*ASCA: Comparing Horizontal Side-Channel Attacks

Vincent Grosso

May 31, 2022

CNRS/laboratoire Hubert Curien
Université Jean Monnet
Saint-Étienne

Side-channel attacks

Side-channel attacks: intuition

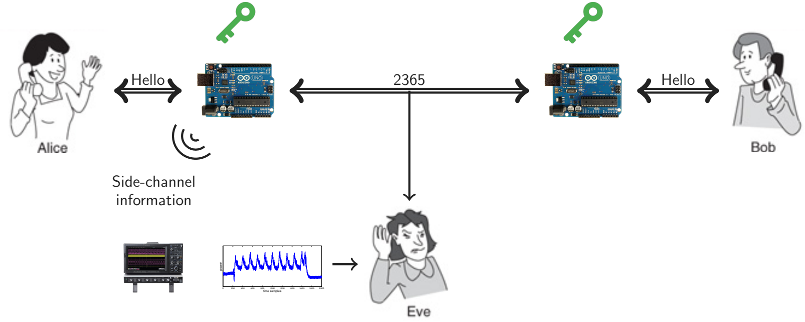
Online poker $\simeq$ Black box model

Only access to cards/outputs of the algorithm

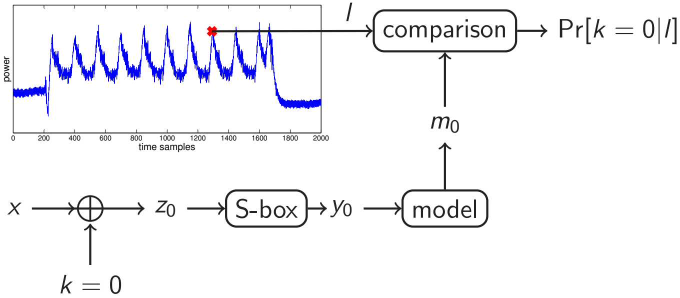
Live poker $\simeq$ Gray box model

Reactions/physical properties of the device can be observed

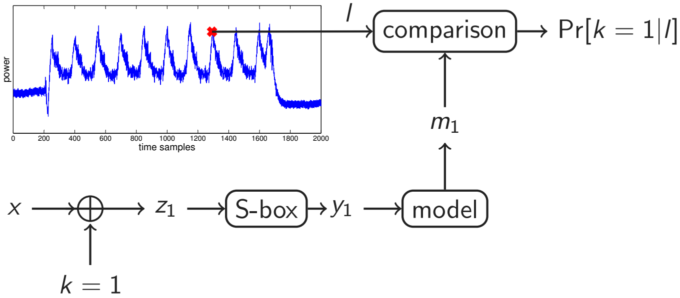
Secure communication

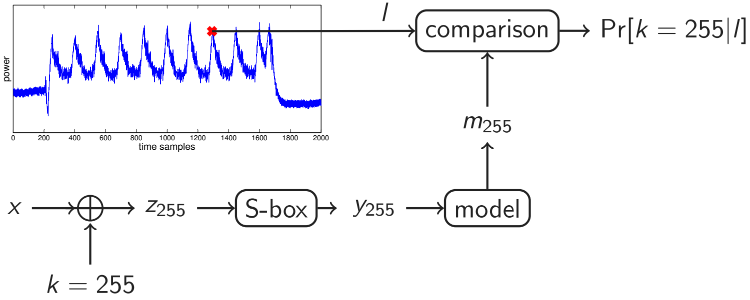


Side-channel attacks: divide-and-conquer strategy

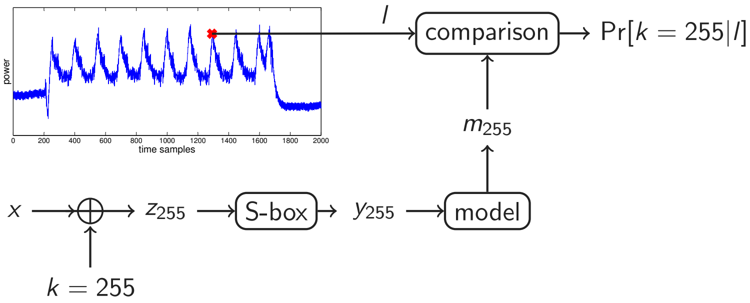


Side-channel attacks: divide-and-conquer strategy




power
0 200 400 600 800 1000 1200 1400 1600 1800 2000
time samples
l
comparison
$\Pr[k = 255|l]$
m_{255}
x
z_{255}
S-box
y_{255}
model
$k = 255$

Side-channel attacks: divide-and-conquer strategy



$$16 \times 2^8 < 2^{128}$$

Information used

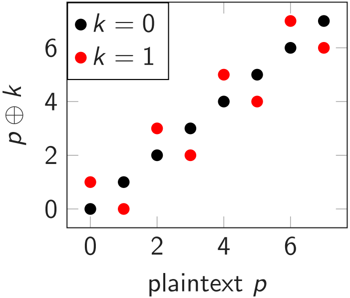


Figure: Before S-box, $\rho = 0.904$

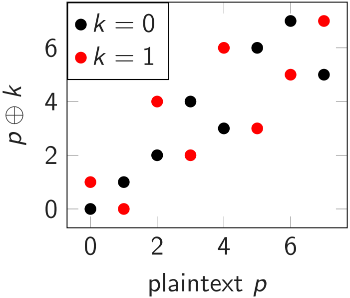


Figure: After S-box, $\rho = 0.571$

Information used

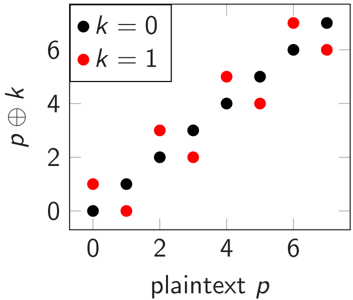


Figure: Before S-box, $\rho = 0.904$

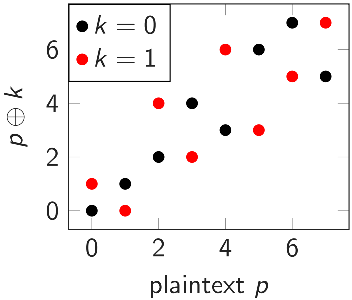
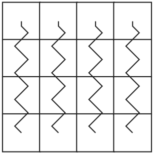


Figure: After S-box, $\rho = 0.571$

Horizontal attacks: use both, use as many information as we can More information $\Rightarrow$ less traces

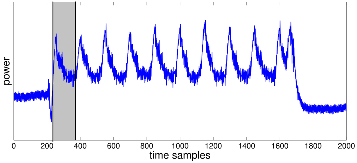
Diffusion issue

Divide and conquer strategy with block cipher with strong diffusion



MixColumns

Few points of the trace can be exploited or large number of bit of the key need to be brute forced



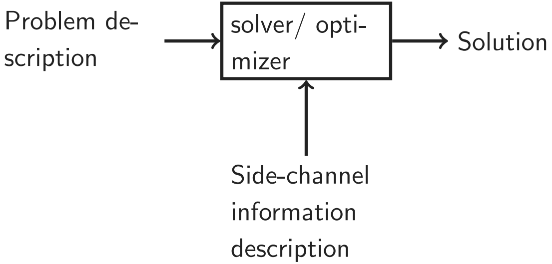
Side-channel exploitable information

- Standard DPA, [Crypto 98] (8 bits)
- Multi target DPA, [Asiacrypt 2014] (32 bits, computationally intensive)
- Combination with cryptanalysis
  - Collision attacks [FSE 2003]
  - *ASCA (ASCA, TASCA, SASCA) [CHES 2009,Asiacrypt 2014]

*ASCA

General description

Use some computational power to reduce data complexity of an attack



Algebraic side-channel attacks (ASCA)

Based on sat solver

- Problem description: Represent the cryptographic algorithm with conjunctive normal form (bit level)

$$(\vee \cdots \vee) \wedge (\vee \cdots \vee) \wedge \cdots \wedge (\vee \cdots \vee)$$

- Side-channel information description: Some more CNFs (only exact information)
- Solver: your favorite SAT solver (cryptominisat -xor clauses-): probabilistic program that outputs UNSAT or SAT with an instantiation of variables such that all clauses are verified

Algebraic side-channel attacks (ASCA)

Based on sat solver

- Problem description: Represent the cryptographic algorithm with conjunctive normal form (bit level)

$$(\vee \cdots \vee) \wedge (\vee \cdots \vee) \wedge \cdots \wedge (\vee \cdots \vee)$$

- Side-channel information description: Some more CNFs (only exact information)
- Solver: your favorite SAT solver (cryptominisat -xor clauses-): probabilistic program that outputs UNSAT or SAT with an instantiation of variables such that all clauses are verified

Any false clause will lead to UNSAT (with high probability)

Tolerant Algebraic side-channel attacks (TASCA)

Based on Satisfiability modulo theories (SMT) solver

- Problem description: Represent the cryptographic algorithm with different equations that will be interpreted according to the adapted theory
  - BitVectors, arrays, integers, real numbers
- Side-channel information description: convert Bitvector to integers (or opposite) and add equation
- Solver: your favorite SMT solver: probabilistic program that outputs UNSAT or SAT with an instantiation of variables such that all clauses are verified

Tolerant Algebraic side-channel attacks (TASCA)

Based on Satisfiability modulo theories (SMT) solver

- Problem description: Represent the cryptographic algorithm with different equations that will be interpreted according to the adapted theory
    - BitVectors, arrays, integers, real numbers
- Side-channel information description: convert Bitvector to integers (or opposite) and add equation
- Solver: your favorite SMT solver: probabilistic program that outputs UNSAT or SAT with an instantiation of variables such that all clauses are verified

Super heavy

Soft Analytical Side-Channel Attacks (SASCA)

Based on LDPC decoding

- Problem description: Represent the cryptographic algorithm as a bipartite graph
  - One set of nodes for operations
  - One set of nodes for intermediate values
- Side-channel information description: leakage operation nodes
- Solver: run belief propagation algorithm (propagate probabilities of each value of each intermediate variable)

Recap: *ASCA

	ASCA	TASCA	SASCA
representation			
speed and mem.			
efficiency			
noise resistance			
rational			

Large register issue

Limitations

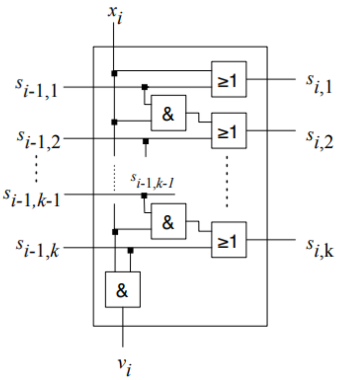
- ASCA: How to efficiently encode HW information?
- TASCA: How to not blow-up memory requirement during solving part ?
- SASCA: How to update messages efficiently ?

Why?

We do not consider information on small word, but about a full register (32-bit vs 8-bit)

- Asymmetric crypto
- permutation based crypto
- Bitsliced countermeasures

Solution: counter



Clauses added : ($\mathcal{O}(wh)$)
Extra variable : ($\mathcal{O}(wh)$)

Toy example

```
for r ← 0 to (n−k)/w do
    s_r = 0
for r ← 0 to (n − k) do
    b = 0
    for c ← 0 to n/z do
        bxor = H_[r,c] and e_c
    t = w/2
    while t > 0 do
        bxor = b ≫ t
        t = t/2
    b = b and 1
    s*_⌊r/w⌋ = s*_⌊r/w⌋ or b ≪ (r mod w)
return s*
```

Results

	8-bit	16-bit	32-bit
time	1	$\times 2$	$\times 16$
memory	1	$\simeq 1$	$\simeq 1$

Recap: *ASCA

	ASCA	TASCA	SASCA
representation			
speed and mem.			
efficiency			
noise resistance			
rational			
Larger register			

Thanks for your attention