RUHR-UNIVERSITÄT BOCHUM

RUB

Strong 8-bit Sboxes with Efficient Masking in Hardware

18th August, 2016.

Erik Boss[1] Vincent Grosso[1] Tim Güneysu[2]

Gregor Leander[1] Amir Moradi[1] Tobias Schneider[1]

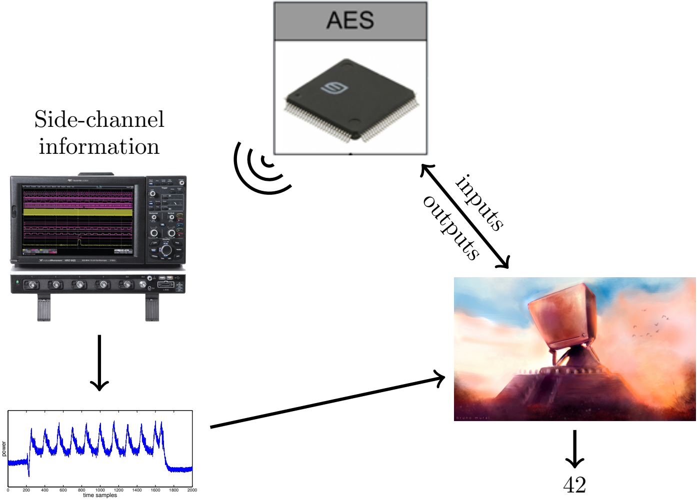
[1] Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany

[2] University of Bremen and DFKI, Germany

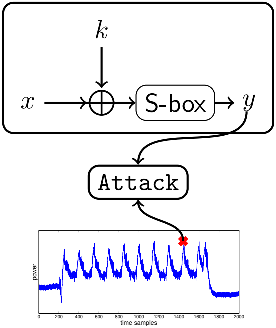
hgi Horst Görtz Institute for IT-Security

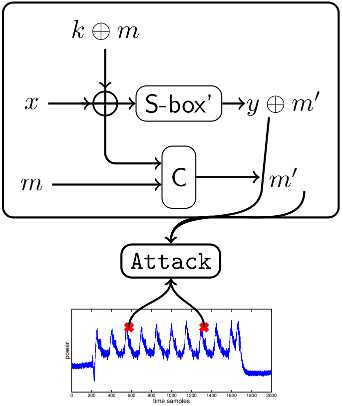
UbiCrypt Cryptography in Ubiquitous Computing

Side-channel attacks



Masking: principle

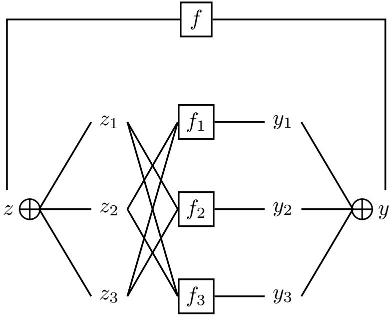




Masking: summary

- Expecting: number of measurements grows up exponentially in the number of shares with noise as a basis
- Security conditions
  - Noise
  - Randomness
  - Independence of the leakages: possible issue in hardware due to glitches

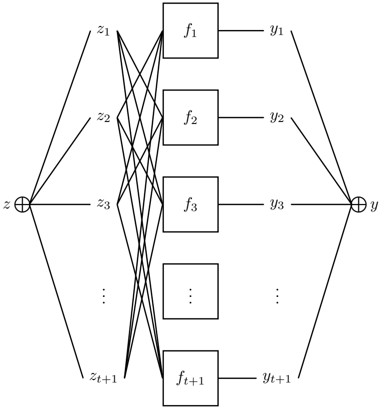
Threshold implementations



- Correctness: the shared functions compute the actual function
- Non-completeness: each sub-circuit is independent of one share
- Uniformity: the output of the shared function is a uniform sharing (use fresh randomness if needed)

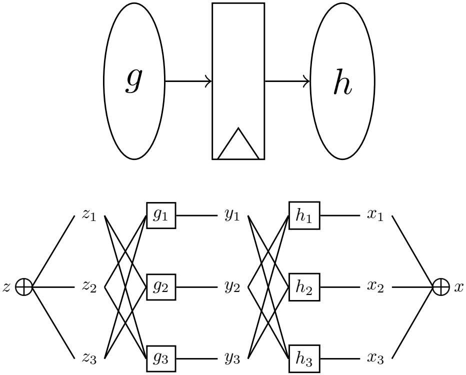
Algebraic decomposition

Number of shares for TI:
degree of f + 1

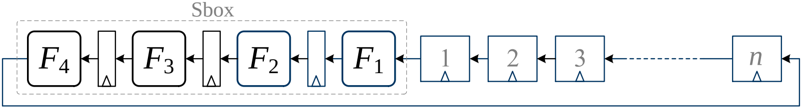


$\Rightarrow$

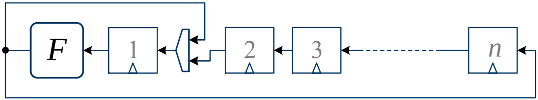
$$f = h \circ g$$



Different implementation techniques



Raw



Iterative

Previous work

- Exhaustive search for small S-boxes (i.e. $n \leqslant 4$)
  4-bit S-boxes: 302 bijective classes $\Rightarrow$ 35 efficient TI with 3 shares [CHES 2012]
- Look for interesting S-boxes and try to find a nice threshold implementation, e.g.:
  - AES [EUROCRYPT 2011, Africacrypt 2014] 8-bit
  - Fides [CHES 2013] 5-bit, 6-bit
  - Keccak [CARDIS 2013] 5-bit
- Large S-box with good cryptographic/threshold implementation properties?

Idea

Use results for small S-boxes to find TI of larger S-boxes

Idea

Use results for small S-boxes to find TI of larger S-boxes

Lot of existing S-boxes use small S-boxes to build larger one

- CLEFIA
- Crypton
- Fantomas
- ICEBERG
- Khazad
- Robin (iterative implementation)
- Scream v3
- Whirlpool

Idea

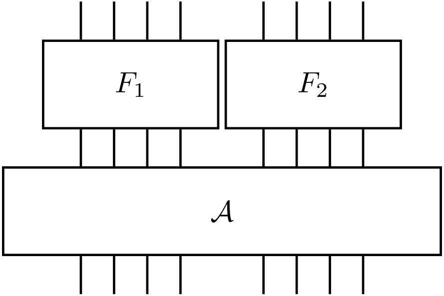
Use results for small S-boxes to find TI of larger S-boxes

Lot of existing S-boxes use small S-boxes to build larger one

- CLEFIA
- Crypton
- Fantomas
- ICEBERG
- Khazad
- Robin (iterative implementation)
- Scream v3
- Whirlpool

Can we achieve better results? Can we take advantage of iterative implementation?

From small to large S-box: SPN

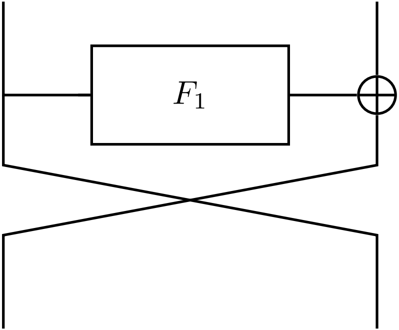


Structure used for: Iceberg, Khazad, Whirlpool,...

- 16! choices for F_1 and F_2
  $\rightarrow F_1, F_2$ easy to share
  4-bit S-box $\rightarrow$ 35
  - $\mathcal{A}$ bit permutation 8!
  - $\mathcal{A}$ $\mathbb{F}_{16}$-linear layer 61200
- Constant: 256

Cost $\simeq 2^{32}$

From small to large S-box: Feistel



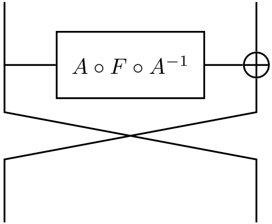
Structure used for: Robin, Scream v3,...
Structure well studied up to 3 rounds

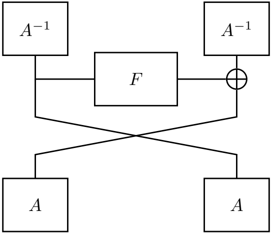
- 2^{64} choices for F_1

Feistel gives uniformity for "free" (if F_1 can be computed in one clock cycle)

Reduce the search space

Affine equivalence: $F_1 = A \circ F \circ B + C$





Reduce the search space from all function 2^{64} to function of the $A \circ F + C$

- F is an instance of an affine class
- A is an affine permutation
- C is a linear mapping

Cost $\simeq 2^{46.5} \Rightarrow$ use GPUs

Cryptographic properties

- Bijective
- Non-linearity
- Differential uniformity
- Algebraic degree

Comparison

	Diff.	Lin.	Deg.	Threshold Implementation Area[GE] Iter.	raw	Stage #	Mask #	Type
AES	Best Known				4244	5	48	Inversion
					3708	3	44	
	4	32	7		3653	3	44	
					2835	3	32	
Whirlpool	8	56	7		2203	9	0	SPN
SB_4 (this work)	8	56	7	202	1507	5	0	Feistel
SB_3 (this work)	8	60	7	273	1498	4	0	SPN
ICEBERG	8	64	7		2115	9	0	SPN
Khazad	8	64	7		2062	9	0	SPN
Scream v3	8	64	6		2204	6	0	Feistel
Fantomas	16	64	5		766	4	0	SPN
Robin	16	64	6	319	1180	6	0	Feistel
SB_1 (this work)	16	64	6	51	1189	8	0	SPN
SB_2 (this work)	16	64	4	253	631	2	0	SPN

Comparison

RUB

	Diff.	Lin.	Deg.	Threshold Implementation Area[GE] Iter.	raw	Stage #	Mask #	Type
AES	Best Known				4244	5	48	Inversion
	4	32	7		3708	3	44	
					3653	3	44	
					2835	3	32	
Whirlpool	8	56	7		2203	9	0	SPN
SB_4 (this work)	8	56	7	202	1507	5	0	Feistel
SB_3 (this work)	8	60	7	273	1498	4	0	SPN
ICEBERG	8	64	7		2115	9	0	SPN
Khazad	8	64	7		2062	9	0	SPN
Scream v3	8	64	6		2204	6	0	Feistel
Fantomas	16	64	5		766	4	0	SPN
Robin	16	64	6	319	1180	6	0	Feistel
SB_1 (this work)	16	64	6	51	1189	8	0	SPN
SB_2 (this work)	16	64	4	253	631	2	0	SPN

Same round functions allow us to make iterative implementation

Comparison

RUB

	Diff.	Lin.	Deg.	Threshold Implementation Area[GE] Iter.	raw	Stage #	Mask #	Type
AES	Best Known				4244	5	48	Inversion
					3708	3	44	
	4	32	7		3653	3	44	
					2835	3	32	
Whirlpool	8	56	7		2203	9	0	SPN
SB_4 (this work)	8	56	7	202	1507	5	0	Feistel
SB_3 (this work)	8	60	7	273	1498	4	0	SPN
ICEBERG	8	64	7		2115	9	0	SPN
Khazad	8	64	7		2062	9	0	SPN
Scream v3	8	64	6		2204	6	0	Feistel
Fantomas	16	64	5		766	4	0	SPN
Robin	16	64	6	319	1180	6	0	Feistel
SB_1 (this work)	16	64	6	51	1189	8	0	SPN
SB_2 (this work)	16	64	4	253	631	2	0	SPN

Interesting tradeoff for different implementation

Comparison

	Diff.	Lin.	Deg.	Threshold Implementation Area[GE] Iter.	raw	Stage #	Mask #	Type
AES	Best Known				4244	5	48	Inversion
					3708	3	44	
	4	32	7		3653	3	44	
					2835	3	32	
Whirlpool	8	56	7		2203	9	0	SPN
SB_4 (this work)	8	56	7	202	1507	5	0	Feistel
SB_3 (this work)	8	60	7	273	1498	4	0	SPN
ICEBERG	8	64	7		2115	9	0	SPN
Khazad	8	64	7		2062	9	0	SPN
Scream v3	8	64	6		2204	6	0	Feistel
Fantomas	16	64	5		766	4	0	SPN
Robin	16	64	6	319	1180	6	0	Feistel
SB_1 (this work)	16	64	6	51	1189	8	0	SPN
SB_2 (this work)	16	64	4	253	631	2	0	SPN

No 8-bit balanced Feistel with identical round functions up to 5 iterations achieve better cryptographic properties than SB_4

Conclusion

- Various S-boxes with decent cryptographic properties and efficient TI
- Even for unprotected implementation they are efficient (cf. the paper)
- Some S-boxes have also good behavior for (masked) bitslice implementation (cf. the paper, SB_2 have similar number of AND gates as Robin and Scream v3)

Thanks!

Questions?