Side-Channel Countermeasures’ Dissection
and the Limits of Closed Source Security Evaluations

Olivier Bronchain       François-Xavier Standaert

CHES 2020, Online
Content

Introduction

Countermeasures' Dissection

Information Extraction

Attack Results

Closed Source Evaluation

Conclusion
Side-Channels: How to Design Security?

How to reach high security levels?
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
- Let’s solve it based on physical solutions
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
- Let’s solve it based on physical solutions
  - Noise addition
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
- Let’s solve it based on physical solutions
  - Noise addition
  - Signal reduction
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
- Let’s solve it based on physical solutions
  - Noise addition
  - Signal reduction
- However it may not be enough to provide high protection
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
- Let’s solve it based on physical solutions
  - Noise addition
  - Signal reduction
- However it may not be enough to provide high protection
  - Noise is not a parameter giving exponential security
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
- Let’s solve it based on physical solutions
  - Noise addition
  - Signal reduction
- However it may not be enough to provide high protection
  - Noise is not a parameter giving exponential security
- Exploit "noise amplification" based on mathematical analysis
Side-Channels: How to Design Security?

How to reach high security levels?

- Side-channel attacks are a physical problem
- Let’s solve it based on physical solutions
  - Noise addition
  - Signal reduction
- However it may not be enough to provide high protection
  - Noise is not a parameter giving exponential security
- Exploit "noise amplification" based on mathematical analysis
  - Requires additional hypothesis (e.g., independence for masking)
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- Open approach
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- **Open approach**
  - Evaluator gets all knowledge/control of the target
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- **Open approach**
  - Evaluator gets all knowledge/control of the target
  - Eased verification of physical assumptions
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- Open approach
  - Evaluator gets all knowledge/control of the target
  - Eased verification of physical assumptions
  - More privileged in academic research (but no only)

- Closed approach
  - Evaluator gets restricted knowledge/control of the target
  - Harder verification of physical assumptions
  - In contradiction with Kerckhoff’s principle
  - In part encouraged by some certification practices (e.g., CC)
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- Open approach
  - Evaluator gets all knowledge/control of the target
  - Eased verification of physical assumptions
  - More privileged in academic research (but not only)

- Closed approach
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- **Open approach**
  - Evaluator gets all knowledge/control of the target
  - Eased verification of physical assumptions
  - More privileged in academic research (but no only)

- **Closed approach**
  - Evaluator gets restricted knowledge/control of the target
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- **Open approach**
  - Evaluator gets all knowledge/control of the target
  - Eased verification of physical assumptions
  - More privileged in academic research (but no only)

- **Closed approach**
  - Evaluator gets restricted knowledge/control of the target
  - Harder verification of physical assumptions
What approaches exist in embedded security evaluation?

- **Open approach**
  - Evaluator gets all knowledge/control of the target
  - Eased verification of physical assumptions
  - More privileged in academic research (but no only)

- **Closed approach**
  - Evaluator gets restricted knowledge/control of the target
  - Harder verification of physical assumptions
  - In contradiction with Kerckhoff’s principle
Open vs. Closed Approaches For Evaluations

What approaches exist in embedded security evaluation?

- **Open approach**
  - Evaluator gets all knowledge/control of the target
  - Eased verification of physical assumptions
  - More privileged in academic research (but no only)

- **Closed approach**
  - Evaluator gets restricted knowledge/control of the target
  - Harder verification of physical assumptions
  - In contradiction with Kerckhoff’s principle
  - In part encouraged by some certification practices (e.g., CC)
What About Real-World Targets?

A few published attacks on real products exist:
What About Real-World Targets?

A few published attacks on real products exist:

- Key recovery for bitstream encryption keys (Moradi et al., 2011)
What About Real-World Targets?

A few published attacks on real products exist:

- Key recovery for bitstream encryption keys (Moradi et al., 2011)
- Update forgery on HP Light Bumps (Ronen et al., 2016)
What About Real-World Targets?

A few published attacks on real products exist:

- Key recovery for bitstream encryption keys (Moradi et al., 2011)
- Update forgery on HP Light Bumps (Ronen et al., 2016)
- Car opening against Tesla Key Fob (Wouters et al., 2019)
What About Real-World Targets?

A few published attacks on real products exist:

- Key recovery for bitstream encryption keys (Moradi et al., 2011)
- Update forgery on HP Light Bumps (Ronen et al., 2016)
- Car opening against Tesla Key Fob (Wouters et al., 2019)

Once (huge) reverse engineering done, attacks are straightforward.
What About Real-World Targets?

A few published attacks on real products exist:

- Key recovery for bitstream encryption keys (Moradi et al., 2011)
- Update forgery on HP Light Bumps (Ronen et al., 2016)
- Car opening against Tesla Key Fob (Wouters et al., 2019)

Once (huge) reverse engineering done, attacks are straightforward.

- These examples are however not reflective of certified products
What About Real-World Targets?

A few published attacks on real products exist:

- Key recovery for bitstream encryption keys (Moradi et al., 2011)
- Update forgery on HP Light Bumps (Ronen et al., 2016)
- Car opening against Tesla Key Fob (Wouters et al., 2019)

Once (huge) reverse engineering done, attacks are straightforward.

- These examples are however not reflective of certified products
- We lack practically relevant examples of "sound combinations of countermeasures"
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:

- **Hardened Library for AES-128 encryption/decryption on ARM Cortex M4 Architecture**
  
  **Authors:** Ryad Benadjila, Louiza Khalil, Emmanuel Prouff and Adrian Thillard
  
  This work is linked to the H2020 funded project REASSURE.

  **Introduction**
  
  The members of ANSSI’s laboratory of embedded security have developed a C library to perform AES-128 encryption and decryption on 32-bit Cortex-M ARM architecture while taking Side-Channel Attacks (SCA for short) into account.
  
  The implementation codes are published for research and pedagogical purposes only.
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:

- From a team of experts

Hardened Library for AES-128 encryption/decryption on ARM Cortex M4 Architecture

Authors: Ryad Benadjila, Louiza Khalil, Emmanuel Prouff and Adrian Thillard

This work is linked to the H2020 funded project REASSURE.

Introduction

The members of ANSSI’s laboratory of embedded security have developed a C library to perform AES-128 encryption and decryption on 32-bit Cortex-M ARM architecture while taking Side-Channel Attacks (SCA for short) into account.

The implementation codes are published for research and pedagogical purposes only.
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:
- From a team of experts
- Mixed countermeasures

Hardened Library for AES-128 encryption/decryption on ARM Cortex M4 Architecture

Authors: Ryad Benadjila, Louiza Khalil, Emmanuel Prouff and Adrian Thillard

This work is linked to the H2020 funded project REASSURE.

Introduction

The members of ANSSI's laboratory of embedded security have developed a C library to perform AES-128 encryption and decryption on 32-bit Cortex-M ARM architecture while taking Side-Channel Attacks (SCA for short) into account.

The implementation codes are published for research and pedagogical purposes only.
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:
- From a team of experts
- Mixed countermeasures
- Preliminary leakage assessment

Hardened Library for AES-128 encryption/decryption on ARM Cortex M4 Architecture

Authors: Ryad Benadjila, Louiza Khatli, Emmanuel Prouff and Adrian Thillard
This work is linked to the H2020 funded project REASSURE.

Introduction
The members of ANSSI’s laboratory of embedded security have developed a c library to perform AES-128 encryption and decryption on 32-bit Cortex-M ARM architecture while taking Side-Channel Attacks (SCA for short) into account.
The implementation codes are published for research and pedagogical purposes only.
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:

- From a team of experts
- Mixed countermeasures
- Preliminary leakage assessment

!! Educational purpose only !!

Hardened Library for AES-128 encryption/decryption on ARM Cortex M4 Architecture

Authors: Ryad Benadjila, Louiza Khalil, Emmanuel Prouff and Adrian Thillard

This work is linked to the H2020 funded project REASSURE.

Introduction

The members of ANSSI’s laboratory of embedded security have developed a C library to perform AES-128 encryption and decryption on 32-bit Cortex-M ARM architecture while taking Side-Channel Attacks (SCA for short) into account. The implementation codes are published for research and pedagogical purposes only.
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:
- From a team of experts
- Mixed countermeasures
- Preliminary leakage assessment

!! Educational purpose only !!

It could be used to study:

- Hardened Library for AES-128 encryption/decryption on ARM Cortex M4 Architecture

Authors: Ryad Benadjila, Louiza Khalil, Emmanuel Prouff and Adrian Thillard
This work is linked to the H2020 funded project REASSURE.

Introduction

The members of ANSSI’s laboratory of embedded security have developed a C library to perform AES-128 encryption and decryption on 32-bit Cortex-M ARM architecture while taking Side-Channel Attacks (SCA for short) into account.

The implementation codes are published for research and pedagogical purposes only.
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:
- From a team of experts
- Mixed countermeasures
- Preliminary leakage assessment

!! Educational purpose only !!

It could be used to study:
1. Effectiveness of mixed countermeasures
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:

- From a team of experts
- Mixed countermeasures
- Preliminary leakage assessment

!! Educational purpose only !!

It could be used to study:

1. Effectiveness of mixed countermeasures
2. Security on popular 32-bit MCU’s

Hardened Library for AES-128 encryption/decryption on ARM Cortex M4 Architecture

Authors: Ryad Benadjila, Louiza Khali, Emmanuel Prouff and Adrian Thillard

This work is linked to the H2020 funded project REASSURE.

Introduction

The members of ANSSI’s laboratory of embedded security have developed a C library to perform AES-128 encryption and decryption on 32-bit Cortex-M ARM architecture while taking Side-Channel Attacks (SCA for short) into account.

The implementation codes are published for research and pedagogical purposes only.
Useful step in this direction: ANSSI’s Implem.

Open-source protected AES:
- From a team of experts
- Mixed countermeasures
- Preliminary leakage assessment

!! Educational purpose only !!

It could be used to study:
1. Effectiveness of mixed countermeasures
2. Security on popular 32-bit MCU’s
3. Impact of open designs for worst-case security evaluations
Profiled Side-Channel Attacks in

Worst-case analysis in two phases:
Profiled Side-Channel Attacks in

Worst-case analysis in two phases:

1. Profiling / Learning target behavior
Worst-case analysis in two phases:

1. Profiling / Learning target behavior
   - Algorithm/Implementation knowledge
Worst-case analysis in two phases:

1. Profiling / Learning target behavior
   - Algorithm/Implementation knowledge
   - Leakage examples in controlled settings (i.e. known randomness)
Profiled Side-Channel Attacks in

Worst-case analysis in two phases:

1. Profiling / Learning target behavior
   - Algorithm/Implementation knowledge
   - Leakage examples in controlled settings (i.e. known randomness)
Profiled Side-Channel Attacks in

Worst-case analysis in two phases:

1. Profiling / Learning target behavior
   - Algorithm/Implementation knowledge
   - Leakage examples in controlled settings (i.e. known randomness)

2. Attack
Profiled Side-Channel Attacks in

Worst-case analysis in two phases:
1. Profiling / Learning target behavior
   - Algorithm/Implementation knowledge
   - Leakage examples in controlled settings (i.e. known randomness)
2. Attack
   - Extract information from leakage
Profiled Side-Channel Attacks in

Worst-case analysis in two phases:

1. Profiling / Learning target behavior
   - Algorithm/Implementation knowledge
   - Leakage examples in controlled settings (i.e. known randomness)

2. Attack
   - Extract information from leakage
   - Processing for secret recovery
Content

Introduction

Countermeasures' Dissection

Information Extraction

Attack Results

Closed Source Evaluation

Conclusion
Countermeasures

At a high level:
Countermeasures

At a high level:

- Affine masking on bytes
At a high level:

- **Affine masking on bytes**
  - Multiplicative mask $r_m$ (same for all the 16-bytes)
At a high level:
- **Affine masking on bytes**
  - Multiplicative mask $r_m$ (same for all the 16-bytes)
  - Additive mask $r_a$
At a high level:

- **Affine masking on bytes**
  - Multiplicative mask $r_m$ (same for all the 16-bytes)
  - Additive mask $r_a$
  - Requires alternative Sbox table pre-computation
At a high level:

- **Affine masking on bytes**
  - Multiplicative mask $r_m$ (same for all the 16-bytes)
  - Additive mask $r_a$
  - Requires alternative Sbox table pre-computation
- **Shuffled execution**
Countermeasures

At a high level:

- Affine masking on bytes
  - Multiplicative mask $r_m$ (same for all the 16-bytes)
  - Additive mask $r_a$
  - Requires alternative Sbox table pre-computation

- Shuffled execution
  - One permutation for the 16 Sboxes
Countermeasures

At a high level:

- **Affine masking on bytes**
  - Multiplicative mask $r_m$ (same for all the 16-bytes)
  - Additive mask $r_a$
  - Requires alternative Sbox table pre-computation

- **Shuffled execution**
  - One permutation for the 16 Sboxes
  - Another permutation for the 4 MixColumns
At a high level:

- **Affine masking on bytes**
  - Multiplicative mask $r_m$ (same for all the 16-bytes)
  - Additive mask $r_a$
  - Requires alternative Sbox table pre-computation

- **Shuffled execution**
  - One permutation for the 16 Sboxes
  - Another permutation for the 4 MixColumns
  - Both are pre-computed
Countermeasures

Inputs  | Pre-computation  | Encryption

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R} \]

AddRoundKey

\[ r \in Sbox \]

ShiftRows

MixColumns

\[ p \vec{C} \]

\[ p \vec{R} \]

MultiplicativePre-Computation

\[ r \in Sbox' \]

Perm. over \{0, ..., 15\}

Computation

\[ p \vec{C} \]

\[ p \vec{R} \]

seed

1

seed

2

16

Perm. over \{0, 1, 2, 3\}

Computation

\[ p' \vec{C} \]

\[ p' \vec{R} \]

seed'

1

seed'

2
Countermeasures

Inputs

$\vec{R}_a$

$\vec{P}$

$r_m, r_in, r_out$

Pre-computation

Encryption

Multiplicative Pre-Computation

AddRoundKey

Sbox

MixColumns

ShiftRows

Perm. over $\{0,...,15\}$

Perm. over $\{0,1,2,3\}$
Countermeasures

\[ C = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

AddRoundKey

\[ r_{\text{in}} \rightarrow \text{Sbox}' \]

\[ r_{\text{out}} \rightarrow \text{ShiftRows} \]

\[ \text{MixColumns} \]

\[ \vec{P} \rightarrow \text{Multiplicative Pre-Computation} \]

\[ r_m, r_{\text{in}}, r_{\text{out}} \rightarrow \text{Perm. over } \{0, \ldots, 15\} \]

\[ \text{Computation} \]

\[ \vec{R}_a \rightarrow \text{Perm. over } \{0, 1, 2, 3\} \]

\[ \text{Computation} \]

\[ \vec{C}, \vec{R}_a \rightarrow \text{seed}_1 \]

\[ \vec{C}, \vec{R}_a \rightarrow \text{seed}_2 \]

\[ 16 \rightarrow \text{Sbox}' \]

\[ 16 \rightarrow \text{Encryption} \]
Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]
Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

**Pre-computation**
- \( \vec{R}_a \)
- \( \vec{P} \)
- \( r_m, r_{in}, r_{out} \)
- Multiplicative Pre-Computation
- \( \text{Sbox}' \)

**Encryption**
- \( \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \)
\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

Inputs
- \( \vec{R}_a \)
- \( \vec{P} \)
- \( r_m, r_{in}, r_{out} \)

Pre-computation
- Multiplicative Pre-Computation
- \( \text{Sbox}' \)

Encryption
- \( \text{AddRoundKey} \)
- \( r_{in} \)
Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

\[ \text{AddRoundKey} \]

\[ r_{in} \]

\[ \text{Sbox}' \]

\[ \text{Sbox}' \]

\[ r_{in}, r_{in}, r_{out} \]

\[ \vec{P} \]

\[ \vec{R}_a \]

\[ \vec{R}_a \]

\[ \text{Pre-computation} \]

\[ \text{Encryption} \]

\[ \text{Inputs} \]

\[ \text{Multiplicative Pre-Computation} \]
Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

AddRoundKey

\[ r_{in}, r_{out} \]

Multiplicative Pre-Computation

Sbox'
Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

AddRoundKey

\[ r_{in} \]

\[ r_{out} \]

ShiftRows

MixColumns

Multiplicative Pre-Computation

Sbox'
Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

\[ \vec{P} \]

\[ r_m, r_{in}, r_{out} \]

\[ \text{Sbox}' \]

\[ \text{AddRoundKey} \]

\[ r_{in} \]

\[ r_{out} \]

\[ \text{ShiftRows} \]

\[ \text{MixColumns} \]

\[ \text{ShiftRows} \]

\[ \text{MixColumns} \]

\[ \text{seed}_1 \]

\[ \text{seed}_2 \]

\[ \text{seed}'_1 \]

\[ \text{seed}'_2 \]
Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R_a} \]

- **Inputs**: \( \vec{R_a} \)
- **Pre-computation**: Multiplicative Pre-Computation, Sbox', Perm. over \( \{0, \ldots, 15\} \) Computation
  - seed\(_1\), seed\(_2\)
- **Encryption**: AddRoundKey, Sbox', Perm. over \( \{0, 1, 2, 3\} \) Computation
  - seed\(_1'\), seed\(_2'\)
  - \( r_{in}, r_{out} \)
  - ShiftRows, MixColumns
Countermeasures

Countermeasures

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

AddRoundKey

MixColumns

ShiftRows

Perm. over \{0, \ldots, 15\}

Computation

Perm. over \{0, 1, 2, 3\}

Computation
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

\[ f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2] \]
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

\[ f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2] \]

Mult. mask
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

\[ f[\overline{l}|x] \propto \sum r_m \sum r_a \sum o_1 \sum o_2 \cdot f[\overline{l}|r_m, r_a, c, o_1, o_2] \]

- Add. mask
- Mult. mask

Optimal but rapidly out of reach:
- One template per randomness combination
- Sum over all the possible randomness

\[ \Rightarrow \] Hypotheses needed
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

\[
f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2]
\]
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

\[
f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2]
\]
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

\[
f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2]
\]

Optimal but rapidly out of reach:

▶ One template per randomness combination
▶ Sum over all the possible randomness $\Rightarrow$ Hypotheses needed
Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

$$f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2]$$

Optimal but rapidly out of reach:

- One template per randomness combination
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

$$f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2]$$

Optimal but rapidly out of reach:

- One template per randomness combination
- Sum over all the possible randomness
Optimal Distinguisher

Profiled attacks are based on secret conditional distribution which depends on the countermeasures.

Full expression is written as

$$f[l|x] \propto \sum_{r_m} \sum_{r_a} \sum_{o_1} \sum_{o_2} f[l|r_m, r_a, c, o_1, o_2]$$

Optimal but rapidly out of reach:

- One template per randomness combination
- Sum over all the possible randomness

\[ \Rightarrow \text{Hypotheses needed} \]
Countermeasures Dissection

Assuming $\perp$ leakages on secret:

$$f([\vec{l}]x) \propto \sum_{r_m} \Pr[r_m|\vec{l}_m] \cdot \sum_{r_a} \cdot \left( \sum_{o_1} f[\vec{l}_a|r_a, o_1] \cdot \Pr[o_1|\vec{l}_o] \right) \cdot \left( \sum_{o_2} f[\vec{l}_c|c, o_2] \cdot \Pr[o_2|\vec{l}_o] \right)$$
Countermeasures Dissection

Assuming \( \perp \) leakages on secret:

\[
f[\vec{l}|x] \propto \sum_{r_m} \Pr[r_m|\vec{l}_m] \cdot \sum_{r_a} \left( \sum_{o_1} f[\vec{l}_a|o_1] \cdot \Pr[o_1|\vec{l}_o] \cdot \left( \sum_{o_2} f[\vec{l}_c|o_2] \cdot \Pr[o_2|\vec{l}_o] \right) \right)
\]
Countermeasures Dissection

Assuming $\perp$ leakages on secret:

$$f[\bar{l}|x] \propto \sum_{r_m} \Pr[r_m|\bar{l}_{r_m}] \cdot \sum_{r_a} \cdot \left( \sum_{o_1} f[\bar{l}_r|o_1, r_a] \cdot \Pr[o_1|\bar{l}_{o_1}] \right) \cdot \left( \sum_{o_2} f[\bar{l}_c|c, o_2] \cdot \Pr[o_2|\bar{l}_{o_2}] \right)$$

Mult. mask

Add. mask + Perm

- What: From combined countermeasures, expected multiplicative effect
- Reduce it to a small factor, ideally of 1.
- How: Bias the sums by independent partial attacks on secrets (i.e. shares)
- $\Downarrow$ attack time complexity because terms are removed
- $\Downarrow$ number of templates because not joint on all randomness
Countermeasures Dissection

Assuming ⊥ leakages on secret:

\[ f[\tilde{l}_x] \propto \sum_{r_m} \Pr[r_m|\tilde{l}_m] \cdot \sum_{r_o} \left( \sum_{o_1} f[\tilde{l}_{o_1}, r_a, o_1] \cdot \Pr[o_1|\tilde{l}_{o_1}] \right) \cdot \left( \sum_{o_2} f[\tilde{l}_{o_2}, c, o_2] \cdot \Pr[o_2|\tilde{l}_{o_2}] \right) \]
Countermeasures Dissection

Assuming $\bot$ leakages on secret:

$$f[\vec{l}|x] \propto \sum_{r_m} \Pr[r_m|\vec{l}_{r_m}] \cdot \sum_{r_a} \cdot \left(\sum_{o_1} f[\vec{l}_{r_a}|r_a, o_1] \cdot \Pr[o_1|\vec{l}_{o_1}]ight) \cdot \left(\sum_{o_2} f[\vec{l}_{o_2}|c, o_2] \cdot \Pr[o_2|\vec{l}_{o_2}]\right)$$

Countermesures' Dissection:

- Mult. mask
- Add. mask + Perm
- Enc. + Perm

- What: From combined countermeasures, expected multiplicative effect
- Reduce it to a small factor, ideally of 1.
- How: Bias the sums by independent partial attacks on secrets (i.e. shares)
- ↓ attack time complexity because terms are removed
- ↓ number of templates because not joint on all randomness
Countermeasures' Dissection

Assuming \( \perp \) leakages on secret:

\[
f[\vec{l}|x] \propto \sum_{r_m} \Pr[r_m|\vec{l}] \cdot \sum_{r_a} \cdot \left( \sum_{o_1} f[\vec{l}_a|r_a, o_1] \cdot \Pr[o_1|\vec{l}_o] \right) \cdot \left( \sum_{o_2} f[\vec{l}_c|c, o_2] \cdot \Pr[o_2|\vec{l}_o] \right)
\]

Countermeasures' Dissection:

- **What:** From combined countermeasures, expected multiplicative effect

- **How:** Bias the sums by independent partial attacks on secrets (i.e. shares)

- **\( \downarrow \)** attack time complexity because terms are removed

- **\( \downarrow \)** number of templates because not joint on all randomness
Countermeasures’ Dissection

Assuming $\perp$ leakages on secret:

$$f[\vec{l}|x] \propto \sum_{r_m} \Pr[r_m|\vec{l}_m] \cdot \sum_{r_a} \cdot \left( \sum_{o_1} f[\vec{l}_a|r_a, o_1] \cdot \Pr[o_1|\vec{l}_o] \right) \cdot \left( \sum_{o_2} f[\vec{l}_c|c, o_2] \cdot \Pr[o_2|\vec{l}_o] \right)$$

Enc. + Perm

Mult. mask

Add. mask + Perm

Countermeasures’ Dissection:

- **What:** From combined countermeasures, expected multiplicative effect
  - Reduce it to a small factor, ideally of 1.
Countermeasures Dissection

Assuming \( \perp \) leakages on secret:

\[
f[\vec{l}|x] \propto \sum_{r_m} Pr[r_m|\vec{l}_m] \cdot \sum_{r_a} \left( \sum_{o_1} f[\vec{l}_a|r_a, o_1] \cdot Pr[o_1|\vec{l}_o] \right) 
\cdot \left( \sum_{o_2} f[\vec{l}_c|c, o_2] \cdot Pr[o_2|\vec{l}_o] \right)
\]

Countermeasures’ Dissection:

- **What:** From combined countermeasures, expected multiplicative effect
  - Reduce it to a small factor, ideally of 1.
- **How:** Bias the sums by independent partial attacks on secrets (i.e. shares)
Countermeasures Dissection

\[ f[\vec{l}|x] \propto \sum_{r_m} \Pr[r_m|\vec{l}_m] \cdot \sum_{r_a} \cdot \left( \sum_{o_1} f[\vec{l}_{r_a}|r_a, o_1] \cdot \Pr[o_1|\vec{l}_o] \right) \cdot \left( \sum_{o_2} f[\vec{l}_c|c, o_2] \cdot \Pr[o_2|\vec{l}_o] \right) \]

Assuming \( \perp \) leakages on secret:

Countermeasures’ Dissection:

- **What:** From combined countermeasures, expected multiplicative effect
  - Reduce it to a small factor, ideally of 1.
- **How:** Bias the sums by independent partial attacks on secrets (i.e. shares)
  - \( \downarrow \) attack time complexity because terms are removed
Countermeasures' Dissection:

- What: From combined countermeasures, expected multiplicative effect
  - Reduce it to a small factor, ideally of 1.
- How: Bias the sums by independent partial attacks on secrets (i.e. shares)
  - \( \triangleleft \) attack time complexity because terms are removed
  - \( \triangleleft \) number of templates because not joint on all randomness
Content

Introduction

Countermeasures' Dissection

Information Extraction

Attack Results

Closed Source Evaluation

Conclusion
Measurement Setup

Composed of

- Cortex-M4 Atmel
- High end EM Probe
- PicoScope 5000 series sampling at 1GHz
Measurement Setup

Composed of

- Cortex-M4 Atmel
- High end EM Probe
- PicoScope 5000 series sampling at 1GHz
Measurement Setup

Composed of

- Cortex-M4 Atmel
- High end EM Probe
- PicoScope 5000 series sampling at 1GHz
Measurement Setup

Composed of

- Cortex-M4 Atmel
- High end EM Probe
- PicoScope 5000 series sampling at 1GHz

How to extract information in

.pack?
Profiling (e.g., permutation)

1. Compute SNR
Profiling (e.g., permutation)

1. Compute SNR
2. Select points of interest
Proﬁling (e.g., permutation)

1. Compute SNR
2. Select points of interest
Proﬁling (e.g., permutation)

1. Compute SNR
2. Select points of interest
3. Train projection
Profiling (e.g., permutation)

1. Compute SNR
2. Select points of interest
3. Train projection
Profiling (e.g., permutation)

1. Compute SNR
2. Select points of interest
3. Train projection
4. Project to subspace
**Profiling (e.g., permutation)**

1. Compute SNR
2. Select points of interest
3. Train projection
4. Project to subspace
5. Fit pdf estimation (i.e. gauss.)

\[ f[\tilde{l}_{o_1}|o_1 = 0] \]
Profiling (e.g., permutation)

1. Compute SNR
2. Select points of interest
3. Train projection
4. Project to subspace
5. Fit pdf estimation (i.e. gauss.)

\[ f[\vec{l}_0 | o_1 = 0] \]

\[ f[\vec{l}_0 | o_1 = 1] \]
Partial Attacks

1. Measure a trace

![Signal vs Time Graph](image-url)
Partial Attacks

1. Measure a trace
2. Keep only points of interest
Partial Attacks

1. Measure a trace
2. Keep only points of interest
3. Project to subspace

- Measure a trace
- Keep only points of interest
- Project to subspace

**Figure:**
- Signal [mV]
- Time [s] × 10⁻³
- PCA Training
- PCA
- 3000
- 3
- (Almost) Perfect Dissection

**Equation:**
\[
\Pr[O_1 = 0 | \vec{l}_1] \propto \sum_r m \Pr[r | \vec{l}_r] \cdot \sum_r a \cdot (\sum_o O_1 f[\vec{l}_r | r, o_1] \cdot \Pr[O_1 | \vec{l}_o]) \cdot (\sum_o O_2 f[\vec{l}_c | c, o_2] \cdot \Pr[O_2 | \vec{l}_o])
\]

**Table:**

<table>
<thead>
<tr>
<th>Signal [mV]</th>
<th>Time [s] × 10⁻³</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.01</td>
<td>0.0</td>
</tr>
<tr>
<td>0.01</td>
<td>0.4</td>
</tr>
<tr>
<td>0.02</td>
<td>0.6</td>
</tr>
</tbody>
</table>

**Graphs:**
- PCA Analysis
- Signal Distribution
Partial Attacks

1. Measure a trace
2. Keep only points of interest
3. Project to subspace
4. Estimate probability from pdf
**Partial Attacks**

1. Measure a trace
2. Keep only points of interest
3. Project to subspace
4. Estimate probability from pdf

\[
\begin{align*}
    f[l|x] & \propto \sum_{r_m} \Pr[r_m|l_{m}] \cdot \sum_{r_a} \\
    & \cdot \left( \sum_{o_1} f[l_{a}|r_a, o_1] \cdot \Pr[o_1|l_{o_1}] \right) \\
    & \cdot \left( \sum_{o_2} f[l_{c}|c, o_2] \cdot \Pr[o_2|l_{o_2}] \right)
\end{align*}
\]

- **Pr** \[ o_1 = 0 | l_{o_1} \]
- **Pr** \[ o_1 = 1 | l_{o_1} \]
Partial Attacks

1. Measure a trace
2. Keep only points of interest
3. Project to subspace
4. Estimate probability from pdf
### Partial Attacks

1. Measure a trace
2. Keep only points of interest
3. Project to subspace
4. Estimate probability from pdf

\[
f[\vec{I}|x] \propto \sum_{r_m} Pr[r_m|\vec{I}_m] \cdot \sum_{r_a} \left( \sum_{o_1} f[\vec{I}_a|r_a, o_1] \cdot Pr[o_1|\vec{I}_o] \right) \cdot \left( \sum_{o_2} f[\vec{I}_c|c, o_2] \cdot Pr[o_2|\vec{I}_o] \right)
\]

\[
Pr[o_1 = 0|\vec{I}_o]
\]
Partial Attacks

1. Measure a trace
2. Keep only points of interest
3. Project to subspace
4. Estimate probability from pdf

\[
Pr[\omega_1 = 0 | \vec{l}_{\omega_1}] \quad Pr[\omega_1 = 1 | \vec{l}_{\omega_1}]
\]

Signal [mV] × 10^{-3}

Signal [mV] × 10^{-3}

(Almost) Perfect Dissection

Ineffective permutations and \( r_m \)

Pr\[o_1 = 0 | l_{\omega_1}\]
Content

Introduction

Countermeasures' Dissection

Information Extraction

Attack Results

Closed Source Evaluation

Conclusion
Attack Path’s

Inputs | Pre-computation | Encryption
--- | --- | ---
\( \hat{R}_a \) | Multiplicative Pre-Computation | AddRoundKey
\( r_m, r_m', r_{out} \) | \( \hat{C} = (r_m \otimes \hat{P}) \oplus \hat{R}_a \) | \( p_{C'} \rightarrow \{ \text{ShiftRows} \} \rightarrow p_{\hat{R}_a} \)
\( \hat{P} \) | Sbox' | \( r_{in} \)
seed\_1 16 16 2 2 | Perm. over \( \{0, \ldots, 15\} \) Computation | \( r_{out} \)
seed\_2 | Perm. over \( \{0, 1, 2, 3\} \) Computation | MixColumns
seed\_1' 2 2 | \( p_{C'} \rightarrow \{ \text{MixColumns} \} \rightarrow p'_{\hat{R}_a} \) |
Attack Path’s

Attacker should at least:

- Get information \( r_m \)
- Get information \( r_a \) and \( c \)
- Uneven shuffling:
  - No permutation
  - 2-bit seeded permutations
  - 16-bit seeded permutations
  - All permutations can be enumerated
- We focus on the 2-bit seeded permutation

\[
\vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a
\]
Attack Path’s

Inputs

Pre-computation

Encryption

\[ \vec{C} = (r_m \otimes \vec{P}) \oplus \vec{R}_a \]

Attacker should at least:

- Get information \( r_m \)
Attack Path's

Attacker should at least:
- Get information $r_m$
- Get information $r_a$ and $c$
Attack Path’s

Attacker should at least:

- Get information $r_m$
- Get information $r_a$ and $c$

Uneven shuffling:
Attack Path's

Attacker should at least:
- Get information $r_m$
- Get information $r_a$ and $c$

Uneven shuffling:
- No permutation
Attack Path’s

Attacker should at least:

- Get information $r_m$
- Get information $r_a$ and $c$

Uneven shuffling:

- No permutation
- 2-bit seeded permutations
Attack Path’s

Attacker should at least:

- Get information $r_m$
- Get information $r_a$ and $c$

Uneven shuffling:

- No permutation
- 2-bit seeded permutations
- 16-bit seeded permutations
Attack Path’s

Attacker should at least:

▶ Get information $r_m$
▶ Get information $r_a$ and $c$

Uneven shuffling:

▶ No permutation
▶ 2-bit seeded permutations
▶ 16-bit seeded permutations

▶ All permutations can be enumerated
Attacker should at least:

- Get information $r_m$
- Get information $r_a$ and $c$

Uneven shuffling:

- No permutation
- 2-bit seeded permutations
- 16-bit seeded permutations

- All permutations can be enumerated
- We focus on the 2-bit seeded permutation
Attack Results

Divide & Conquer:

1. On each 16 bytes:
   - Entropy \downarrow with measurements
   - Less than a bit with 3,000 traces
   - One "harder" byte per column

2. On full key:
   - Entropy \downarrow with measurements
   - Less than a bit with 4,000 traces
   - About 1,100 with post-processing

Full key in 1 minute of measurement
**Attack Results**

**Divide & Conquer:**

1. On each 16 bytes:
Attack Results

Divide & Conquer:

1. On each 16 bytes:
   - Entropy $\downarrow$ with measurements

![Graph showing entropy decrease over number of measurements for different columns and the average.](image-url)
Attack Results

Divide & Conquer:
1. On each 16 bytes:
   - Entropy ↘ with measurements
   - Less than a bit with 3,000 traces
Attack Results

Divide & Conquer:

1. On each 16 bytes:
   - Entropy \( \downarrow \) with measurements
   - Less than a bit with 3,000 traces
   - One "harder" byte per column
Attack Results

Divide & Conquer:

1. On each 16 bytes:
   - Entropy with measurements
   - Less than a bit with 3,000 traces
   - One "harder" byte per column

2. On full key:
**Attack Results**

**Divide & Conquer:**

1. **On each 16 bytes:**
   - Entropy \( \downarrow \) with measurements
   - Less than a bit with 3,000 traces
   - One "harder" byte per column

2. **On full key:**
   - Entropy \( \downarrow \) with measurements
Divide & Conquer:

1. On each 16 bytes:
   - Entropy \(\downarrow\) with measurements
   - Less than a bit with 3,000 traces
   - One "harder" byte per column

2. On full key:
   - Entropy \(\downarrow\) with measurements
   - Less than a bit with 4,000 traces
**Attack Results**

Divide & Conquer:

1. On each 16 bytes:
   - Entropy $\downarrow$ with measurements
   - Less than a bit with 3,000 traces
   - One "harder" byte per column

2. On full key:
   - Entropy $\downarrow$ with measurements
   - Less than a bit with 4,000 traces
   - About 1,100 with post-processing
Divide & Conquer:

1. On each 16 bytes:
   - Entropy \( \downarrow \) with measurements
   - Less than a bit with 3,000 traces
   - One "harder" byte per column

2. On full key:
   - Entropy \( \downarrow \) with measurements
   - Less than a bit with 4,000 traces
   - About 1,100 with post-processing

Full key in 1 minute of measurement
Content

Introduction
Countermeasures' Dissection
Information Extraction
Attack Results
Closed Source Evaluation
Conclusion
Can this be automated in ☐️ ?

How the knowledge of the target helps in a worst-case evaluation ?
Can this be automated in 📦 ?

How the knowledge of the target helps in a worst-case evaluation?

- Evaluators do not always have full control on the target
Can this be automated in 📦 ?

How the knowledge of the target helps in a worst-case evaluation?

▶ Evaluators do not always have full control on the target
▶ If it helps, worrying for long term security:
Can this be automated in 📦 ?

How the knowledge of the target helps in a worst-case evaluation?

- Evaluators do not always have full control on the target
- If it helps, worrying for long term security:
  - Adversary with a better strategy can be more powerful than the evaluator
Can this be automated in 📦 ?

How the knowledge of the target helps in a worst-case evaluation?

- Evaluators do not always have full control on the target
- If it helps, worrying for long term security:
  - Adversary with a better strategy can be more powerful than the evaluator

Experiments with machine learning:
Can this be automated in 📦?

How the knowledge of the target helps in a worst-case evaluation?

▶ Evaluators do not always have full control on the target
▶ If it helps, worrying for long term security:
  ▶ Adversary with a better strategy can be more powerful than the evaluator

Experiments with machine learning:

▶ Representative of closed approach since able to deal with unknown countermeasures
Can this be automated in 📦?

How the knowledge of the target helps in a worst-case evaluation?

- Evaluators do not always have full control on the target
- If it helps, worrying for long term security:
  - Adversary with a better strategy can be more powerful than the evaluator

Experiments with machine learning:

- Representative of closed approach since able to deal with unknown countermeasures
- We instantiate MLP classifiers in simulated settings
Simulated Experimental Setting
Simulated Experimental Setting

Boolean Masking with leakage on:

\[
x \oplus r \\
\downarrow \\
HW(\cdot) \\
\downarrow \\
+ \leftarrow \eta_1 \\
\zeta \\
l_1
\]

\[
r \leftarrow \{0, \ldots, 255\} \\
\downarrow \\
HW(\cdot) \\
\downarrow \\
+ \leftarrow \eta_2 \\
\zeta \\
l_2
\]

\[
1 \\
\eta_1 \\
l_3
\]
Simulated Experimental Setting

Boolean Masking with leakage on:

- Two shares

\[
\begin{align*}
    x \otimes r & \\
    \downarrow & \\
    r & \leftarrow \{0, \ldots , 255\} & 1
\end{align*}
\]
Simulated Experimental Setting

Boolean Masking with leakage on:
- Two shares
- Hamming weight + Gaussian noise
Simulated Experimental Setting

Boolean Masking with leakage on:
- Two shares
- Hamming weight + Gaussian noise

Affine Masking with leakage on:
## Simulated Experimental Setting

### Boolean Masking with leakage on:
- Two shares
- Hamming weight + Gaussian noise

### Affine Masking with leakage on:
- Two shares + Multiplicative mask
Simulated Experimental Setting

Boolean Masking with leakage on:
- Two shares
- Hamming weight + Gaussian noise

Affine Masking with leakage on:
- Two shares + Multiplicative mask
- Hamming weight + Gaussian noise
Comparison Open vs. Closed Approaches

For:
▶ Schemes are equivalent
▶ No need to learn multiplications

For:
▶ Schemes are not equivalent
▶ Need to learn multiplications based on leakage
▶ Harder with \( \uparrow \) field size
▶ Profiling cost of such a closed evaluation will be prohibitive
▶ While comes for free in the box
Comparison Open vs. Closed Approaches

For ☐:

For ☐:

3-bit
Comparison Open vs. Closed Approaches

For [ ]:
- Schemes are equivalent

For [ ]:
- Schemes are not equivalent

3-bit
Comparison Open vs. Closed Approaches

For \( \square \):

- Schemes are equivalent
- No need to learn multiplications

For \( \Box \):

- Schemes are not equivalent
- Need to learn multiplications based on leakage
Comparison Open vs. Closed Approaches

For 📦:
- Schemes are equivalent
- No need to learn multiplications

For 📦:
- Schemes are not equivalent
- Need to learn multiplications based on leakage

4-bit
Comparison Open vs. Closed Approaches

For 

- Schemes are equivalent
- No need to learn multiplications

For 

- Schemes are not equivalent
- Need to learn multiplications based on leakage

6-bit
Comparison Open vs. Closed Approaches

For 

- Schemes are equivalent
- No need to learn multiplications

For 

- Schemes are not equivalent
- Need to learn multiplications based on leakage
- Harder with \(\uparrow\) field size
Comparison Open vs. Closed Approaches

For 📦:
- Schemes are equivalent
- No need to learn multiplications

For 📦:
- Schemes are not equivalent
- Need to learn multiplications based on leakage
- Harder with ↗ field size

- Profiling cost of such a closed evaluation will be prohibitive
Comparison Open vs. Closed Approaches

For 
- Schemes are equivalent
- No need to learn multiplications

For 
- Schemes are not equivalent
- Need to learn multiplications based on leakage
- Harder with \( \uparrow \) field size

Profiling cost of such a closed evaluation will be prohibitive
While comes for free in withe box
Content

Introduction

Countermeasures' Dissection

Information Extraction

Attack Results

Closed Source Evaluation

Conclusion
Technical Summary

This analysis of mixed countermeasures shows:
Technical Summary

This analysis of mixed countermeasures shows:

▶ Online attack in less than a minute with:
Technical Summary

This analysis of mixed countermeasures shows:

▶ Online attack in less than a minute with:
  ▶ With old state-of-the-art pdf estimation tools
Technical Summary

This analysis of mixed countermeasures shows:

- Online attack in less than a minute with:
  - With old state-of-the-art pdf estimation tools
  - Some equations depending on the countermeasures

- Preliminary leakage assessment found no weakness with 100,000 traces

- Difficulty to protect 32-bit software:
  - Inherent to low noise on the platform and not to optimized shuffling

- Knowledge needed to reproduce on other targets:
  - Source code and randomness knowledge during profiling
  - Sufficient understanding of countermeasures

- Not so much time!
Technical Summary

This analysis of mixed countermeasures shows:

- Online attack in less than a minute with:
  - With old state-of-the-art pdf estimation tools
  - Some equations depending on the countermeasures
  - Sounded hypotheses
Technical Summary

This analysis of mixed countermeasures shows:

► **Online attack in less than a minute with:**
  ▶ With old state-of-the-art pdf estimation tools
  ▶ Some equations depending on the countermeasures
  ▶ Sounded hypotheses

► **Preliminary leakage assessment found no weakness with 100,000 traces**
This analysis of mixed countermeasures shows:

- Online attack in less than a minute with:
  - With old state-of-the-art pdf estimation tools
  - Some equations depending on the countermeasures
  - Sounded hypotheses

- Preliminary leakage assessment found no weakness with 100,000 traces

- Difficulty to protect 32-bit software:
Technical Summary

This analysis of mixed countermeasures shows:

- Online attack in less than a minute with:
  - With old state-of-the-art pdf estimation tools
  - Some equations depending on the countermeasures
  - Sounded hypotheses

- Preliminary leakage assessment found no weakness with 100,000 traces

- Difficulty to protect 32-bit software:
  - Inherent to low noise on the platform and not to optimized shuffling
Technical Summary

This analysis of mixed countermeasures shows:

▶ Online attack in less than a minute with:
  ▶ With old state-of-the-art pdf estimation tools
  ▶ Some equations depending on the countermeasures
  ▶ Sounded hypotheses

▶ Preliminary leakage assessment found no weakness with 100,000 traces

▶ Difficulty to protect 32-bit software:
  ▶ Inherent to low noise on the platform and not to optimized shuffling

Knowledge needed to reproduce on other targets:
Technical Summary

This analysis of mixed countermeasures shows:

- Online attack in less than a minute with:
  - With old state-of-the-art pdf estimation tools
  - Some equations depending on the countermeasures
  - Sounded hypotheses

- Preliminary leakage assessment found no weakness with 100,000 traces

- Difficulty to protect 32-bit software:
  - Inherent to low noise on the platform and not to optimized shuffling

Knowledge needed to reproduce on other targets:

- Source code and randomness knowledge during profiling
Technical Summary

This analysis of mixed countermeasures shows:

▶ Online attack in less than a minute with:
  ▶ With old state-of-the-art pdf estimation tools
  ▶ Some equations depending on the countermeasures
  ▶ Sounded hypotheses

▶ Preliminary leakage assessment found no weakness with 100,000 traces
▶ Difficulty to protect 32-bit software:
  ▶ Inherent to low noise on the platform and not to optimized shuffling

Knowledge needed to reproduce on other targets:

▶ Source code and randomness knowledge during profiling
▶ Sufficient understanding of countermeasures
Technical Summary

This analysis of mixed countermeasures shows:

- Online attack in less than a minute with:
  - With old state-of-the-art pdf estimation tools
  - Some equations depending on the countermeasures
  - Sounded hypotheses

- Preliminary leakage assessment found no weakness with 100,000 traces

- Difficulty to protect 32-bit software:
  - Inherent to low noise on the platform and not to optimized shuffling

Knowledge needed to reproduce on other targets:

- Source code and randomness knowledge during profiling
- Sufficient understanding of countermeasures
- Not so much time!
Time Line
**Time Line**

- **Day 0**: Code is Online

![Scrolling Twitter](image)
Time Line

Day 0: Code is Online

Code Available
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it

Entering Hacker Mode
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it

Finding MCU
### Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it

---

Removing Capacitors
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it

Engraving EM Probe
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready

Entering Hacker Mode
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery

Really Happy
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery

Entering Hacker Mode
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks

Really Happy

Olivier Bronchain
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks

Entering Hacker Mode
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks
- **Day 11**: Key enumeration
**Time Line**

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks
- **Day 11**: Key enumeration

Really Happy
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks
- **Day 11**: Key enumeration

Entering Hacker Mode
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks
- **Day 11**: Key enumeration
- **Day 15**: Full attack
Time Line

- **Day 0**: Code is Online
- **Day 1**: Start looking at it
- **Day 5**: Setup ready
- **Day 6**: Multiplicative mask recovery
- **Day 10**: First attacks
- **Day 11**: Key enumeration
- **Day 15**: Full attack

Really Happy
Take Home Message

ANSSI’s implementation was a stimulating first step:
Take Home Message

ANSSI’s implementation was a stimulating first step:

▶ Nice research challenge to design/evaluate more secure implementations
Take Home Message

ANSSI’s implementation was a stimulating first step:

▶ Nice research challenge to design/evaluate more secure implementations
▶ Possibly dealing with limited physical noise
Take Home Message

ANSSI’s implementation was a stimulating first step:
- Nice research challenge to design/evaluate more secure implementations
- Possibly dealing with limited physical noise

Thanks!

Twitter: @BronchainO
email: olivier.bronchain@uclouvain.be