0	7e	3d
3	bd	ab	7f	66	b0	d1	6c	02	1c	72	30	51	2d	34	1f	09
4	13	82	95	0b	77	91	ef	06	b2	5b	da	3c	ea	74	9c	0d
5	a0	64	d6	1d	19	aa	71	cd	79	c5	e1	52	fc	37	7a	be
6	7b	e5	57	c6	fe	17	cc	2a	61	87	a3	4a	d8	49	04	9a
7	38	f3	e4	20	60	dc	a2	11	5a	e9	68	d4	3e	fa	12	d9
8	26	ed	05	c1	2b	97	16	a5	ee	5d	23	9f	df	1b	0c	c7
9	65	fb	b6	27	b5	5c	78	9e	d5	33	e8	01	39	a8	1a	84
A	41	85	c8	03	ad	1e	3a	86	32	8e	55	e6	e2	29	9b	5f
B	f2	63	8b	15	c3	25	a4	4d	f9	10	6e	88	f4	6a	7d	ec
C	f6	e0	cb	d2	ae	cf	8d	e3	fd	93	2e	4f	99	80	54	42
D	c2	81	0f	43	47	73	94	af	b1	8a	92	a6	08	44	35	76
E	70	69	e7	f1	c9	a7	40	21	c0	a1	b9	d7	45	53	22	3b
F	b4	f8	d3	90	d0	eb	a9	9d	7c	48	f5	ce	24	67	b3	ff

Table 7. The quadratic shift-invariant permutation S

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	6d	f1	8f	3d	80	b4	31	50	82	3f	2e	51	0f	1c	c1
1	a0	c4	25	12	5d	67	a4	65	81	1e	e0	1d	38	e5	97	05
2	19	f3	da	03	ba	91	07	b5	9e	7f	c7	77	32	76	a3	e1
3	98	93	94	5c	7e	17	c2	0a	70	43	cb	a6	5e	ac	7c	a1
4	8b	a5	d6	2a	18	ed	c0	57	9a	6b	23	06	88	08	2b	cd
5	24	7b	d2	2c	e7	59	69	dc	9f	0e	61	75	20	89	fc	ff
6	0c	bd	27	9d	16	b9	86	fd	73	d7	b1	5a	f0	5f	14	40
7	74	e3	df	d5	f2	36	e6	64	2f	e9	92	e4	fa	71	be	b2
8	9c	ce	41	42	b6	63	87	a2	30	29	cc	ef	8c	68	c6	3c
9	4a	66	b0	c9	bc	dd	8e	45	21	90	d1	ae	1f	62	56	db
A	48	96	f6	ab	8d	a7	58	b7	22	f8	ec	28	0d	f7	bb	f5
B	2d	6a	4d	fe	eb	0b	01	13	52	ea	7a	10	f9	72	7d	8a
C	6c	6e	34	95	d0	c5	6f	49	ee	4b	b3	4c	af	3b	a8	4f
D	4e	39	c3	9b	a9	84	78	11	60	55	aa	85	15	02	fb	09
E	37	ca	79	47	3e	f4	d8	e2	53	d9	26	3a	99	e8	c8	33
F	de	54	5b	b8	1a	83	46	35	d3	ad	44	d4	bf	04	cf	1b

Table 8. The overall Sbox

Outline

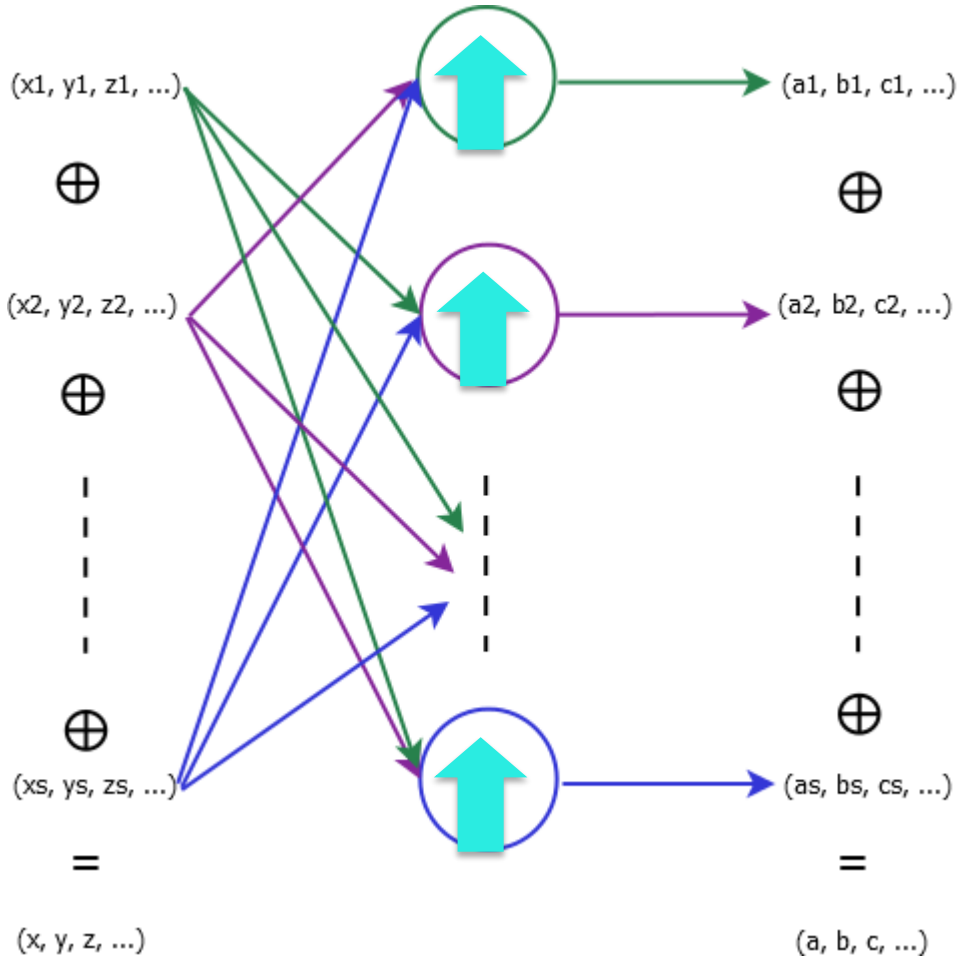
- Introduction
- Design Philosophy
- Sbox Constructions
- Implementations
- Summary

Implementations

- Software
  - Target platform
    - ARM M0 (Thumb)/M3 (ARM)
    - 32 bit data width
  - Possible trade-offs
    - Size-based: same share, different bit
    - Moderate trade-off

    $(x_1^{[1]}\ x_1^{[2]} \ldots\ y_1^{[1]} y_1^{[2]} \ldots z_1^{[1]}\ z_1^{[2]} \ldots)$

    $>>>_1$ becomes $>>>_b$



Implementations

- Software
  - Results
    - No fresh-randomness
    - Otherwise, not “that” fast…
  - Not a fair comparison
    - Effort spent on optimizing AES/PRESENT’s Sbox

Number of concurrent Sboxes

	Size	Diff.	Lin.	Deg.	1st Order Protected		
					Randomness	Cycles	
						Thumb	ARM
PRESENT(BS) [36]	4	4	8	3	64	n/a	796/16
PRESENT($F \circ G$) [36]	4	4	8	3	128	n/a	686/8
$S4$ (our result)	4	4	8	3	0	870/8	654/8
AES(BS) [36]	8	4	32	7	512	n/a	4698/16
AES(KHL) [36]	8	4	32	7	192	n/a	2309/8
$S8$ (our result)	8	8	64	6	0	3627/4	2169/4

Table 5. Software Performance of various Sboxes

Implementations

- Software
  - Not a fair comparison
    - Possible security overhead [EUROCRYPT 17]
    - Could be not as trivial as it sounds …..

Practical flaws

factor up to 2 in the security order [1]. This is clearly a chip-dependent matter whereas our study does not focus on a particular chip but on generic ARM assembly. That is why we do not solve this issue for our implementation. We stress that solving this issue on a given chip might be a time-consuming engineering problem but we expect that a hardened implementation should have performances close to our original implementation. Indeed, and as aforementioned, the update merely consists in clearing the data path at some specific points in the assembly, which should not imply a very strong overhead. We hence believe

Implementations

- Software
  - Security Evaluation
    - Cortex M0: NXP LPC1114

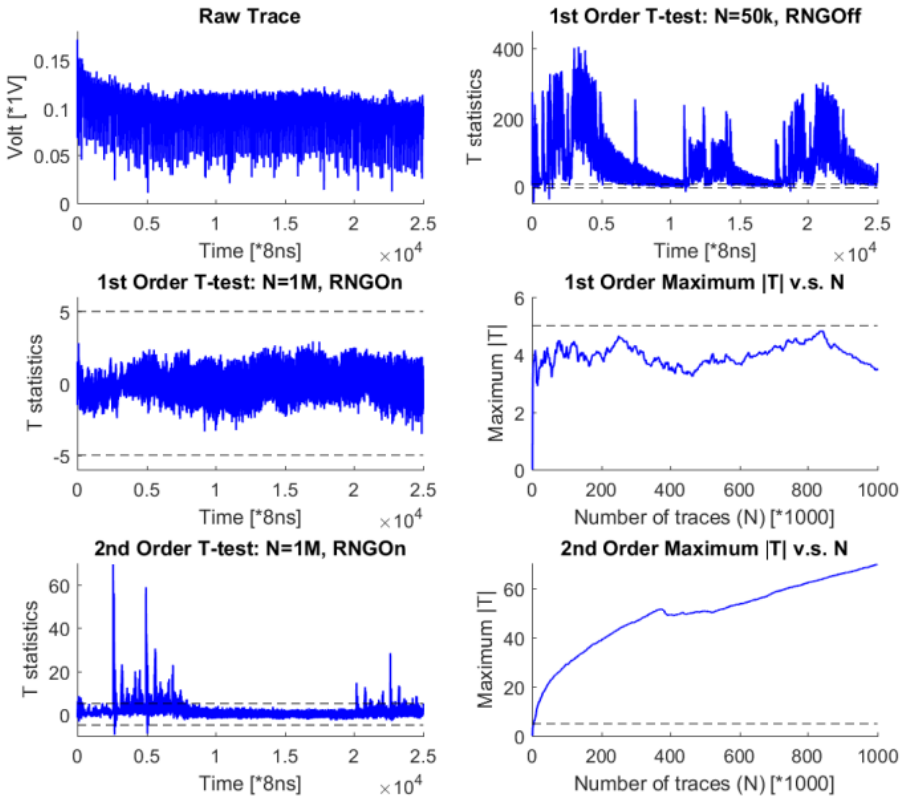


Fig. 3. Software evaluation of S4

Implementations

- Hardware
  - Selected trade-off
    - 1-bit implementation
    - 2D rotation: possible on hardware
  - Possible pitfall
    - Glitches' leakage on shifting shares
    - Pre-charge the input to 0
    - 1 extra cycle (per "shift-share")

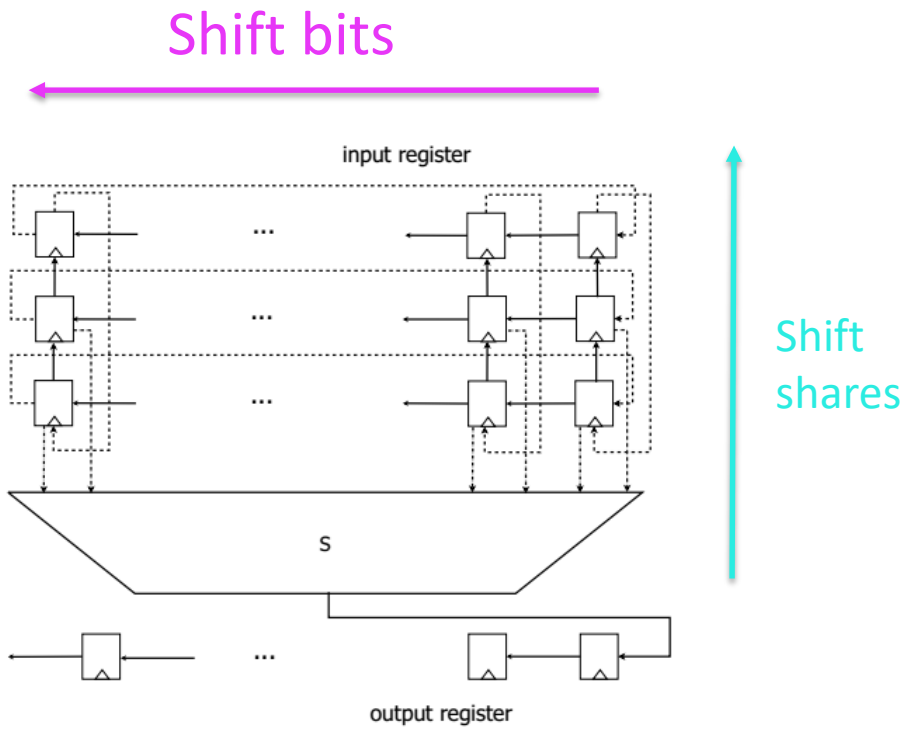


Fig. 2. Hardware schematic of shift invariant transformation S

Implementations

- Hardware
  - Results
    - Trade cycles for GEs
    - Add cost to the control logic…
    - Not attractive for *n=8*

	Size	Diff.	Lin.	Deg.	Rounds	Protected		
						Area(GE)	Delay(ns)	Cycles
PRESENT [24]	4	4	8	3	n/a	151	—	6
GIFT [7]	4	6	8	3	n/a	172.5	—[6]	6
$S4$	4	4	8	3	2	54	0.72	28
AES [38]	8	4	32	7	n/a	2224	—	3
SB_1 [15]	8	16	64	6	8	51	1.09	8
SB_4 [15]	8	8	56	7	5	202	2.10	5
$S8$	8	8	64	6	3	181	1.89	78

Table 6. Hardware evaluation of various Sboxes

Implementations

- Hardware
  - Security Evaluation
    - SAKURA-X: Kintex-7 FPGA

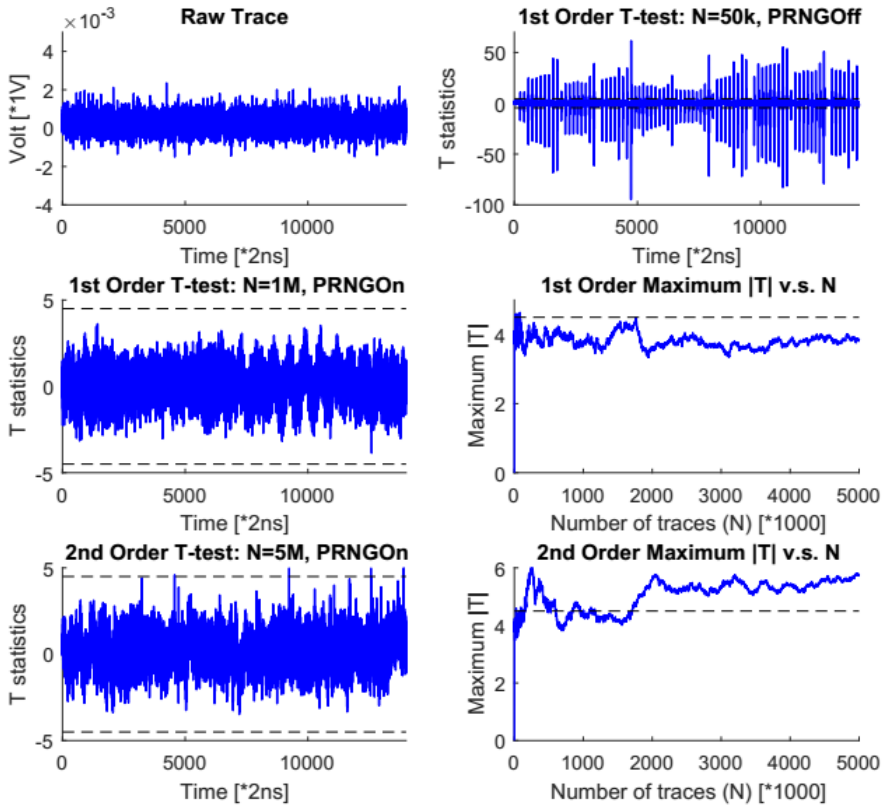


Fig. 6. Hardware evaluation of S8

Outline

- Introduction
- Design Philosophy
- Sbox Constructions
- Implementations
- Summary

Summary

- TI-friendly Sbox designs
  - Shift-invariant permutations
    - 3-share implementation
    - shift-invariant TI-form
    - Easier for bit-slicing
  - Results
    - 4-bit Sbox: 2 rounds/ 8-bit Sbox: 3 rounds
    - HW/SW Implementation
    - Security evaluation with TVLA test

Summary

- Discussion
  - 8-bit Sbox constructions
    - Using 4-bit Sboxes as building blocks still seems more attractive [CHES 16]
    - Shift-invariant for non-Sbox designs?
  - Implementation pitfalls
    - Non-academic, yet not an easy task for engineers!
    - Better understanding of processors & leakages --- obscure & time consuming
    - More security margins (eg. more shares) --- higher cost!

Reference

- [CHES 12] Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., St¨utz, G.: Threshold Implementations of All 3×3 and 4×4 S-Boxes. In: Cryptographic Hardware and Embedded Systems- CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12,2012. Proceedings
- [FSE 17] Bozilov, D., Bilgin, B., Sahin, H.A.: A Note on 5-bit Quadratic Permutations' Classification. IACR Trans. Symmetric Cryptol. 2017 398-404
- [BFA18] De Meyer, L., Bilgin, B.: Classification of Balanced Quadratic Functions. IACR Cryptology ePrint Archive 2018
- [CHES 16] Boss, E., Grosso, V., G¨uneysu, T., Leander, G., Moradi, A., Schneider, T.: Strong 8-bit Sboxes with efficient masking in hardware extended version. J. Cryptographic Engineering 7(2) (2017) 149-165
- [SITB 17] Meyer, L.D., Varici, K.: More Constructions for strong 8-bit S-boxes with efficient masking in hardware. In: Proceedings of the 38th Symposium on Information Theory in the Benelux, Delft,NE, Werkgemeenschap voor Informatie- en Communicatietheorie (2017) 11

Reference

- [COSADE 13] Kutzner, S., Nguyen, P.H., Poschmann, A., Wang, H.: On 3-Share Threshold Implementations for 4-Bit S-boxes. In: Constructive Side-Channel Analysis and Secure Design - 4th International Workshop, COSADE 2013, Paris, France, March 6-8, 2013, Revised Selected Papers. (2013) 99-113
- [COSADE 18] Wegener, F., Moradi, A.: A First-Order SCA Resistant AES Without Fresh Randomness. In: Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings. (2018) 245-262
- [COSADE 18'] Sasdrich, P., Bock, R., Moradi, A.: Threshold Implementation in Software - Case Study of PRESENT. In: Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings. (2018) 227-244

Reference

- [CARDIS 14] Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.: On the Cost of Lazy Engineering for Masked Software Implementations. In: Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers. (2014) 64-81
- [JDA 95] Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995) http://jda:noekeon:org/
- [CC 18] Mariot, L., Picek, S., Leporati, A., Jakobovic, D.: Cellular automata based s-boxes. Cryptography and Communications (May 2018)
- [COSADE 18’’] Corre, Yann Le et al. “Micro-Architectural Power Simulator for Leakage Assessment of Cryptographic Software on ARM Cortex-M3 Processors.” IACR Cryptology ePrint Archive 2017 (2017): 1253.

Reference

- [EUROCRYPT 17] Goudarzi, D., Rivain, M.: How Fast Can Higher-Order Masking Be in Software? In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. (2017) 567-597