Combining Leakage-Resilient PRFs and Shuffling Towards Bounded Security for Small Embedded Devices

Vincent Grosso, Romain Poussier,
François-Xavier Standaert, Lubos Gaspar.

ICTEAM/ELEN/Crypto Group, Universite catholique de Louvain, Belgium

Cardis 2014, 6th November, 2014

Outline

Background

Our construction

Evaluation setup

Results

Outline

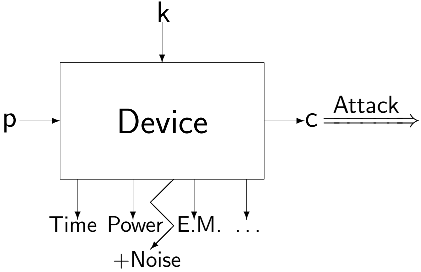
Background

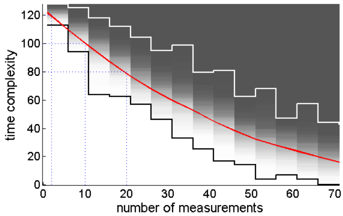
Our construction

Evaluation setup

Results

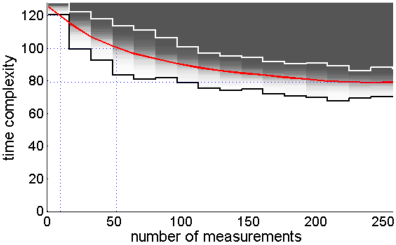
Side channel attacks





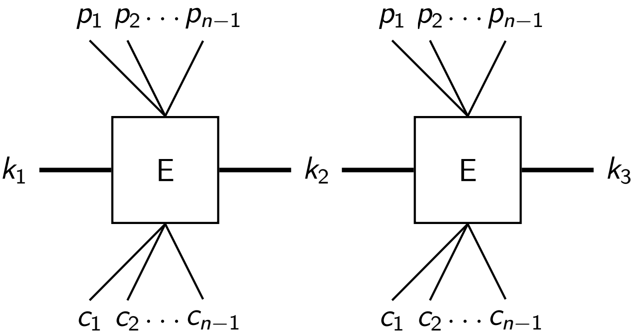
- ▷ Rank of the key decreases exponentially with the number of measurements.

Ideal goal : bounded security



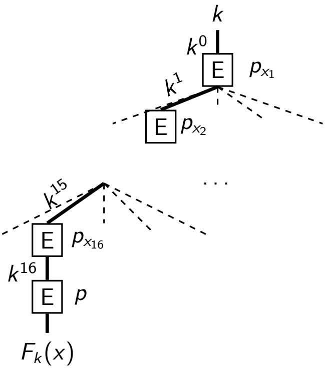
- Limits the number of measurements/plaintexts for 1 key.
- Hope: bound on time complexity independent of the number of measurements.

Leakage resilient PRG [BM84]



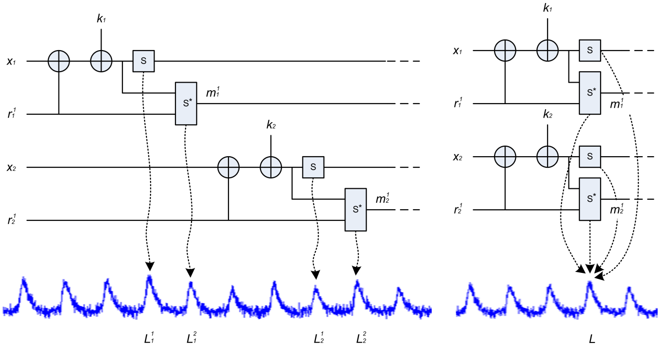
- The construction limits the number of measurements for one key .
- But 2 parties need to be synchronized (stateful) .

Leakage resilient PRF [GGM86]



- The data complexity is limited, not the number of measurements 👎.
- The 2 parties do not need to be synchronized (stateless) 👍.

Implementation hardware vs. software



- In software, the operations are done sequentially. At each time, the trace depends only on one operation.
- In hardware, the operations are done in parallel. The leakage on different operations are at the same time in the trace.

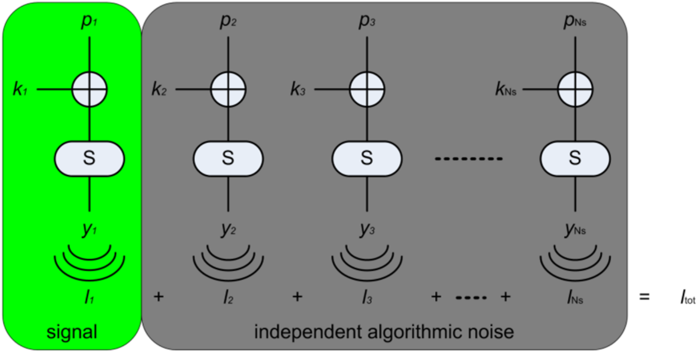
Previous results on leakage resilient primitives

Belaïd et al. have shown that [BGS14]:

	Software	Hardware
LR PRG	bounded	bounded
LR PRF	unbounded	unbounded
LR-PRF+key dependent algorithmic noise [MJS12]	unbounded	bounded*

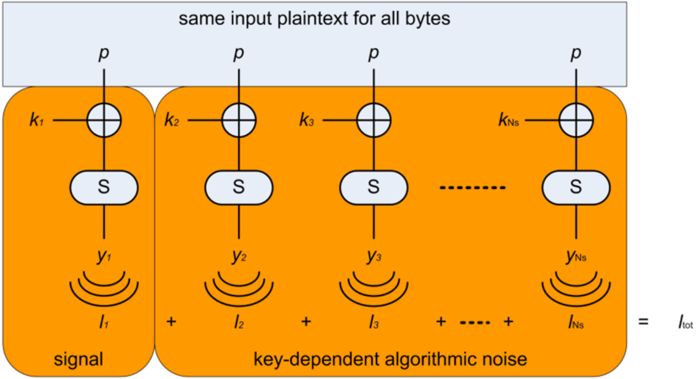
* against specific, but realistic adversaries.

Algorithmic noise in hardware



P_i can be chosen randomly $\Rightarrow$ Divide and conquer attacks could remove the algorithmic noise.

Key dependent algorithmic noise in hardware



Divide and conquer attacks hardly apply in that case [MSJ12].

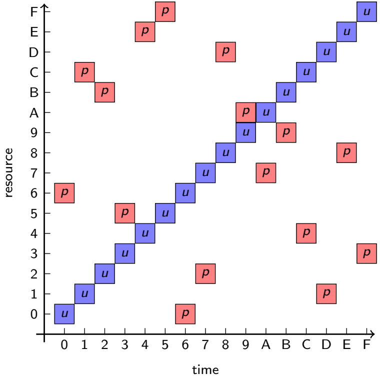
Outline

Research problem: bounded security in software

Natural idea:

- Leakage resilient PRF.
- Shuffled implementation for algorithmic noise.
- Chosen plaintext for key dependent noise.

Shuffling in software



- ▷ Randomize the order of the operations.
- ▷ Trivial attack: integration (sum over all the time) $\Rightarrow$ algorithmic noise.

Key dependent algorithmic noise differences in software/hardware

- There is some information on the permutation used $\Rightarrow$ there is some indication on the subkey used.
- Shuffling can be done on all the operations $\Rightarrow$ key dependent algorithmic noise in all the rounds.

Outline

Background

Our construction

Evaluation setup

Results

Implementation of the counter-measure

- Simulated experiments, parameters:
  - direct leakage on the permutation,
  - indirect resource dependent leakage,
  - indirect time dependent leakage,
  - noise level (not in this presentation).
- On real device, ATmega644p, parameters:
  - direct permutation leakage,
  - indirect leakage as well ($16 \times 16 \times 256$ templates)
  - shuffling implementation (not in this presentation).

Attacks against the target implementation

- Leakage on the S-box outputs:

$$\mathbf{L}_{\mathrm{sprf}}^{\mathrm{r+t}} = [\mathsf{L}_{\mathsf{p}(t),t}(\mathsf{S}(x \oplus k_{\mathsf{p}(t)})) + N_t]_t.$$

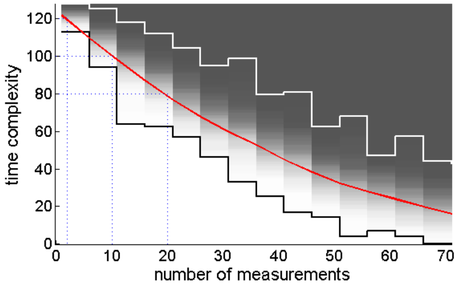
- Direct permutation leakage:

$$\mathbf{L}' = [\mathsf{HW}(\mathsf{p}(t)) + N_t]_t.$$

- Template attacks, worst case scenario [VCMKS12]:

$$\Pr[\mathbf{L} = \mathbf{l} | K_s = k] = \sum_t \frac{\mathsf{f}(t, s, \mathbf{l}')}{\sum_{t'} \mathsf{f}(t', s, \mathbf{l}')} \Pr[L_t = l_t | K_s = k].$$

Metric to evaluate attack



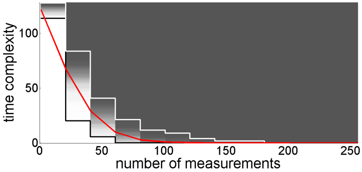
Security margins evaluated thanks to security graph [VCGS13]

Outline

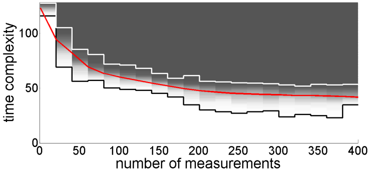
Simulated : indirect resource dependent leakage

No direct leakage.
No time dependency.

Correlation resource leakage : 0.75.



Correlation resource leakage : 0.99.



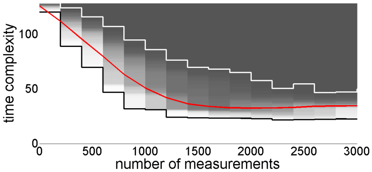
Bounded security if leakage functions are similar enough .

Simulated : indirect time dependent leakage

No direct leakage.
Correlation resource leakage : 0.75.

Correlation time leakage : 0.75.

Correlation time leakage : 0.99.

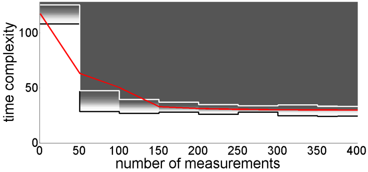


Bounded security when time adds confusion between leakage functions 👍.

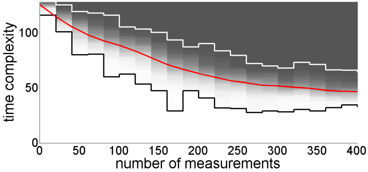
Simulated : direct leakage

Direct leakage.
Correlation resource leakage : 0.75.

Correlation time leakage : 0.75.



Correlation time leakage : 0.99.

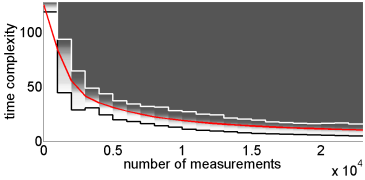


Bound on time complexity when direct leakages are available.

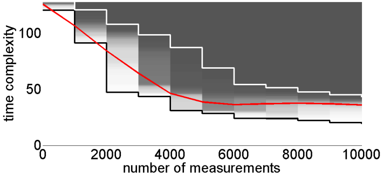
On real device

Correlation resource leakage : 0.86.
Correlation time leakage : 0.84.

Direct leakage.



No direct leakage.



Low bounded security .

Conclusion

- Goal achieved : security bounded stateless primitive in software.
- S-box leakage only (since MixColumns is also protected same result should be observed).
- Concrete security level weak.
- Against specific adversaries.

- Would be interesting to look at other (more noisy) chips and other attack strategies.

THANKS