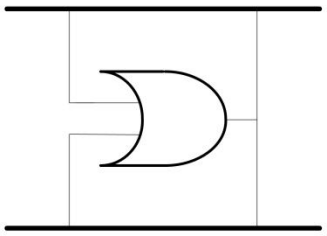
Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs

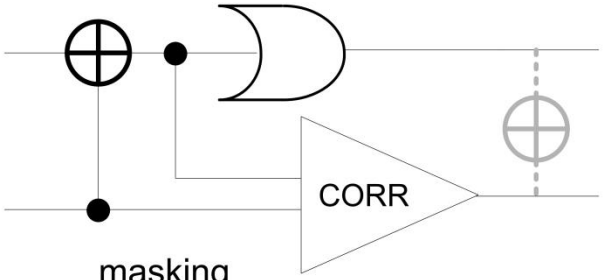
M. Medwed, **F.-X. Standaert**, A. Joux
NXP & UCL Crypto Group & Univ. Versailles

CHES 2012, Leuven, Belgium

SCA security (implementation level)



dual-rail logic styles
[TV02, ...]

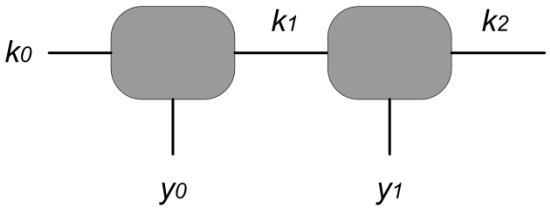


masking
(aka secret sharing) [GP00,CJRR99, ...]

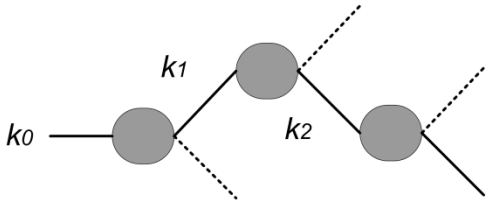
Others:
- noise addition
- random delays
- shuffling
- ...

SCA security (mathametical level)

leakage-resilient stream ciphers
[DP08,YSPY10, ...]



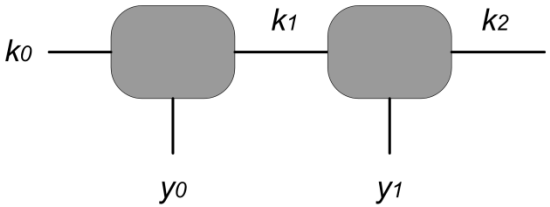
leakage-resilient PRFs
[S+09,DP10, ...]



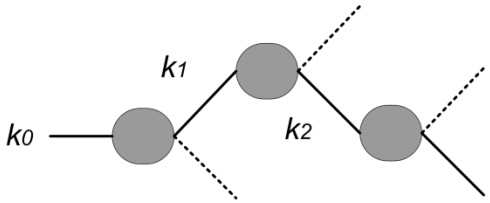
Others:
- auxiliary input model
- bounded retrieval model
- AC0 leakages
- ...

Limitations of current approaches

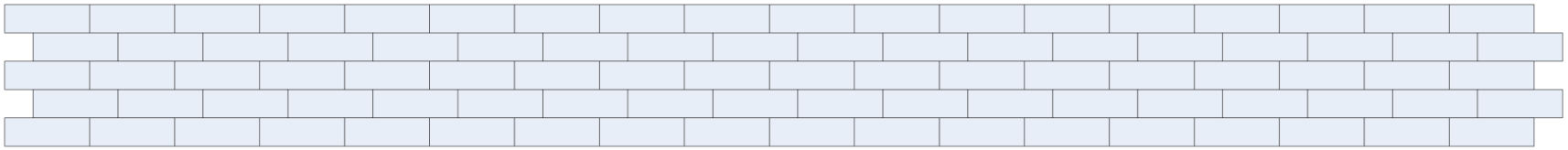
leakage-resilient stream ciphers
[DP08,YSPY10, ...]

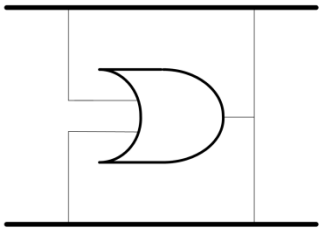


leakage-resilient PRFs
[S+09,DP10, ...]

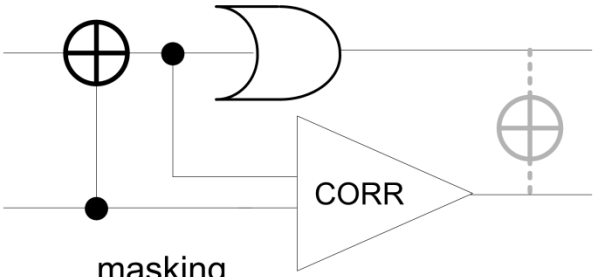


Others:
- auxiliary input model
- bounded retrieval model
- AC0 leakages
- ...





dual-rail logic styles
[TV02, ...]

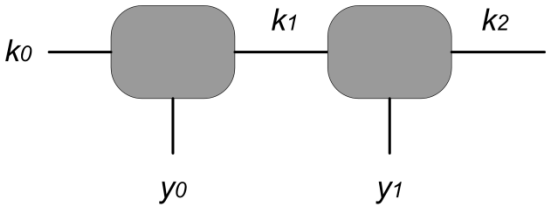


masking
(aka secret sharing) [GP00,CJRR99, ...]

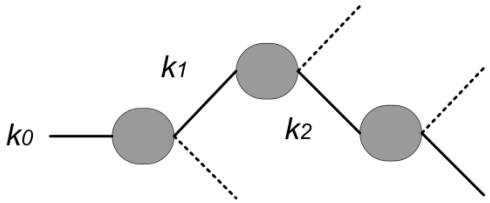
Others:
- noise addition
- random delays
- shuffling
- ...

Direction for improvements #1

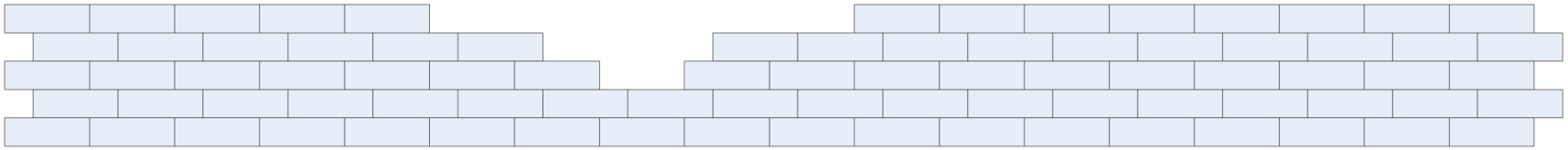
leakage-resilient stream ciphers
[DP08,YSPY10, ...]

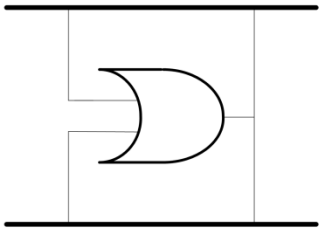


leakage-resilient PRFs
[S+09,DP10, ...]

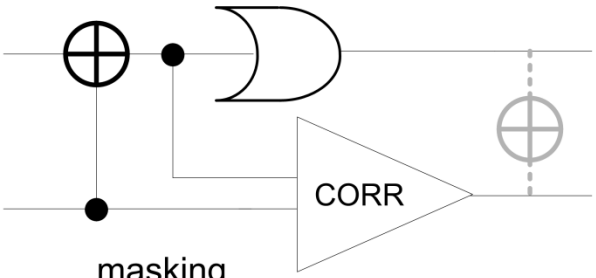


More realistic models
More efficient constructions
Better bounds
...





dual-rail logic styles
[TV02, ...]

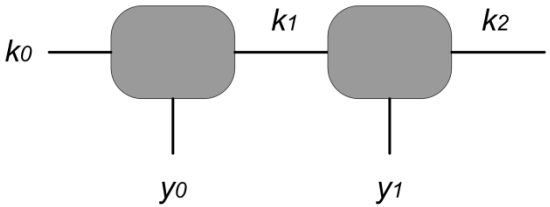


masking
(aka secret sharing) [GP00,CJRR99, ...]

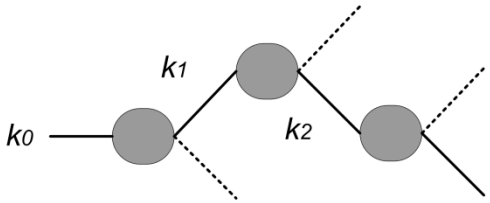
Others:
- noise addition
- random delays
- shuffling
- ...

Direction for improvements #2

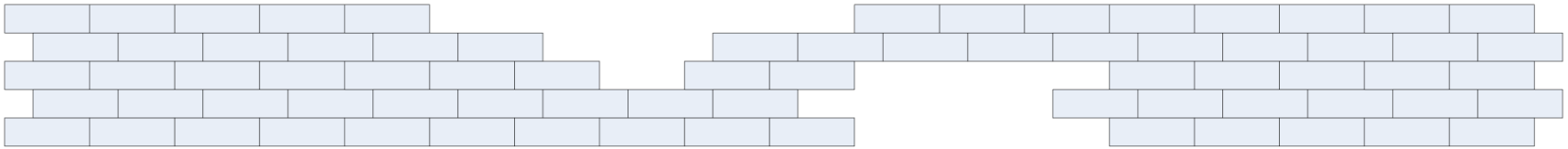
leakage-resilient stream ciphers
[DP08,YSPY10, ...]

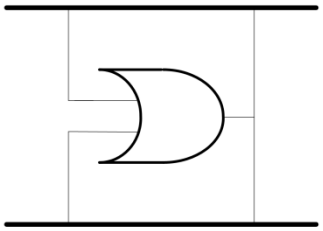


leakage-resilient PRFs
[S+09,DP10, ...]

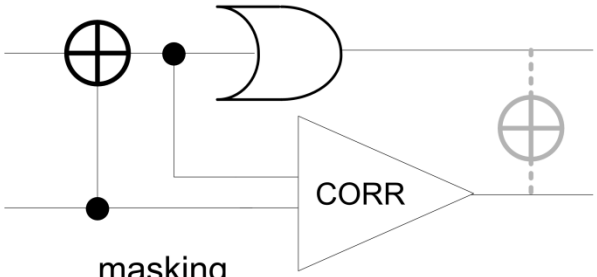


More realistic models
More efficient constructions
Better bounds
...





dual-rail logic styles
[TV02, ...]



masking
(aka secret sharing) [GP00,CJRR99, ...]

More rigorous analyses
Propose sound assumptions
Instantiate constructions
...

This work: leakage-resilient PRFs

- Why PRFs (not PRPs)?
  - One of the most important primitives in symmetric cryptography (see Goldreich's book)
  - **Enough for encryption / authentication**
  - Needed for re-keying / init. of stream ciphers
  - Stateless primitive!
  - …

This work: leakage-resilient PRFs

- Why PRFs (not PRPs)?
  - One of the most important primitives in symmetric cryptography (see Goldreich's book)
  - **Enough for encryption / authentication**
  - Needed for re-keying / init. of stream ciphers
  - Stateless primitive!
  - …

- Main question: can leakage-resilient PRFs be
  - Secure (**super-exponential security**)?
  - Efficient (compared to other countermeasures)?

Secure – in what sense?

- Main focus so far: # of measurements
  - e.g. noise addition: # of measurements increases linearly with the noise variance
  - e.g. masking: # of measurements *may* increase exponentially with the number of masks
    - But requires hardware assumptions (e.g. leakage of shares must be independent)

Secure – in what sense?

- Main focus so far: # of measurements
  - e.g. noise addition: # of measurements increases linearly with the noise variance
  - e.g. masking: # of measurements *may* increase exponentially with the number of masks
    - But requires hardware assumptions (e.g. leakage of shares must be independent)

- Leakage-resilient PRFs approach:
  - Bound the data complexity by design
  - Try to guarantee high time complexity

Outline

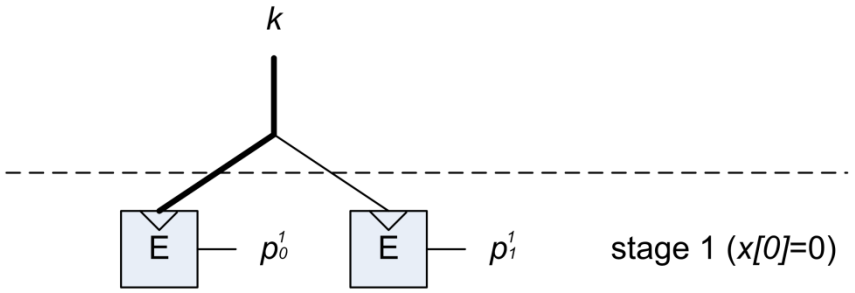
1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
4. Worst case analyses
5. Instantiation issues
6. Conclusions

Outline

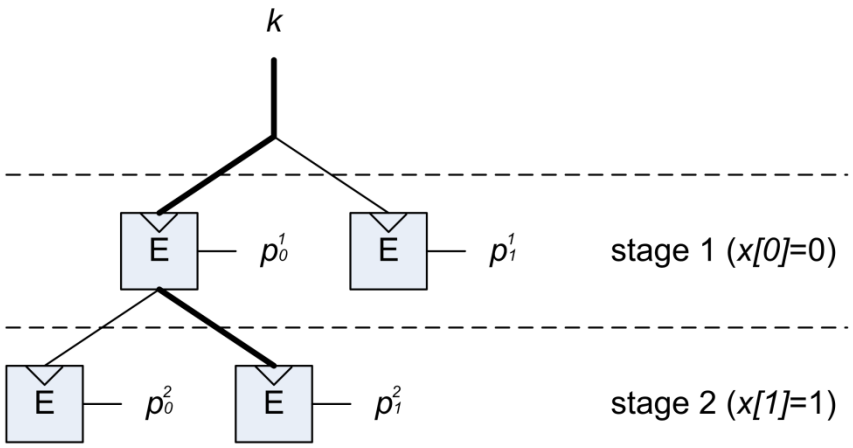
1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
4. Worst case analyses
5. Instantiation issues
6. Conclusions

Tree-based PRF (GGM 86)

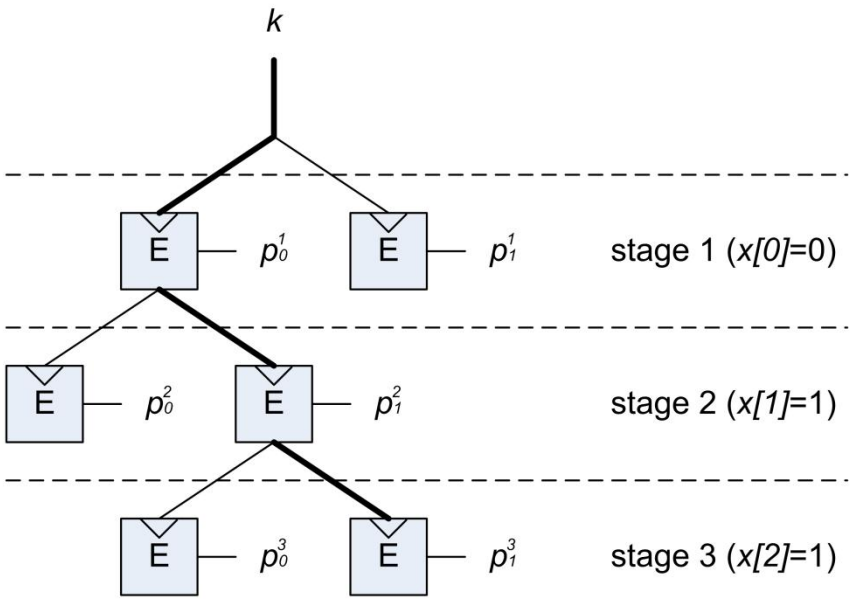
Tree-based PRF (GGM 86)



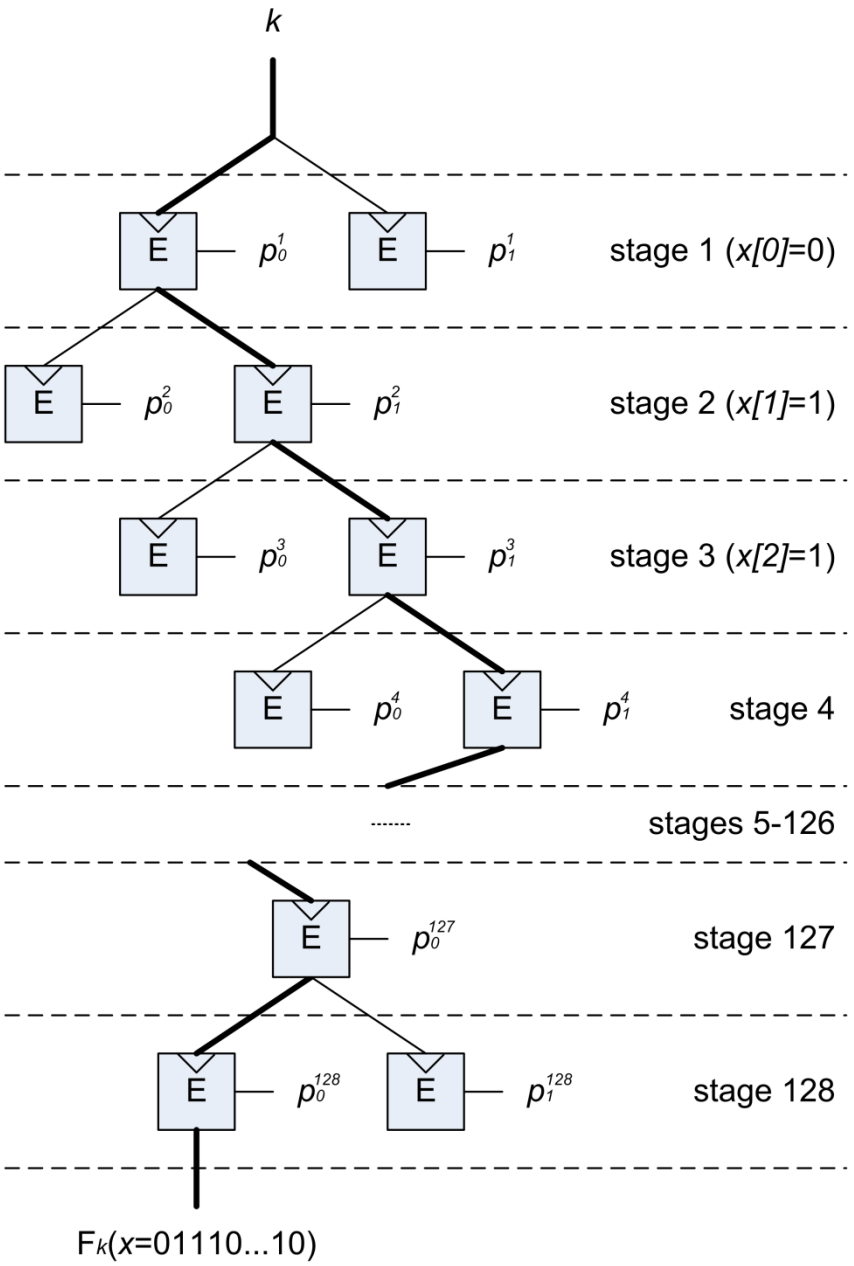
Tree-based PRF (GGM 86)



Tree-based PRF (GGM 86)



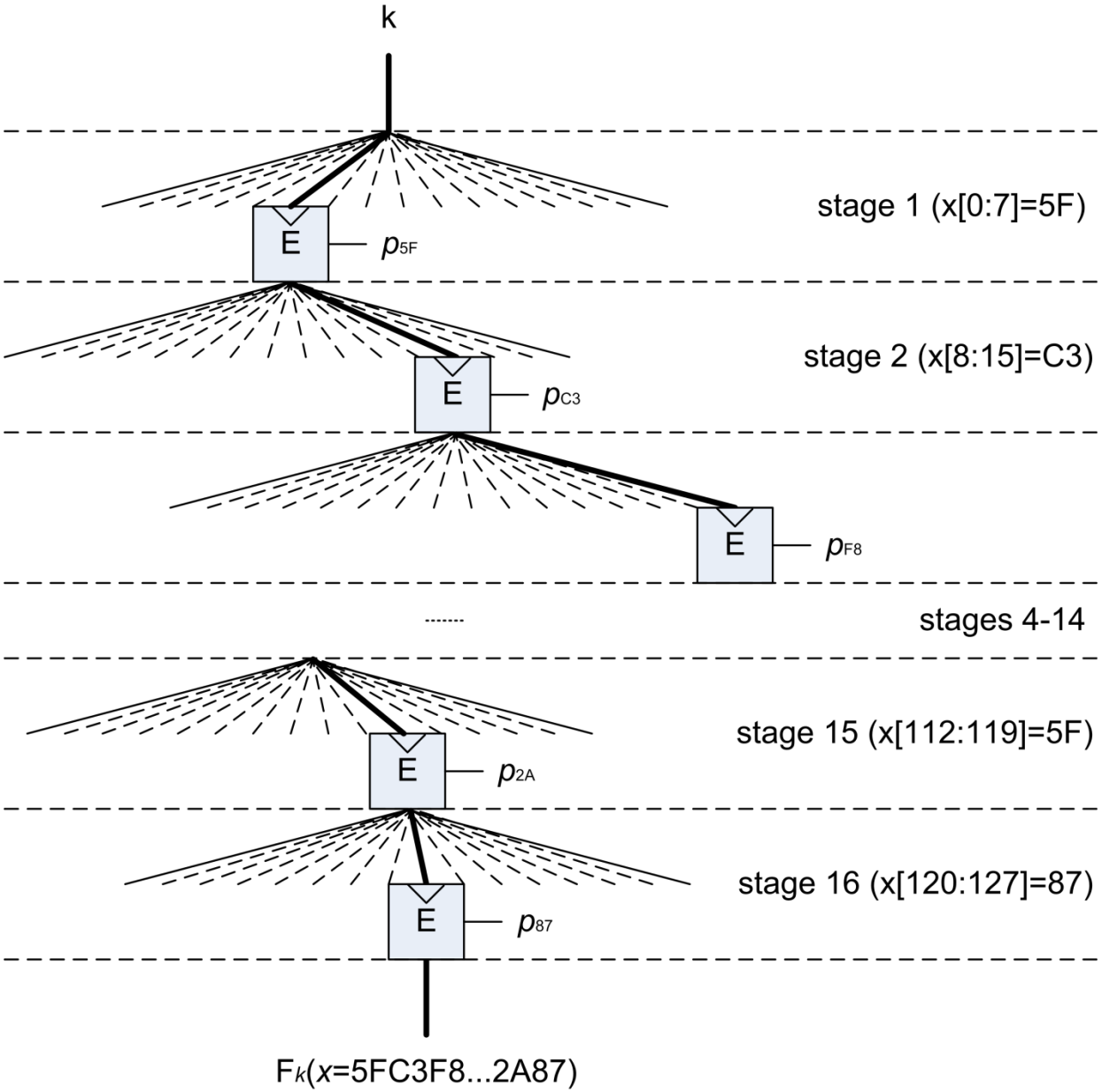
Tree-based PRF (GGM 86)



☺: 2-bounded data complexity

☹: 128 AES per 128-bit input

Efficiency / security tradeoff



☺: 16 AES per 128-bit input

😐: 256-bounded data complexity?

Outline

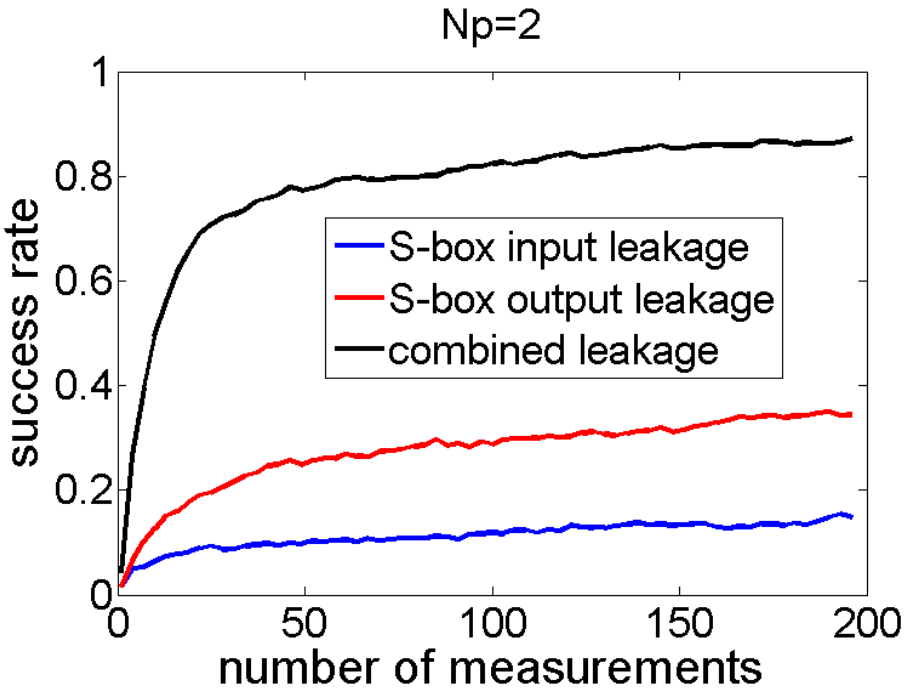
1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
4. Worst case analyses
5. Instantiation issues
6. Conclusions

Is bounded data complexity enough?

- Template attack against an 8-bit u-controller
- Success rate for the first AES S-box

Is bounded data complexity enough?

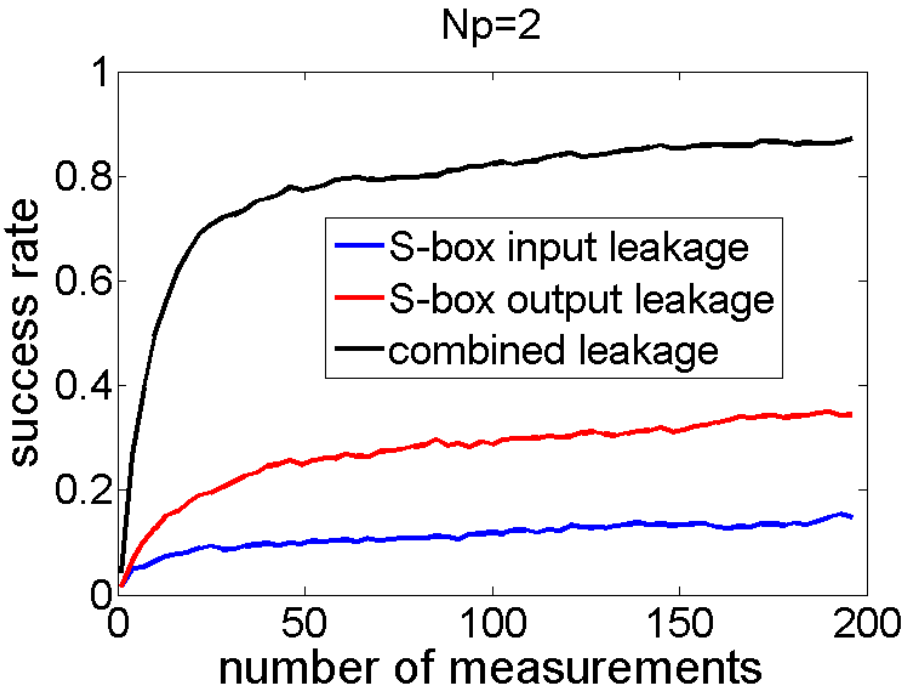
- Template attack against an 8-bit u-controller
- Success rate for the first AES S-box

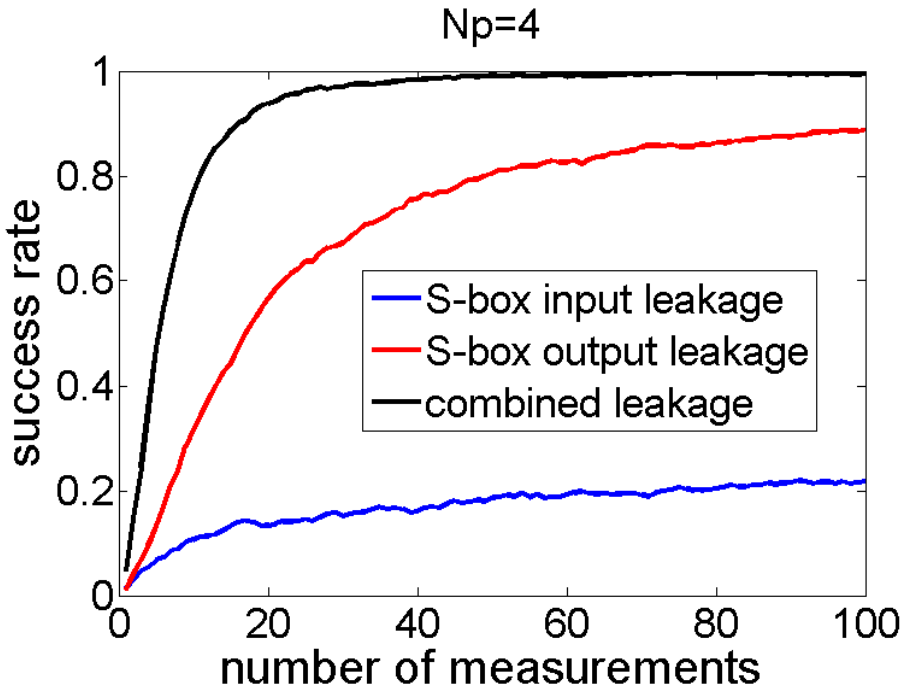


- High success rates already for Np=2 ☹

Is bounded data complexity enough?

- Template attack against an 8-bit u-controller
- Success rate for the first AES S-box





- High success rates already for Np=2 ☹

Possible security improvements

- Add countermeasures (masking, hiding, …)

Possible security improvements

- Add countermeasures (masking, hiding, …)
- Bound the number of measurements rather than the data complexity (i.e. prevent averaging)
  - e.g. store previous paths (but not efficient)

Possible security improvements

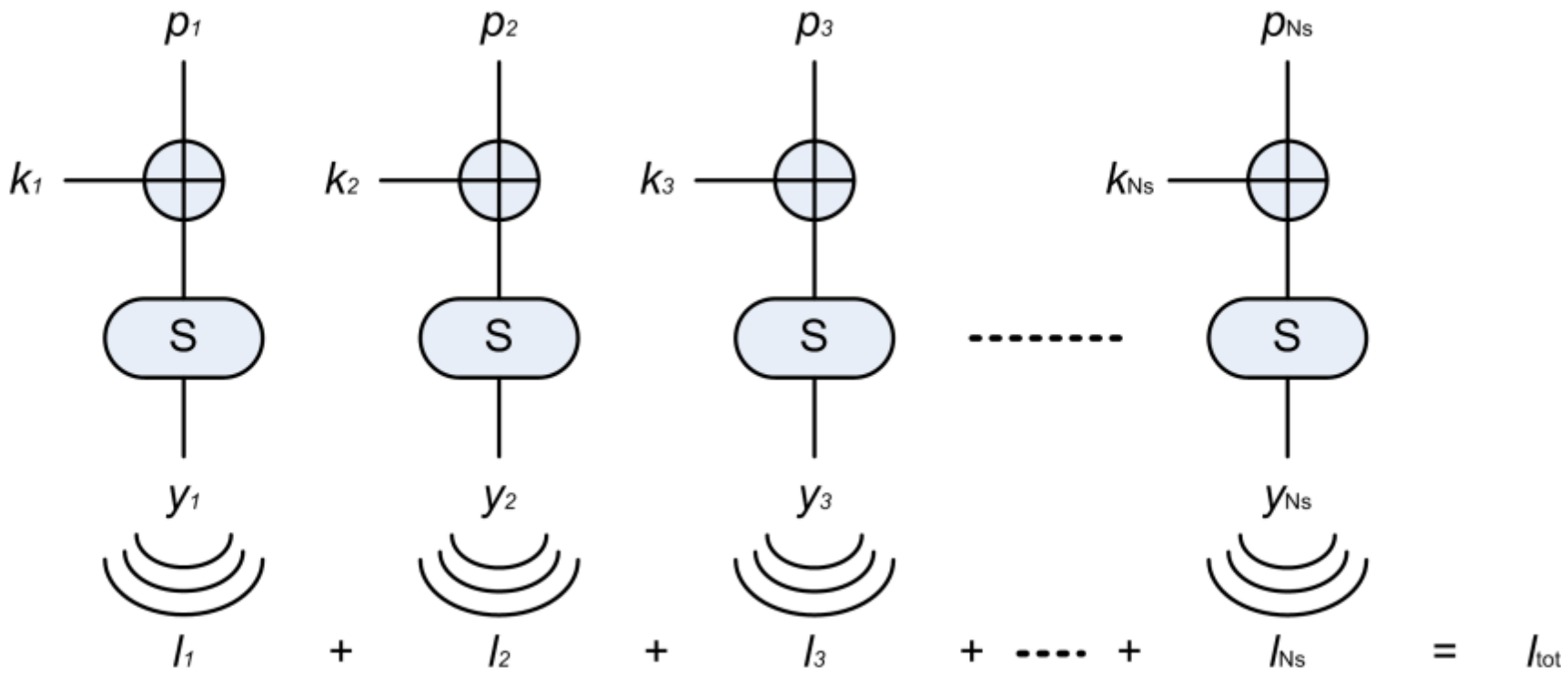
- Add countermeasures (masking, hiding, …)
- Bound the number of measurements rather than the data complexity (i.e. prevent averaging)
  - e.g. store previous paths (but not efficient)
- …

- Take advantage of algorithmic noise (parallelism)

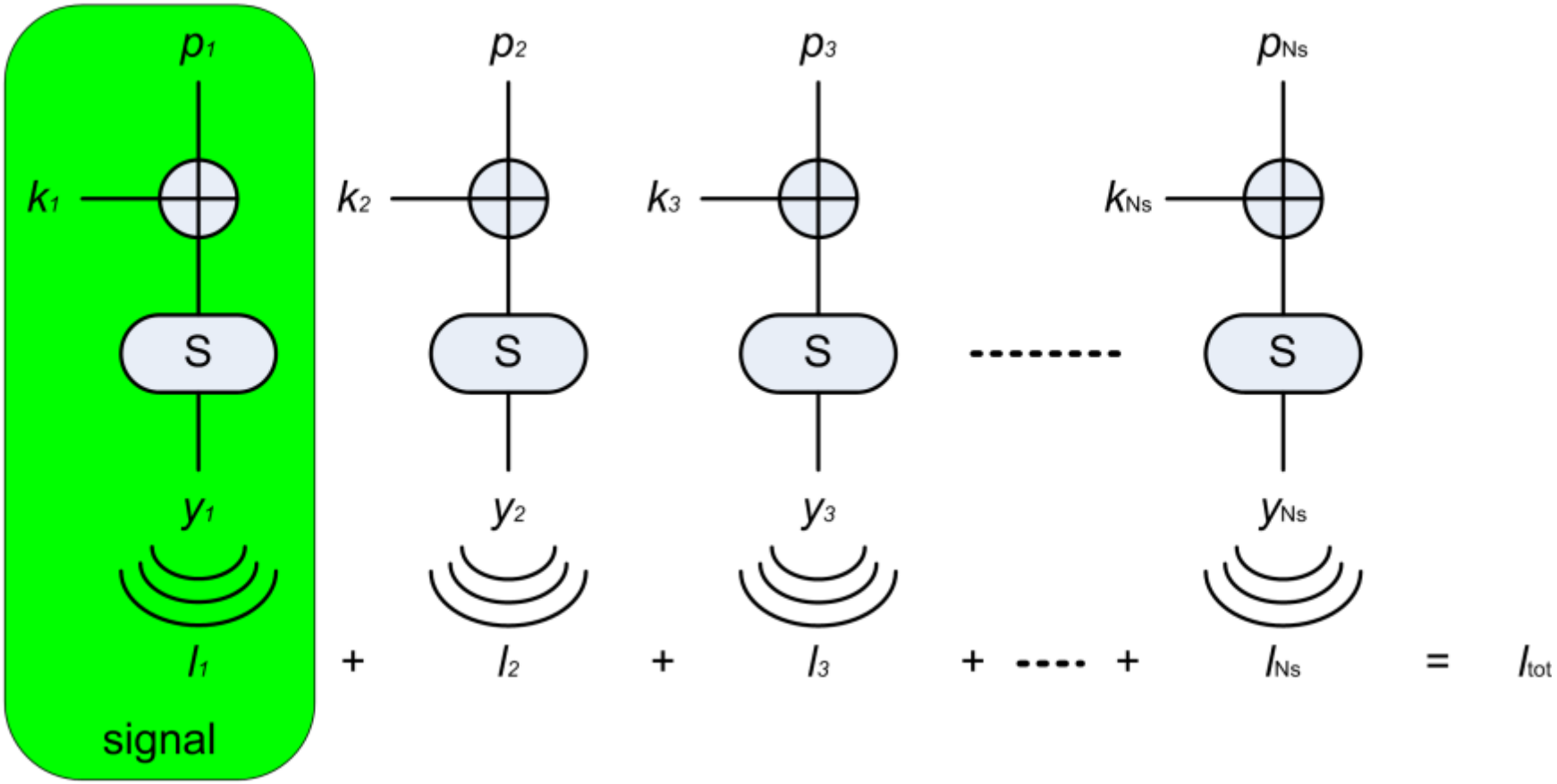
Outline

1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
   a. Previous leakage-resilient PRFs
   b. Our tweak: carefully chosen plaintexts
4. Worst case analyses
5. Instantiation issues
6. Conclusions

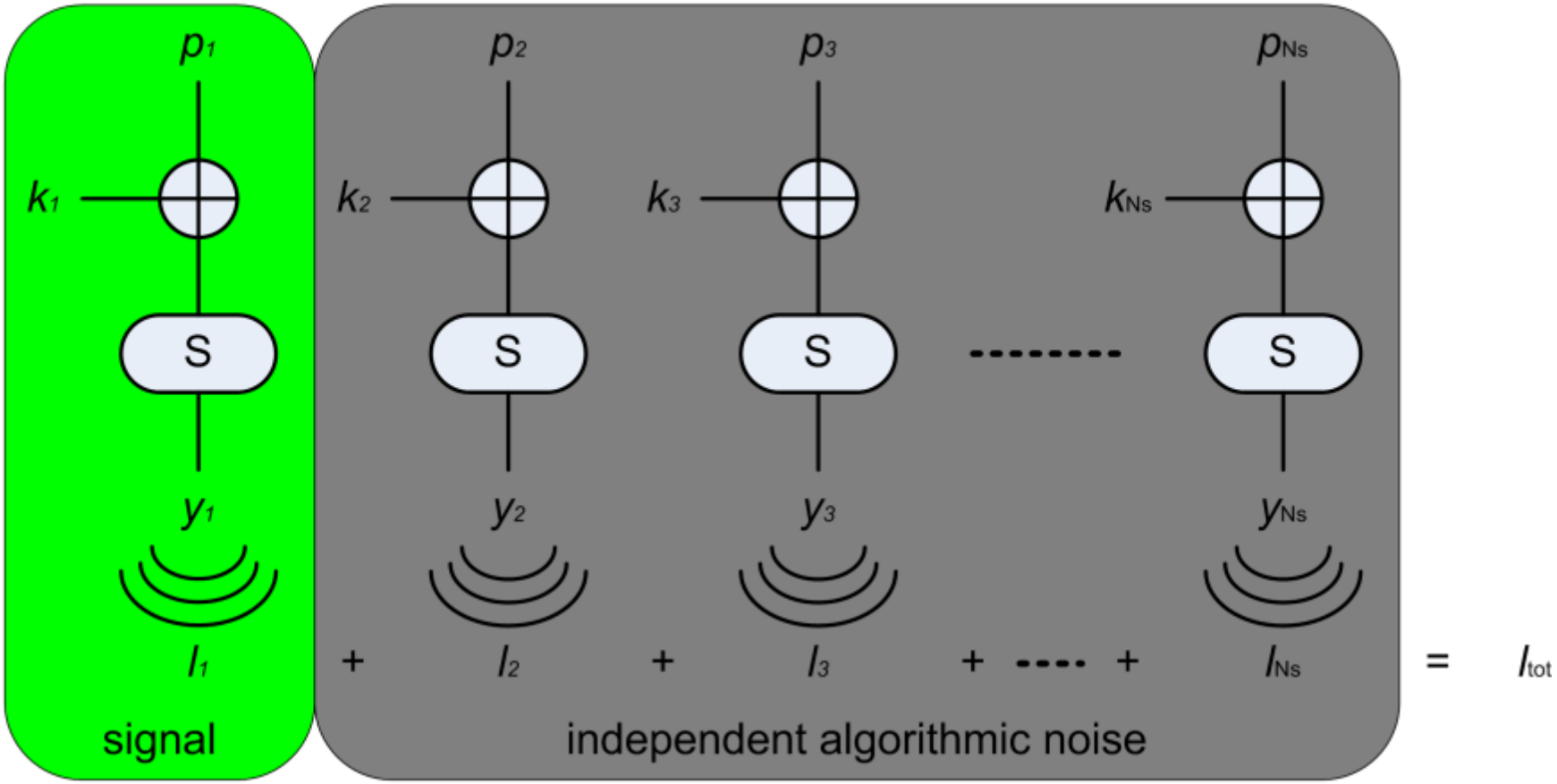
Algorithmic noise (standard DPA)



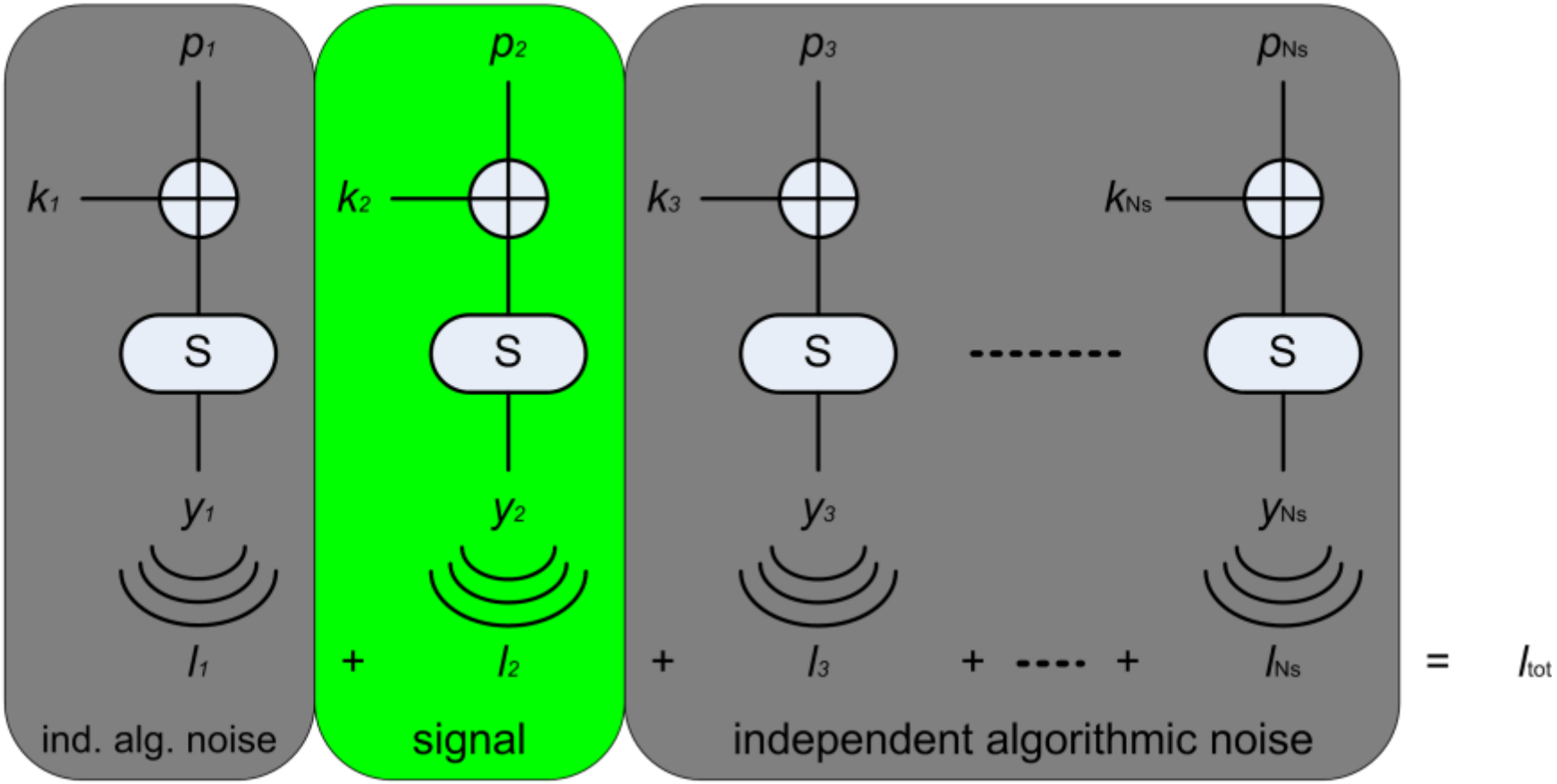
Algorithmic noise (standard DPA)



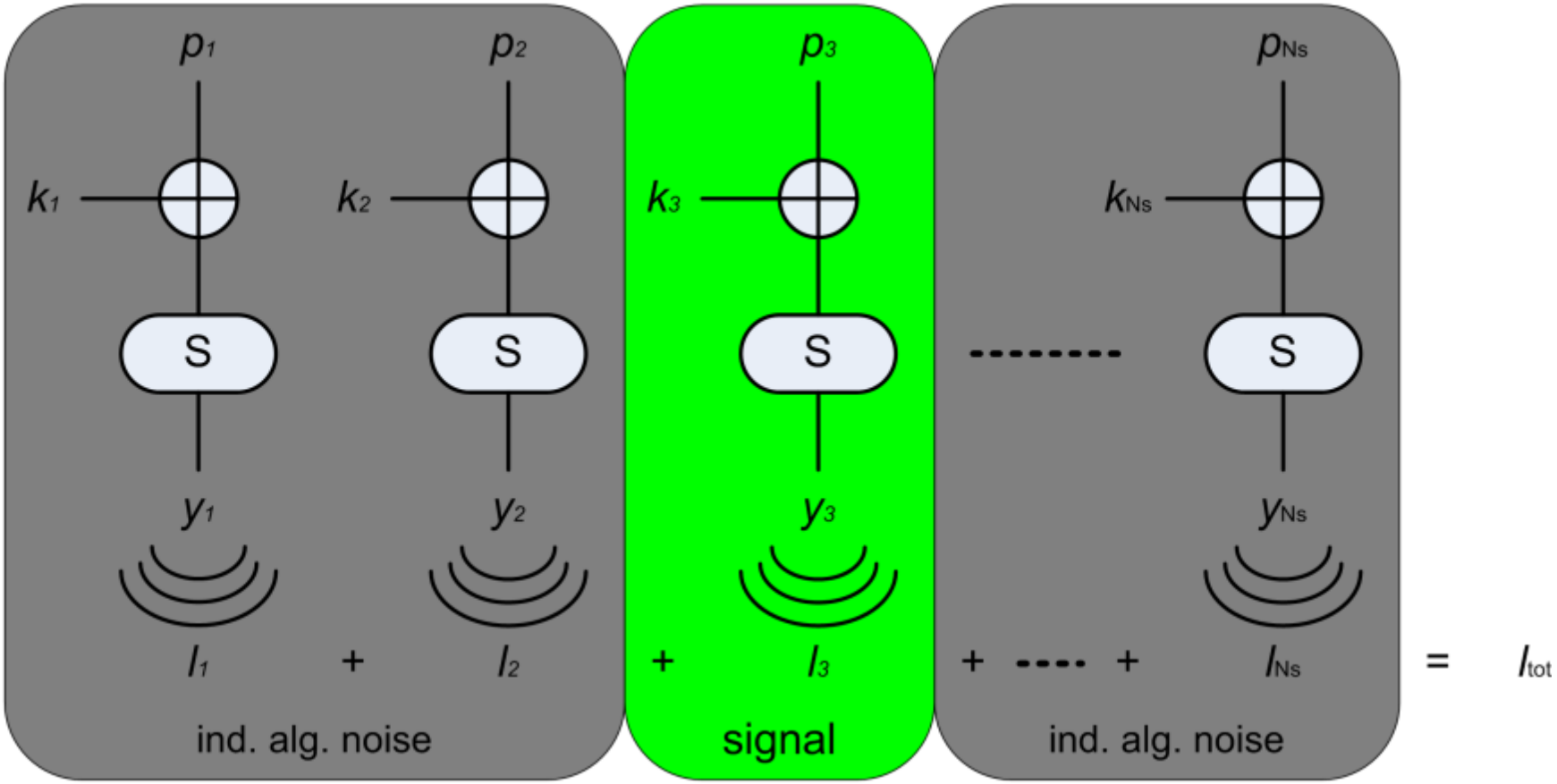
Algorithmic noise (standard DPA)



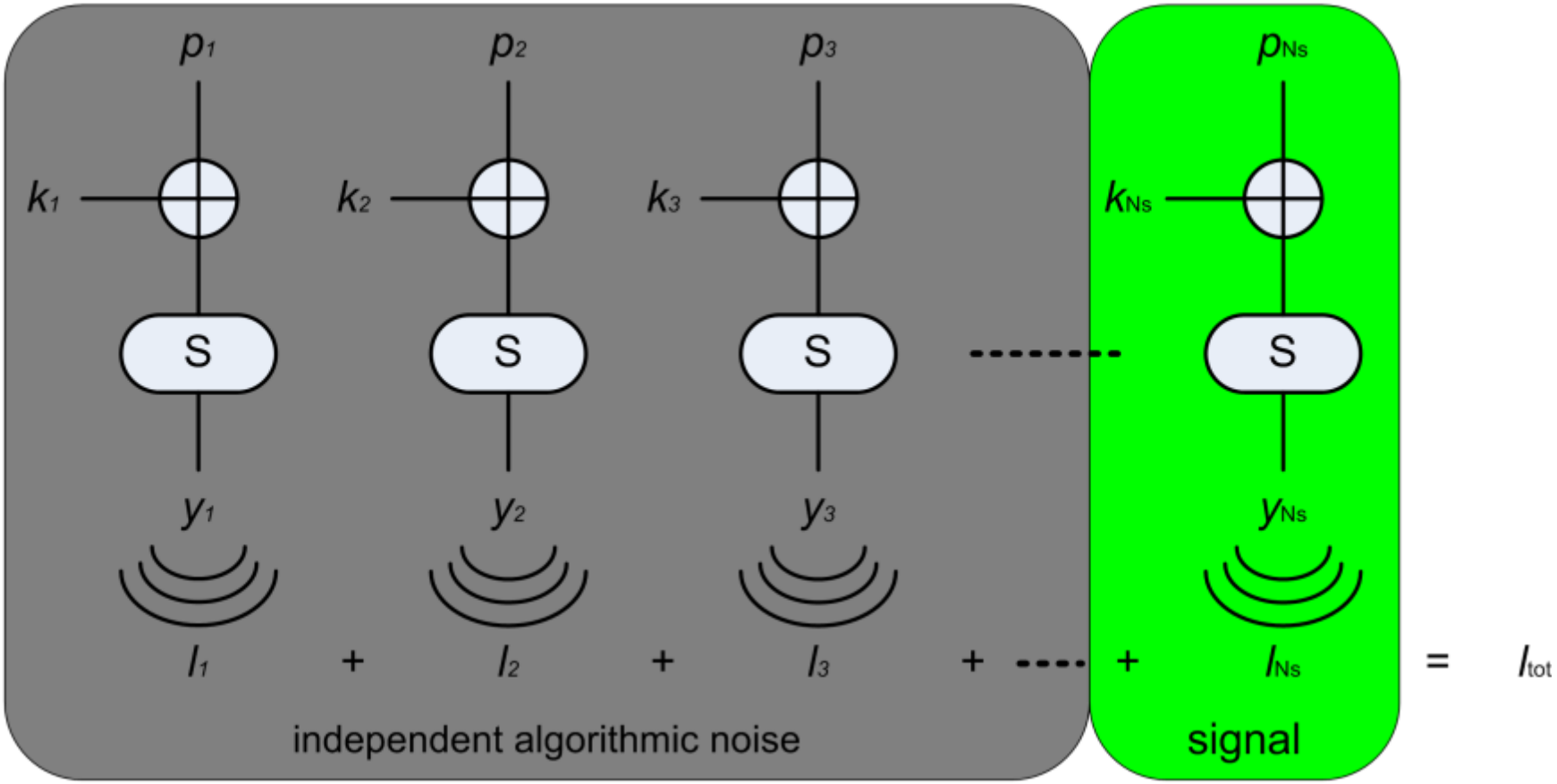
Random p_i's => divide & conquer attacks



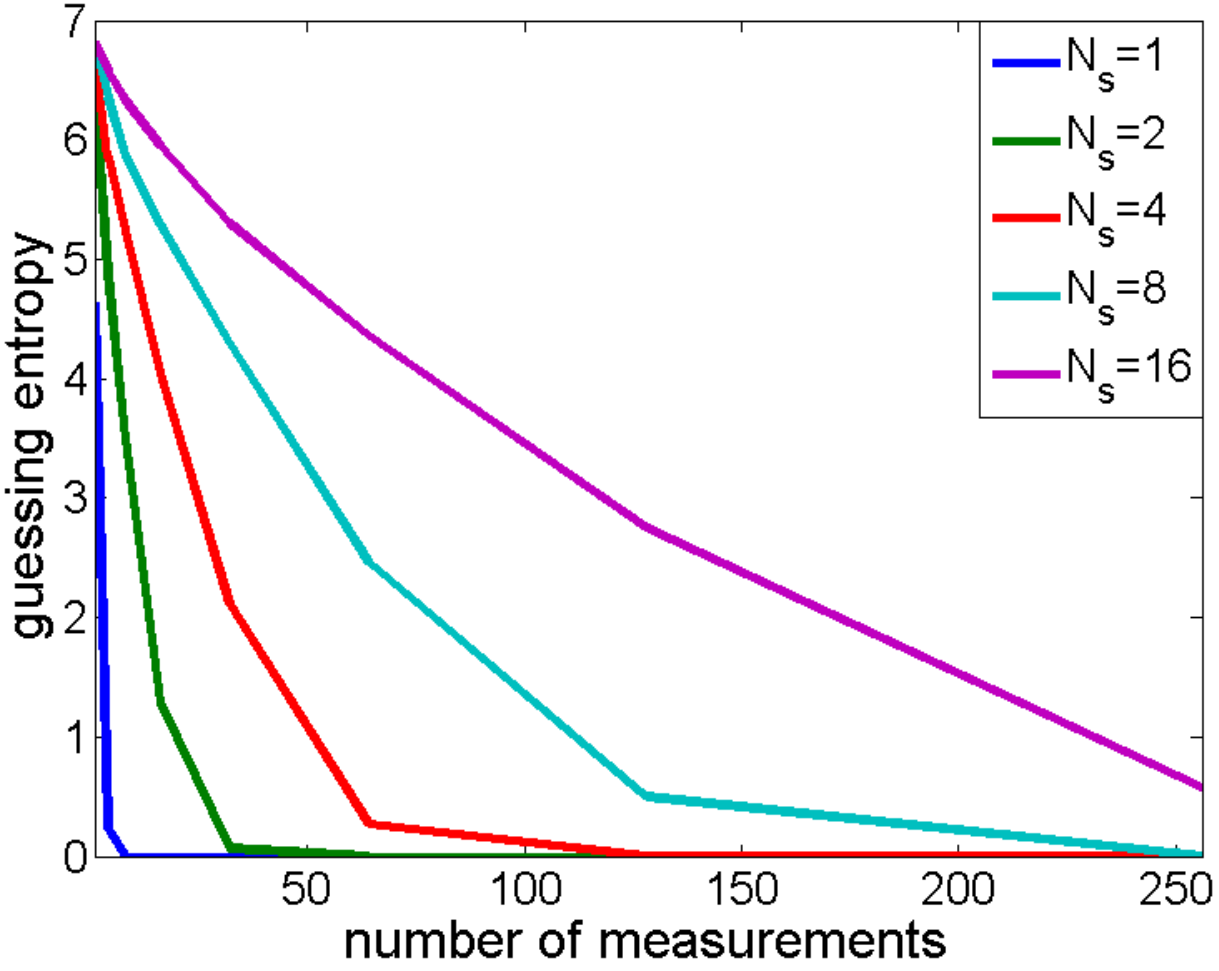
Random p_i's => divide & conquer attacks



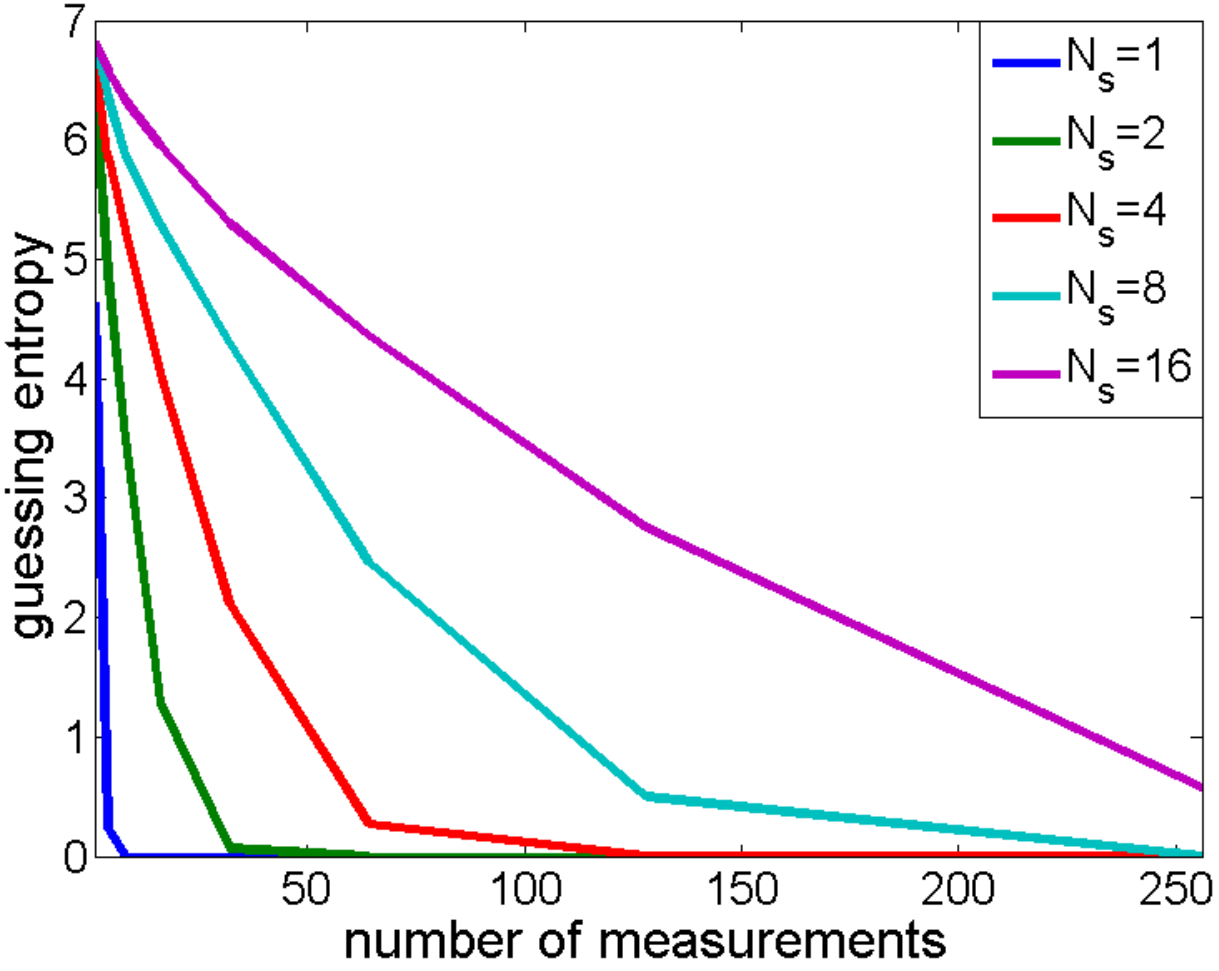
Random p_i's => divide & conquer attacks



Single S-box attack results



Single S-box attack results

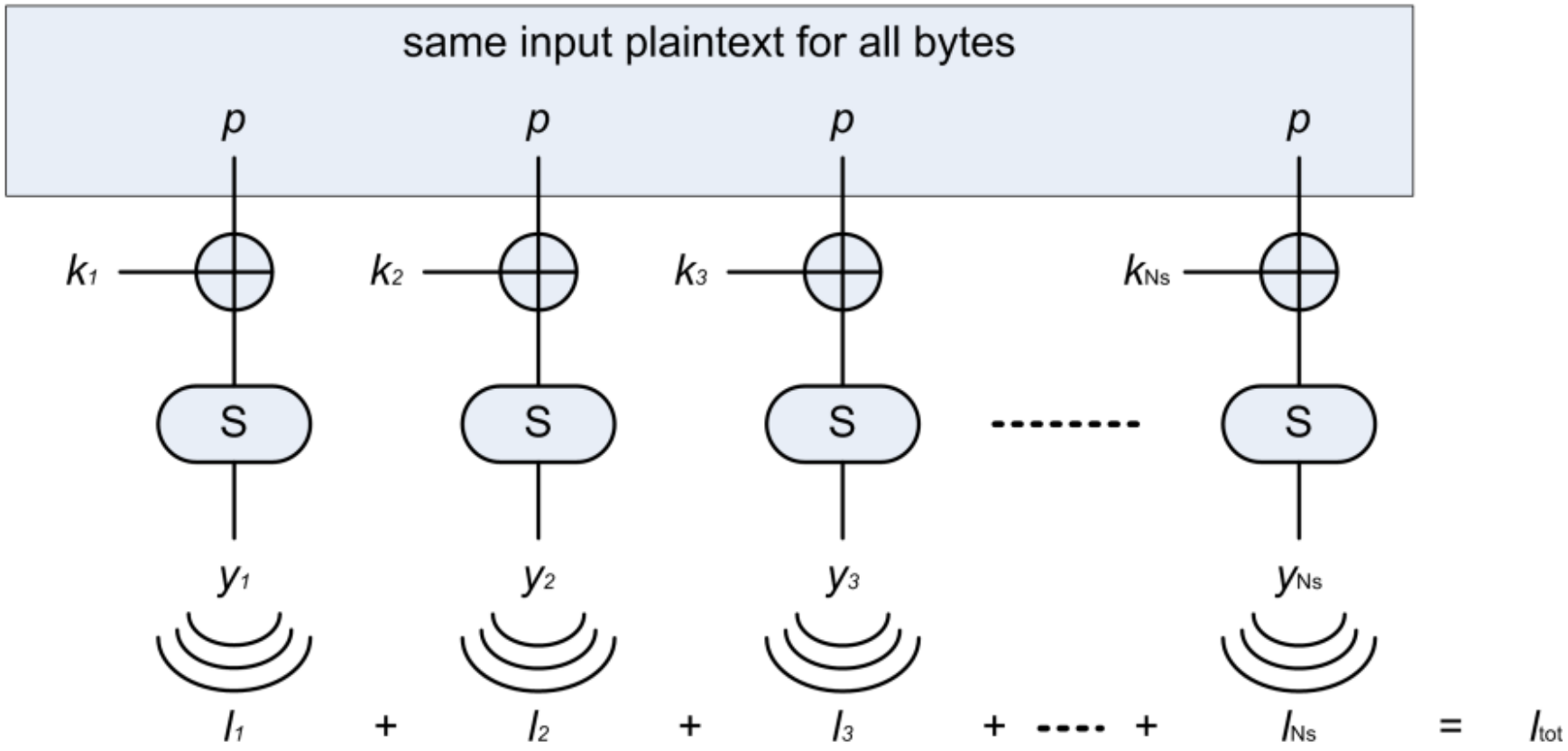


- Noise can be averaged by measuring more ☹

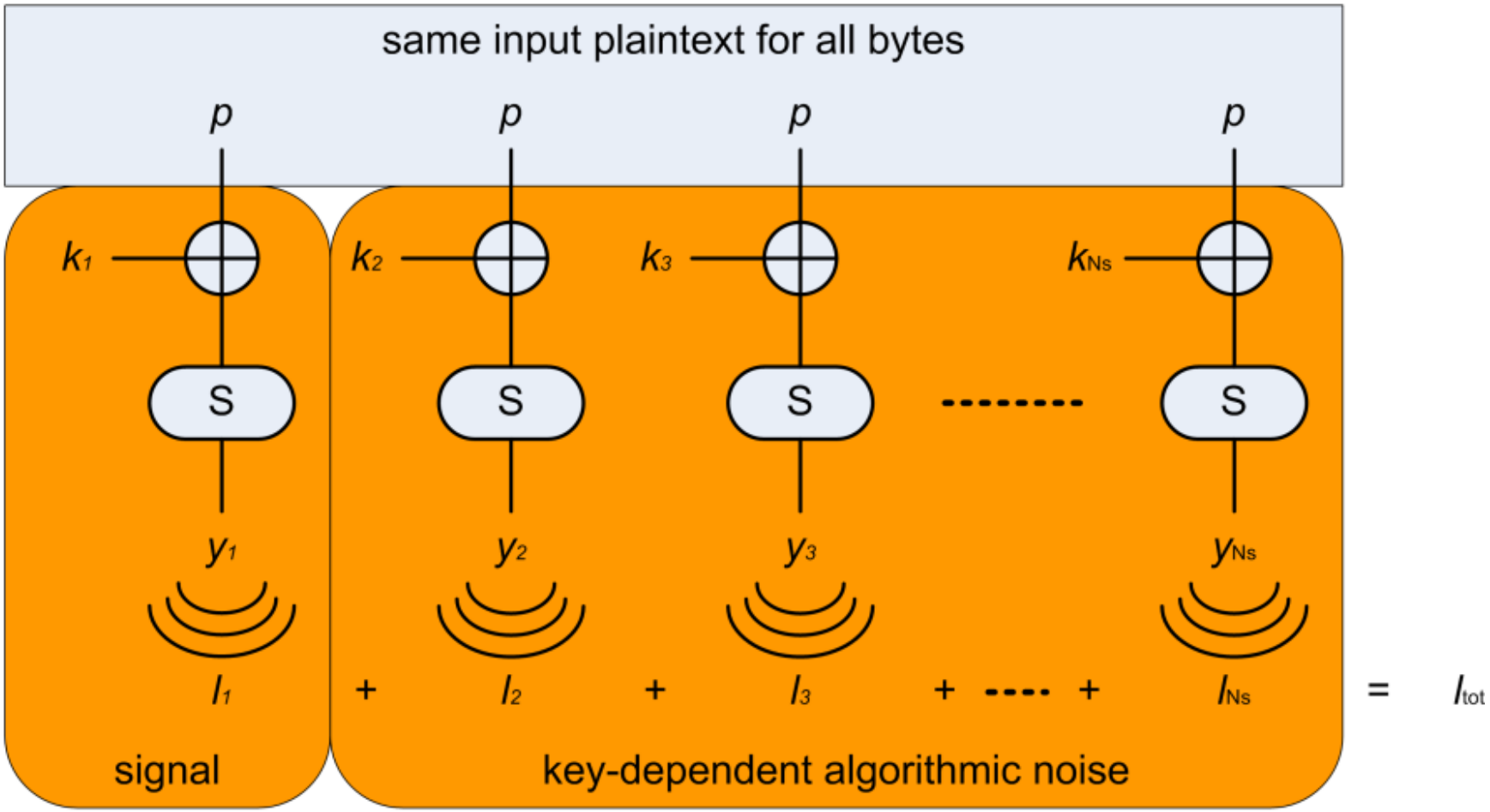
Outline

1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
   a. Previous leakage-resilient PRFs
   b. Our tweak: carefully chosen plaintexts
4. Worst case analyses
5. Instantiation issues
6. Conclusions

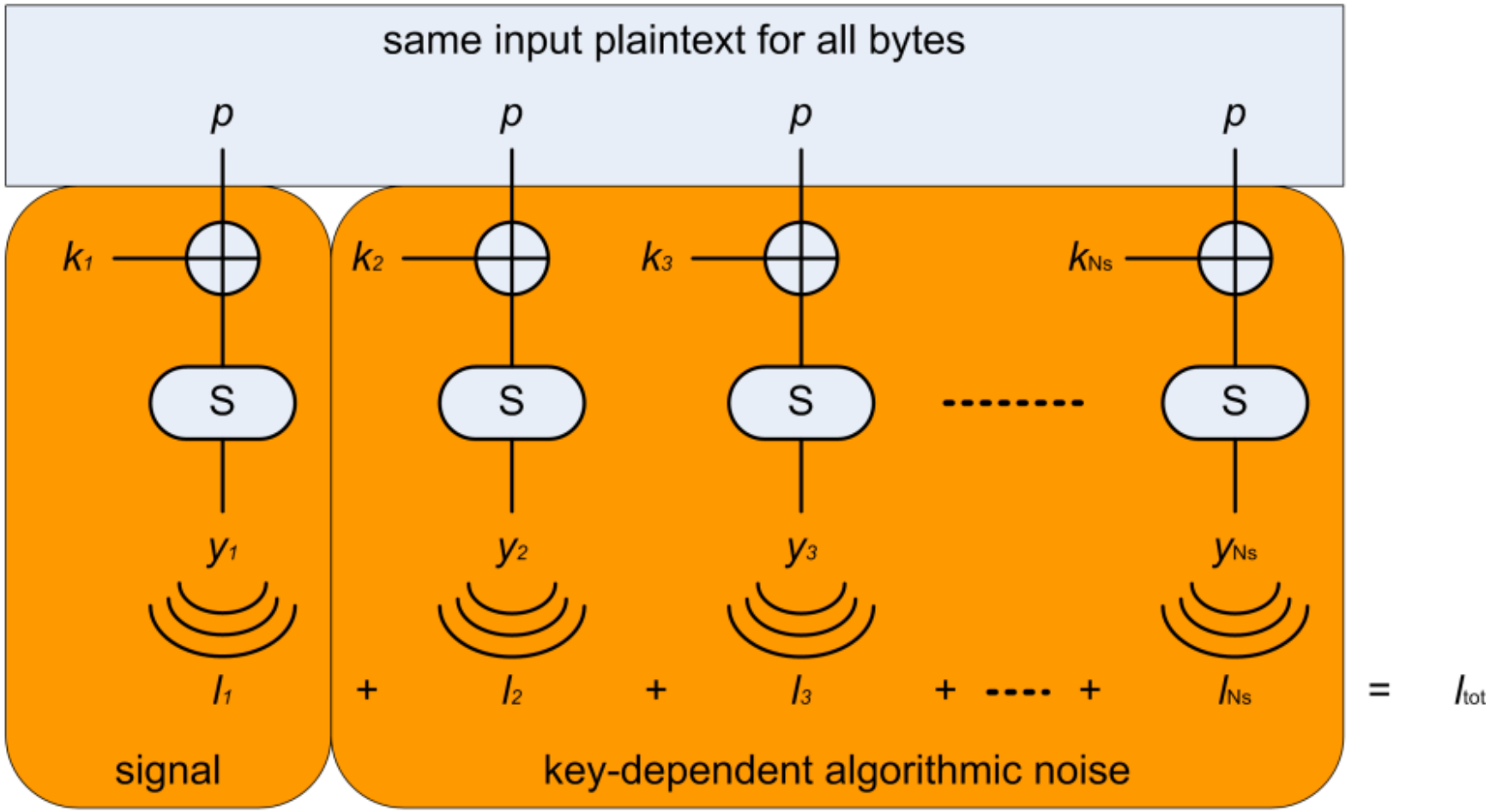
Our tweak: carefully chosen plaintexts (I)



Our tweak: carefully chosen plaintexts (I)



Our tweak: carefully chosen plaintexts (I)



e.g. CPA + HW model: same predictions for 16 key bytes

Our tweak: carefully chosen plaintexts (II)

- Intuition #1: algorithmic noise is key dependent
  => Divide & conquer attacks hardly apply

Our tweak: carefully chosen plaintexts (II)

- Intuition #1: algorithmic noise is key dependent
  => Divide & conquer attacks hardly apply

- Intuition #2: assume the leakage functions are (roughly) identical for all S-boxes
  - Then the models in standard DPA attacks are also identical for all S-boxes

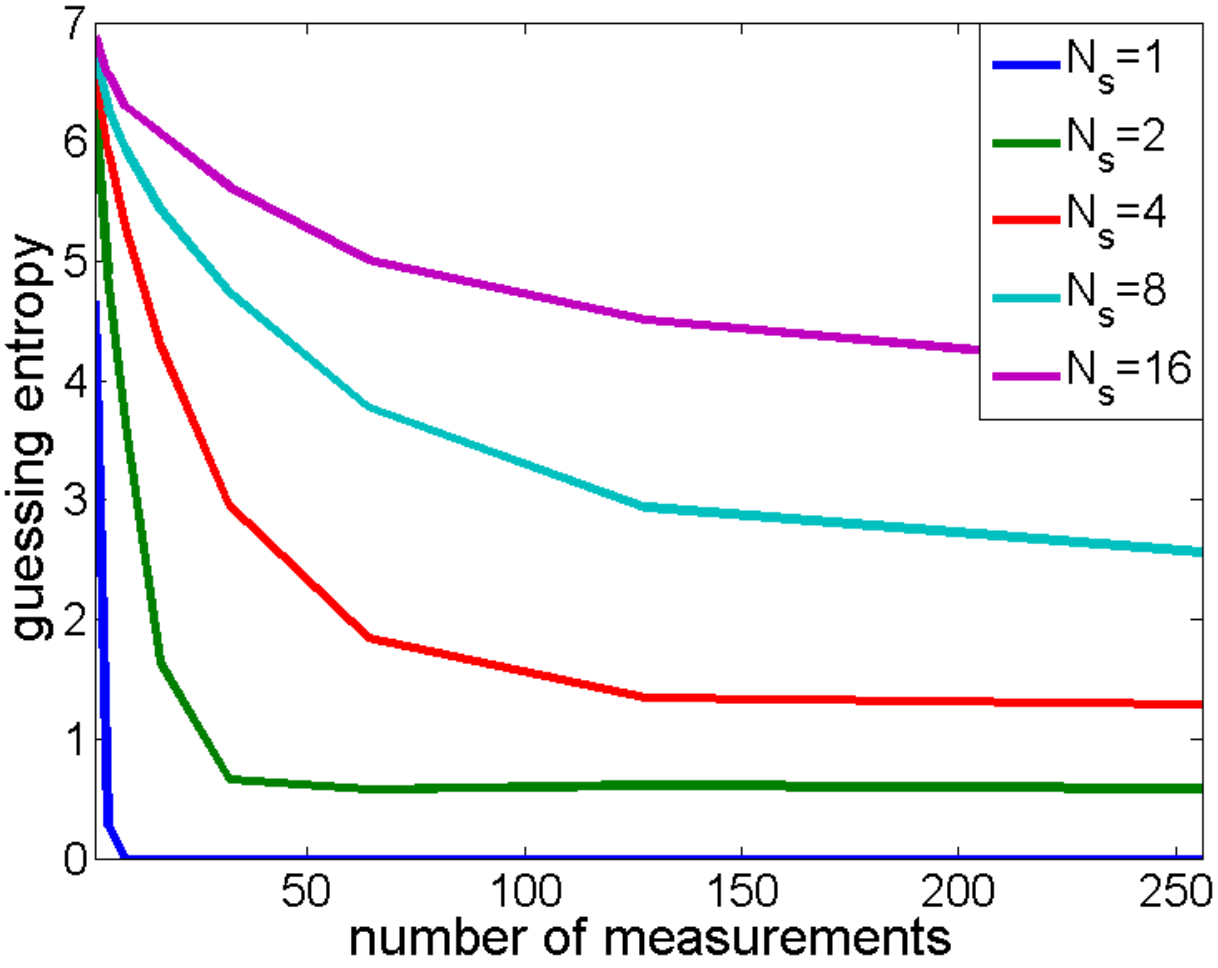
Our tweak: carefully chosen plaintexts (II)

- Intuition #1: algorithmic noise is key dependent
  => Divide & conquer attacks hardly apply

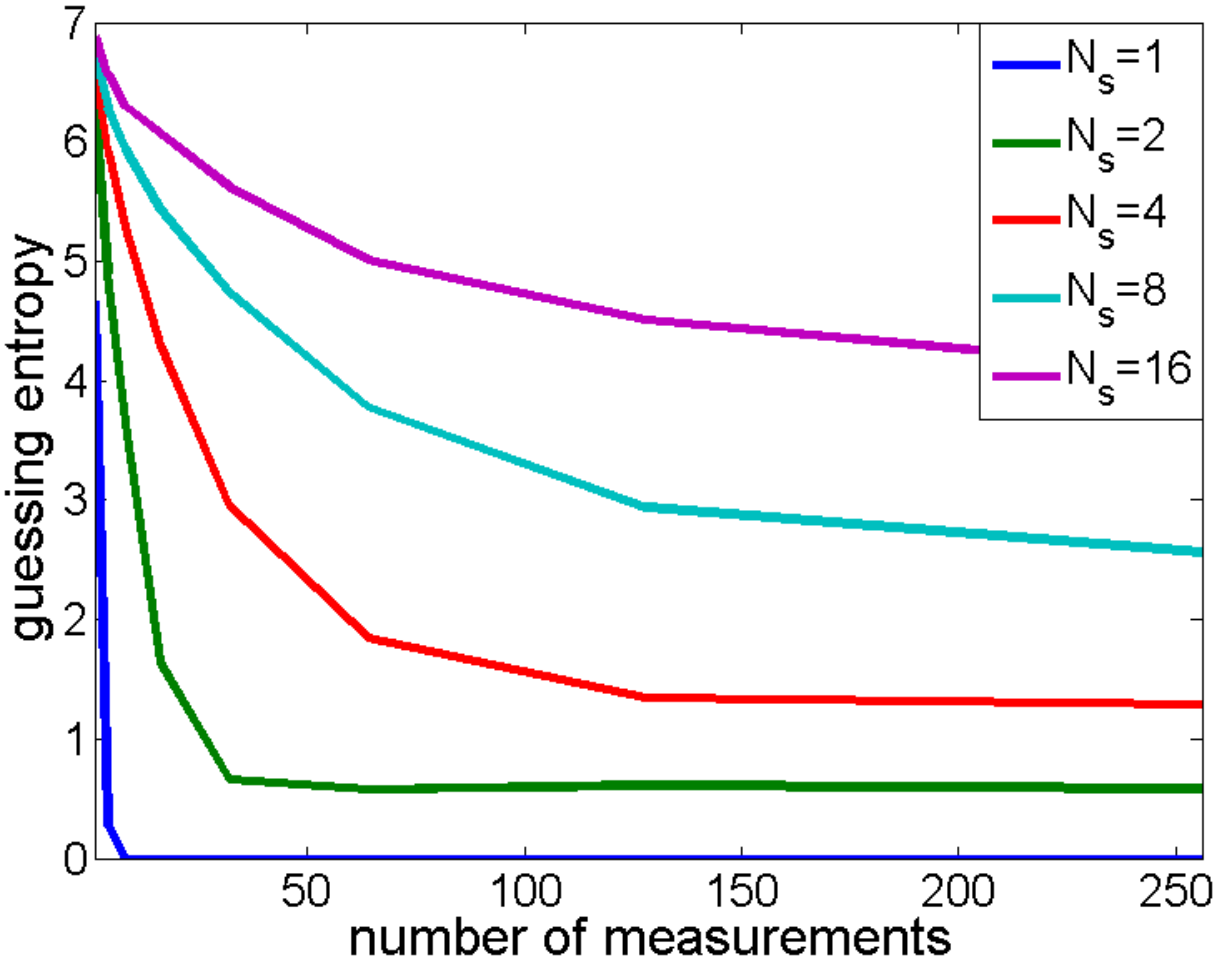
- Intuition #2: assume the leakage functions are (roughly) identical for all S-boxes
  - Then the models in standard DPA attacks are also identical for all S-boxes

  - Even in the (unlikely) situation where the *Ns* key bytes are rated in the first *Ns* positions by DPA, it remains to enumerate *Ns*! Permutations
    - e.g. 16!=2^44, 24!=2^79, 32!=2^117

Single S-box attack results



Single S-box attack results



- Even with 256 meas., noise cannot be averaged ☺

Outline

1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
   a. Previous leakage-resilient PRFs
   b. Our tweak: carefully chosen plaintexts
4. Worst case analyses
5. Instantiation issues
6. Conclusions

Worst case security evaluations (I)

- Standard DPA attacks do not appear very relevant to analyze the security of our tweaked design

=> We considered two alternatives considering noiseless traces as a first-step investigation

Worst case security evaluations (I)

- Standard DPA attacks do not appear very relevant to analyze the security of our tweaked design

=> We considered two alternatives considering noiseless traces as a first-step investigation

1. Iterative DPA-like attack
   - For i=1:*Ns*
     - Perform a DPA and keep best-rated key
     - Remove the hypothetical leakage of this key from the actual leakage traces

Worst case security evaluations (II)

2. Lattice-based attacks:

$$l_j = \sum_{i=1}^{N_s} \mathsf{L}(\mathsf{S}(p_j[i] \oplus k[i]))$$

- Recovering *Ns* key bytes satisfying this relation for *Np* plaintexts is a vectorial knapsack problem
  => We used LLL as a black box for solving it

Worst case security evaluations (II)

2. Lattice-based attacks:

$$l_j = \sum_{i=1}^{N_s} \mathsf{L}(\mathsf{S}(p_j[i] \oplus k[i]))$$

- Recovering *Ns* key bytes satisfying this relation for *Np* plaintexts is a vectorial knapsack problem
  => We used LLL as a black box for solving it

	$N_p = 256$	254	252	251	250	249	248	247	246	245
$N_s = 16$	100	100	100	100	100	100	100	100	100	100
	$1.3s$	$1.4s$	$1.4s$	$1.4s$	$1.5s$	$1.5s$	$3.1s$	$34.8s$	$73.0s$	$131.4s$
24	99.9	100	100	100	100	100	100	100	TBD	TBD
	$1.4s$	$1.4s$	$1.4s$	$1.4s$	$1.5s$	$1.5s$	$3.1s$	$35.5s$	$\approx 88s$	$\approx 143s$
32	79.6	79	79	83	80	79	76	TBD	TBD	TBD
	$1.4s$	$1.5s$	$1.5s$	$1.5s$	$1.6s$	$1.6s$	$3.3s$	$\approx 33s$	$\approx 81s$	$\approx 140s$

Outline

1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
   a. Previous leakage-resilient PRFs
   b. Our tweak: carefully chosen plaintexts
4. Worst case analyses
5. Instantiation issues
6. Conclusions

Main question

- Do different S-boxes leak the same?

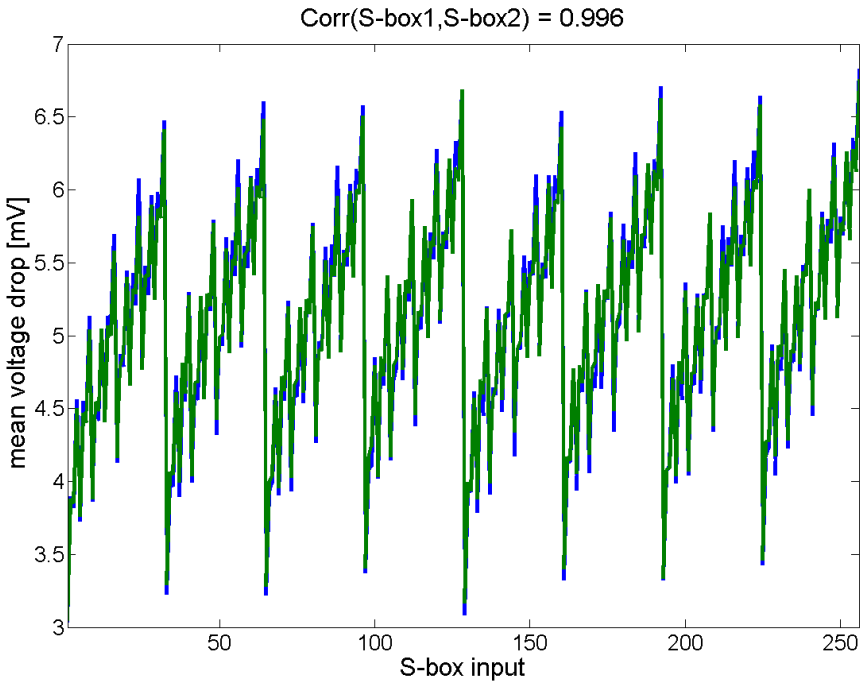
Main question

- Do different S-boxes leak the same?
- FPGA case study with two types of S-boxes

Main question

- Do different S-boxes leak the same?

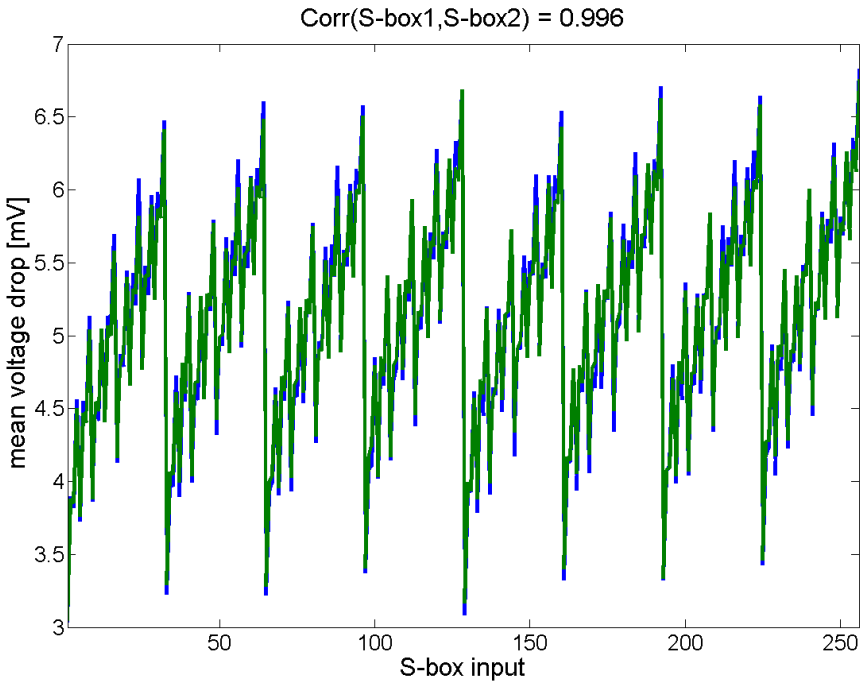
- FPGA case study with two types of S-boxes
  - Using the RAM blocks of modern FPGAs

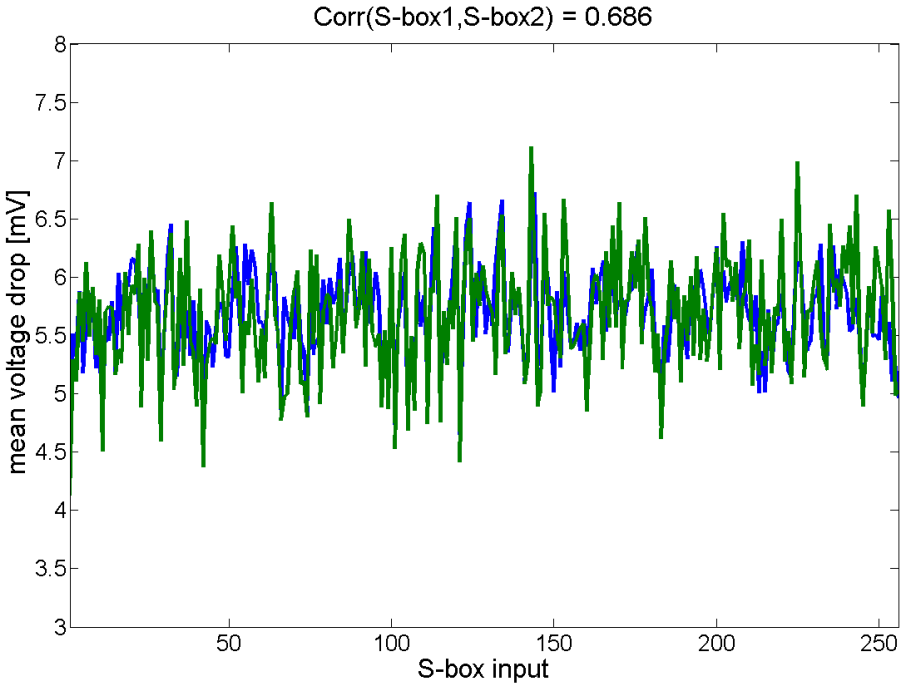


Main question

- Do different S-boxes leak the same?

- FPGA case study with two types of S-boxes
  - Using the RAM blocks of modern FPGAs
  - Combinatorial (from Canright, CHES 2005)



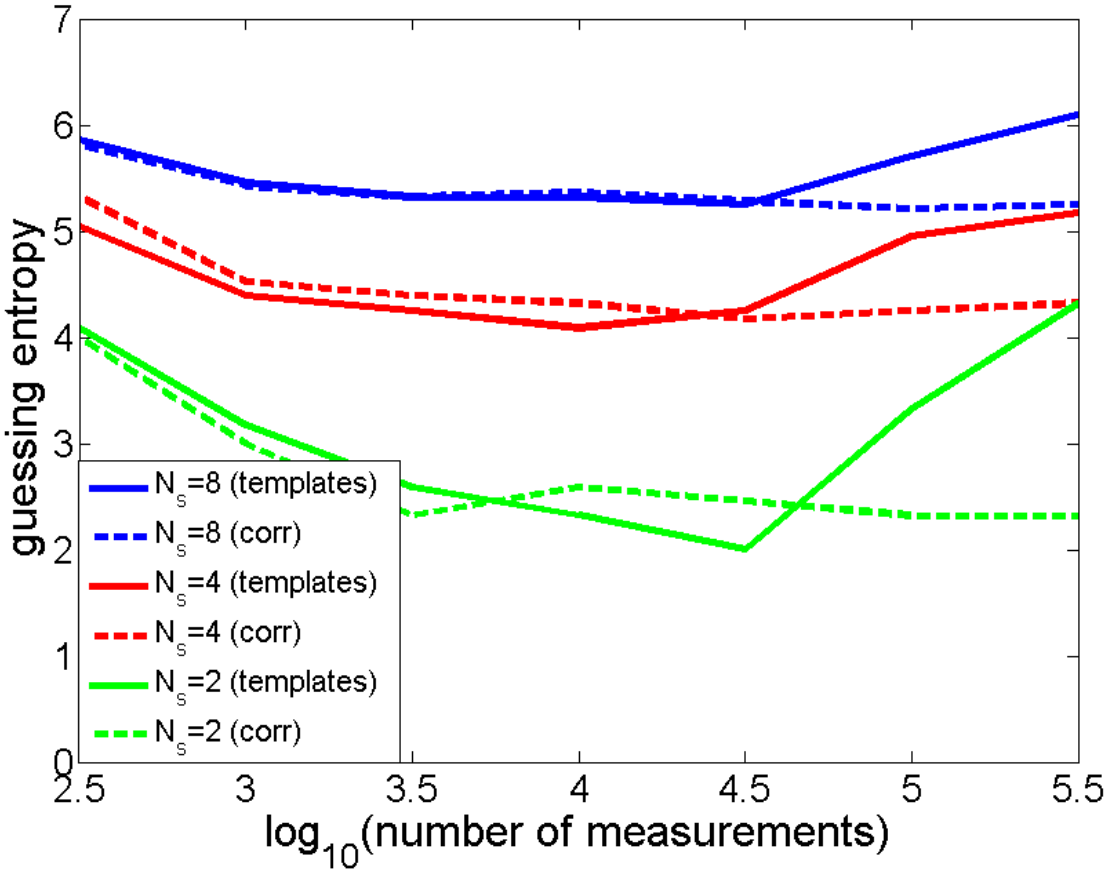


Can we exploit different leakage models?

- Case study using the Canright S-boxes
  - Template attacks, correlation attacks
  - Both using the *Ns* different models

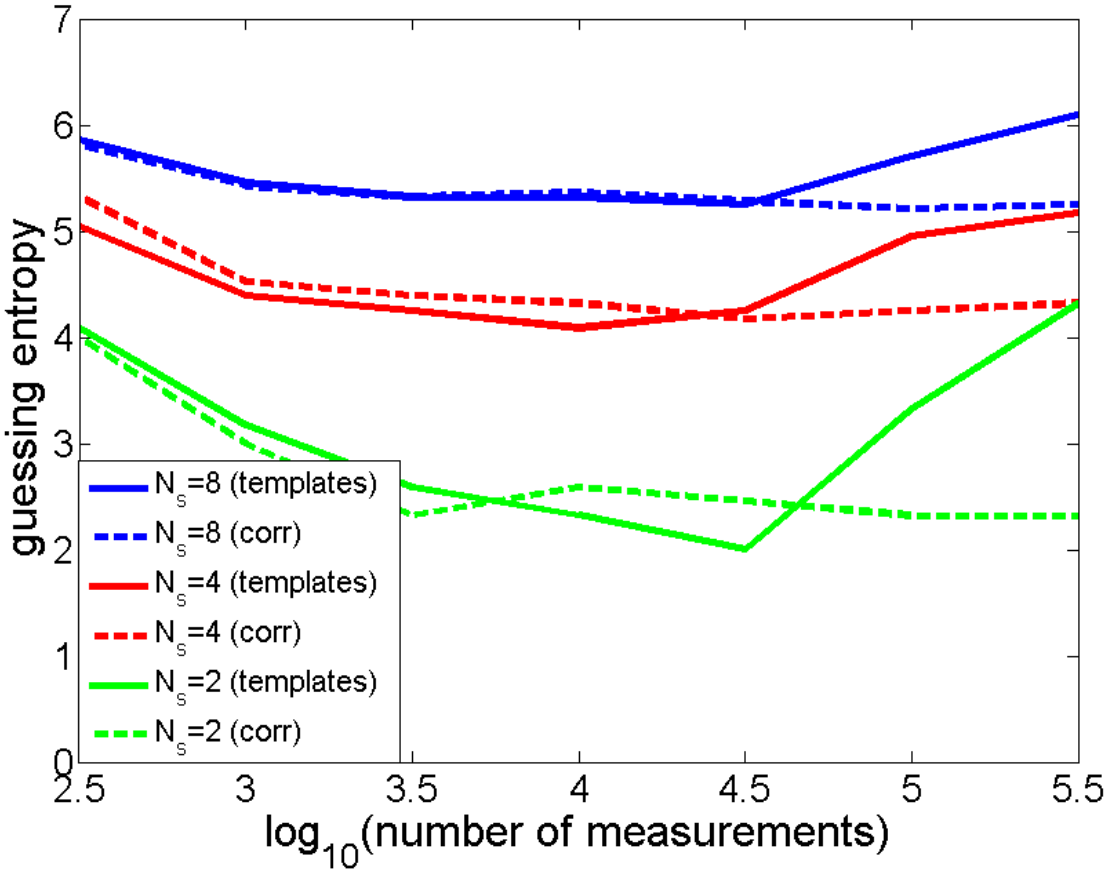
Can we exploit different leakage models?

- Case study using the Canright S-boxes
  - Template attacks, correlation attacks
  - Both using the *Ns* different models



Can we exploit different leakage models?

- Case study using the Canright S-boxes
  - Template attacks, correlation attacks
  - Both using the *Ns* different models



Main message:
the key-dependent algorithmic noise remains hard to exploit

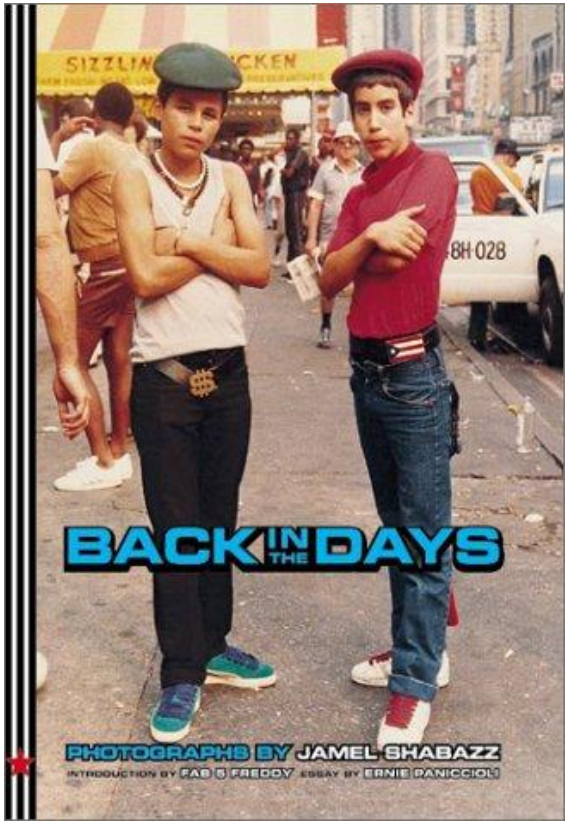
☺

Outline

1. Tree-based PRF (GGM 86)
2. Is bounded data complexity enough?
3. Efficiently exploiting parallelism
   a. Previous leakage-resilient PRFs
   b. Our tweak: carefully chosen plaintexts
4. Worst case analyses
5. Instantiation issues
6. Conclusions

Conclusions (I)

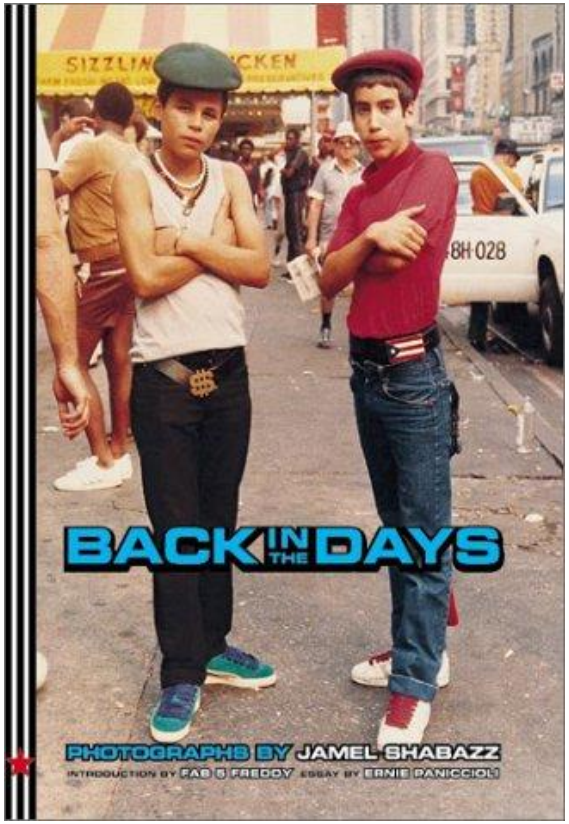
Remember back in the days…



Conclusions (I)

Remember back in the days…

We thought masking was “secure”

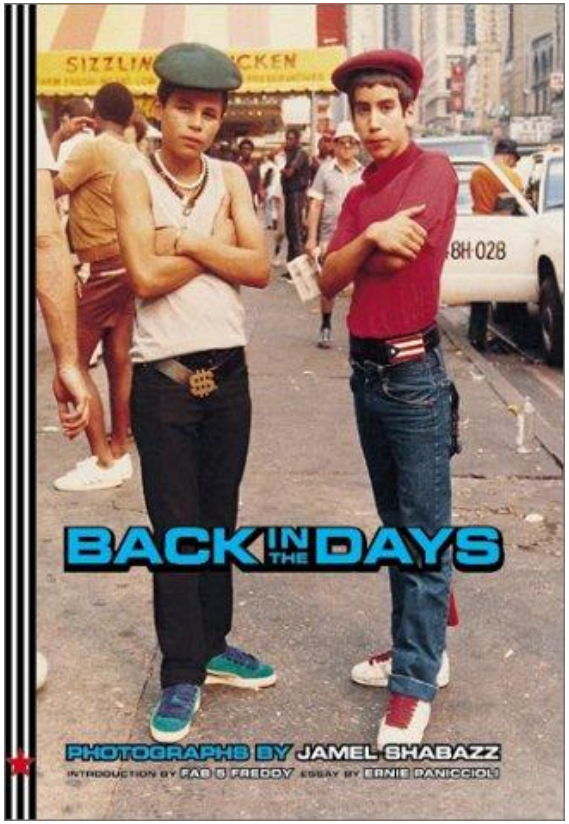


Conclusions (I)

Remember back in the days…

We thought masking was “secure”

Then came HO attacks,



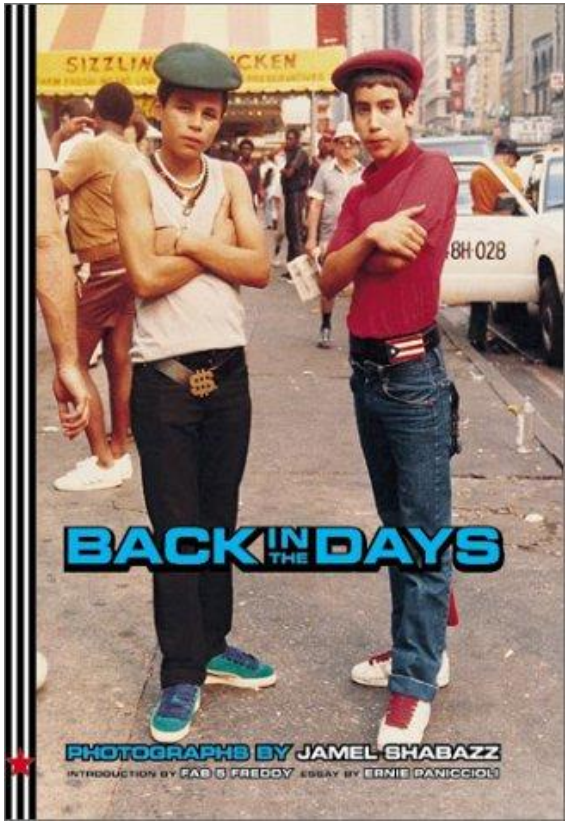
Conclusions (I)

Remember back in the days…

We thought masking was “secure”

Then came HO attacks,

Then came glitches,



Conclusions (I)

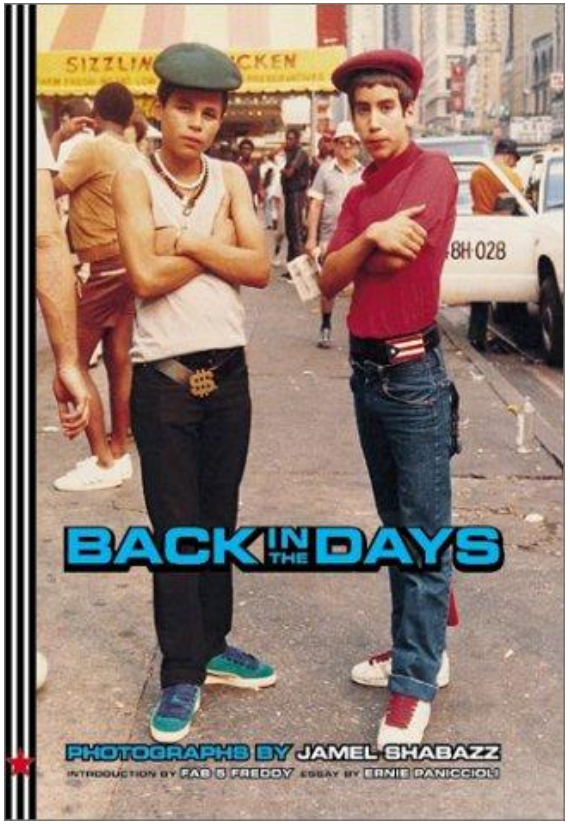
Remember back in the days…

We thought masking was “secure”

Then came HO attacks,

Then came glitches,

Then came early propagation,

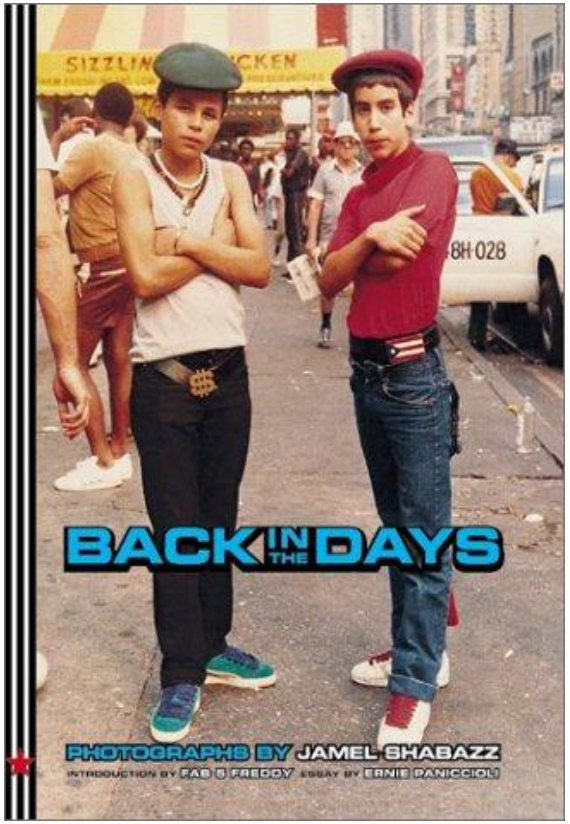


Conclusions (I)

Remember back in the days…

We thought masking was “secure”

Then came HO attacks,
Then came glitches,
Then came early propagation,
Then came coupling,
…

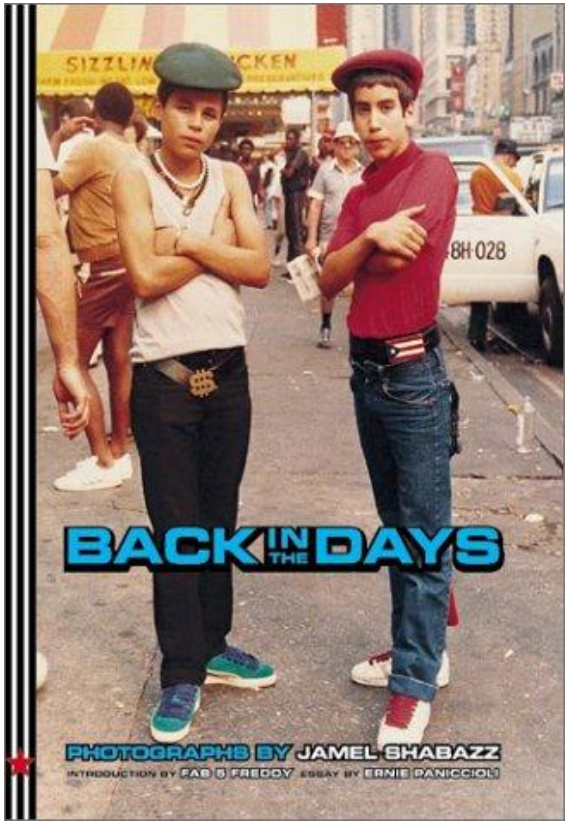


Conclusions (I)

Remember back in the days…

We thought masking was “secure”

Then came HO attacks,
Then came glitches,
Then came early propagation,
Then came coupling,
…



Yet, masking remains one of the frequently used solutions to protect HW and SW implementations!

Conclusions (II)

A similar situation probably holds for leakage resilience

Conclusions (II)

A similar situation probably holds for leakage resilience

- New designs, assumptions, attack techniques

Conclusions (II)

A similar situation probably holds for leakage resilience

- New designs, assumptions, attack techniques
- Raises many open questions, e.g.
  - What about attacks after the S-box?
  - What about EM radiation to “isolate” S-boxes
  - …

Conclusions (II)

A similar situation probably holds for leakage resilience

- New designs, assumptions, attack techniques
- Raises many open questions, e.g.
  - What about attacks after the S-box?
  - What about EM radiation to “isolate” S-boxes
  - …
- Some of them tackled in the paper
- Many other ones to be investigated

Conclusions (II)

A similar situation probably holds for leakage resilience

- New designs, assumptions, attack techniques
- Raises many open questions, e.g.
  - What about attacks after the S-box?
  - What about EM radiation to “isolate” S-boxes
  - …
- Some of them tackled in the paper
- Many other ones to be investigated

We expect that secure & efficient PRFs (e.g. with 16 or 32 block cipher executions per 128-bit input) exist !!

THANKS

http://perso.uclouvain.be/fstandae/