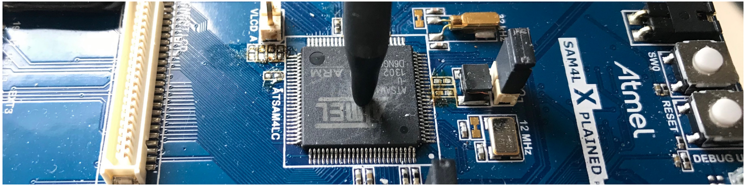


Worst-case side-channel security: from evaluation of countermeasures to new designs

Public Defense

Olivier Bronchain

21 janvier 2022

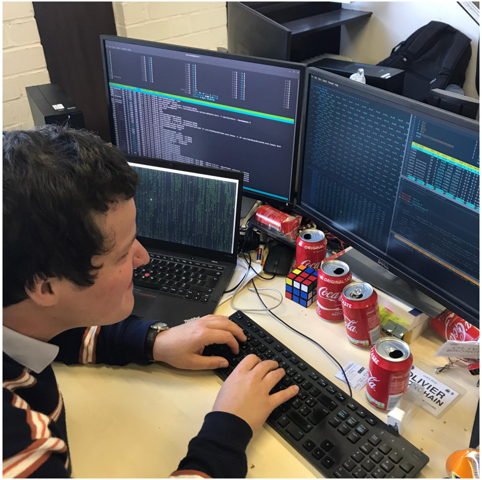
Contenu de la présentation

1. Résumé vulgarisé de ma thèse (en français).
2. Contenu technique (en anglais).

Content

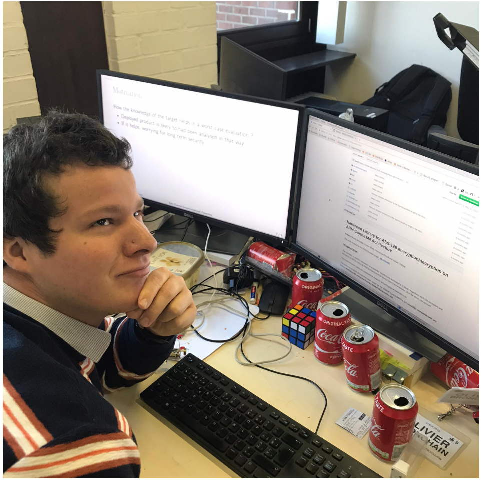
Qu'est-ce qu'une thèse ?

Qu'est-ce qu'une thèse ?



"Coder des trucs ?"

Qu'est-ce qu'une thèse ?

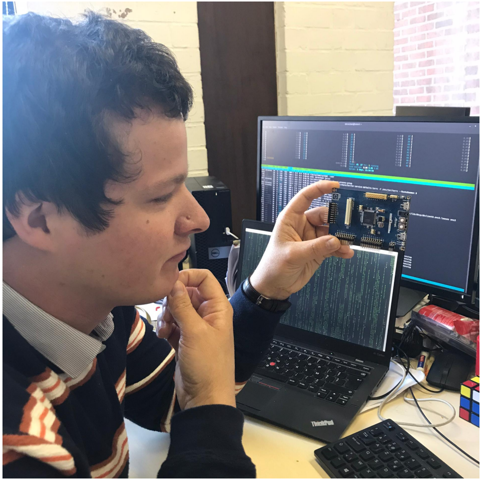


Chercher une bonne idée ?

Qu'est-ce qu'une thèse ?

Souder ?

Qu'est-ce qu'une thèse ?



Bidouiller des cartes électroniques ?

Qu'est-ce qu'une thèse ?

Ecouter de la musique à fond ?

Qu'est-ce qu'une thèse ?

Side-Channel Countermeasures' Dissection

and the Limits of Closed Source Security Evaluations

Olivier Bronchain and François-Xavier Standaert

ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium.
{olivier.bronchain,fstandae}@uclouvain.be

Abstract. We take advantage of a recently published open source implementation of the AES protected with a mix of countermeasures against side-channel attacks to discuss both the challenges in protecting COTS devices against such attacks and the limitations of closed source security evaluations. The target implementation has been proposed by the French ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) to stimulate research on the design and evaluation of side-channel secure implementations. It combines additive and multiplicative secret sharings into an affine masking scheme that is additionally mixed with a shuffled execution. Its preliminary leakage assessment did not detect data dependencies with up to 100,000 measurements. We first exhibit the gap between such a preliminary leakage assessment and advanced attacks by demonstrating how a countermeasures' dissection exploiting a mix of dimensionality reduction, multivariate information extraction and key enumeration can recover the full key with less than 2,000 measurements. We then discuss the relevance of open source evaluations to analyze such implementations efficiently, by pointing out that certain steps of the attack are hard to automate without implementation knowledge (even with machine learning tools), while performing them manually is straightforward. Our findings are not due to design flaws but from the general difficulty to prevent side-channel attacks in COTS devices with limited noise. We anticipate that high security on such devices requires significantly more shares.

Keywords: Side-Channel Attacks · Security Evaluations · Certification · Affine Masking · Shuffling · Worst-Case (Multivariate) Analysis · Open Source Design

Ecrire des articles de recherche ?

Qu'est-ce qu'une thèse ?

Voyager pour présenter des travaux ?

Qu'est-ce qu'une thèse ?

Une thèse: c'est un peu tout cela à la fois ...

... et c'est ce que je vais essayer de vous expliquer!

Ma thèse en quelques mots

Worst-case side-channel security: from evaluation of countermeasures to new designs

Ma thèse en quelques mots

Worst-case side-channel security: from evaluation of countermeasures to new designs

Décryptons le titre mot par mot:

Ma thèse en quelques mots

Worst-case side-channel security: from evaluation of countermeasures to new designs

Décryptons le titre mot par mot:

1. Security

Ma thèse en quelques mots

Worst-case side-channel security: from evaluation of countermeasures to new designs

Décryptons le titre mot par mot:

1. Security
2. Side-channel

Ma thèse en quelques mots

Worst-case side-channel security: from evaluation of countermeasures to new designs

Décryptons le titre mot par mot:

1. Security
2. Side-channel
3. Countermeasures

Ma thèse en quelques mots

Worst-case side-channel security: from evaluation of countermeasures to new designs

Décryptons le titre mot par mot:

1. Security
2. Side-channel
3. Countermeasures
4. Evaluation

Ma thèse en quelques mots

Worst-case side-channel security: from evaluation of countermeasures to new designs

Décryptons le titre mot par mot:

1. Security
2. Side-channel
3. Countermeasures
4. Evaluation
5. Worst-case

Worst-case side-channel security: from evaluation of countermeasures to new designs

Worst-case side-channel security: from evaluation of countermeasures to new designs

J'ai une super idée!!!

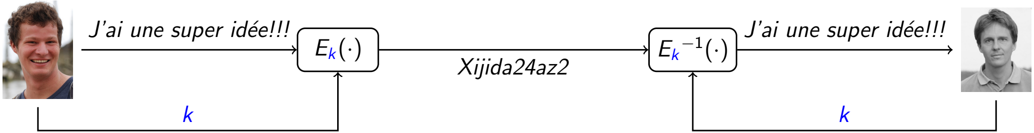
Worst-case side-channel security: from evaluation of countermeasures to new designs

J'ai une super idée!!!

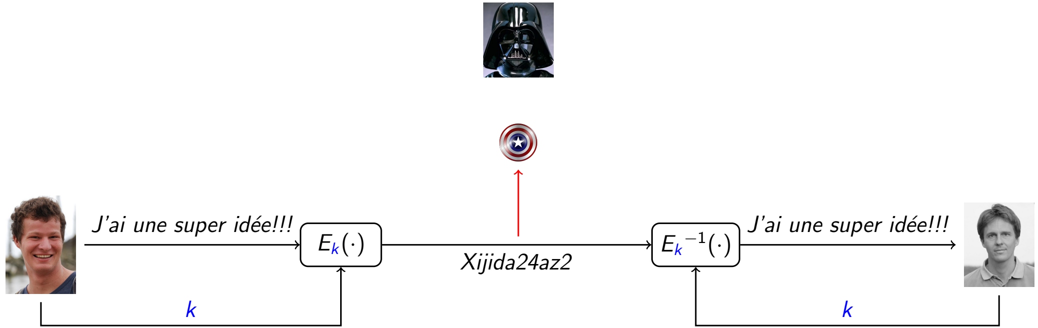
Worst-case side-channel security: from evaluation of countermeasures to new designs

J'ai une super idée!!! $\rightarrow$ $\boxed{E(\cdot)}$ $\xrightarrow{\textit{Xijida24az2}}$ $\boxed{E^{-1}(\cdot)}$ $\rightarrow$ *J'ai une super idée!!!*

Worst-case side-channel security: from evaluation of countermeasures to new designs



Worst-case side-channel security: from evaluation of countermeasures to new designs



Essayons une analogie

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 4953 ?

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 4953 ?
- 2391 ?

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 4953 ?
- 2391 ?
- 1234 ?

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 4953 ?
- 2391 ?
- 1234 ?

$\rightarrow$ Il teste une à une chacune des possibilités.

Worst-case side-channel security: from evaluation of countermeasures to new designs

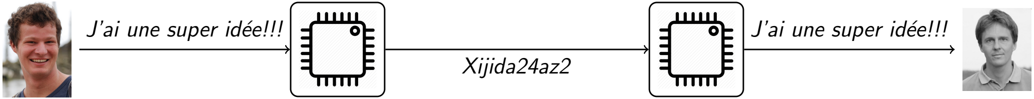
Worst-case side-channel security: from evaluation of countermeasures to new designs

J'ai une super idée!!!

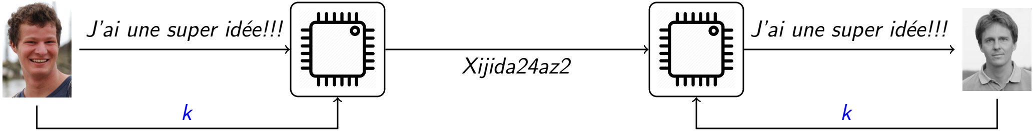
Worst-case side-channel security: from evaluation of countermeasures to new designs

J'ai une super idée!!!

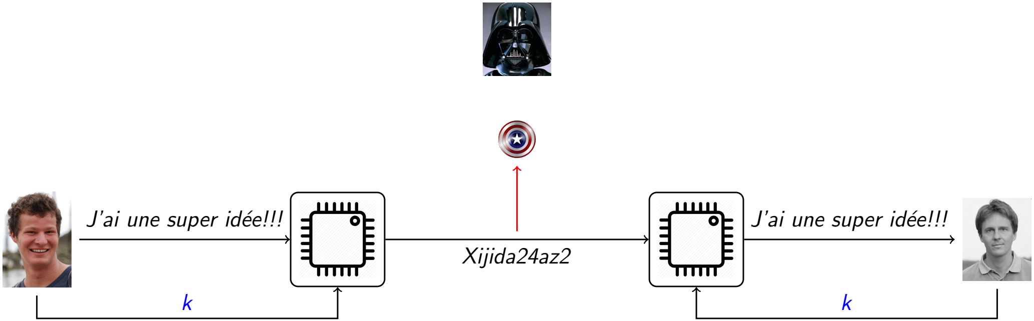
Worst-case side-channel security: from evaluation of countermeasures to new designs



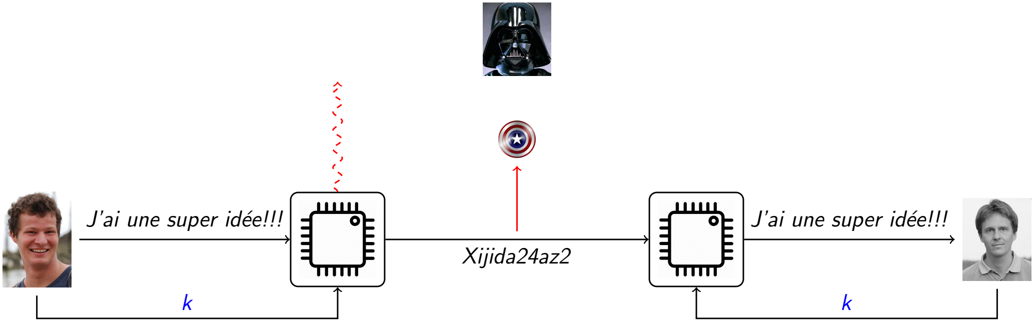
Worst-case side-channel security: from evaluation of countermeasures to new designs



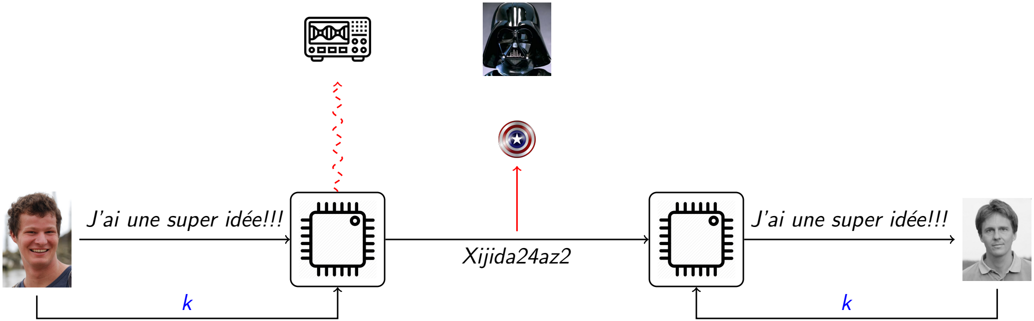
Worst-case side-channel security: from evaluation of countermeasures to new designs



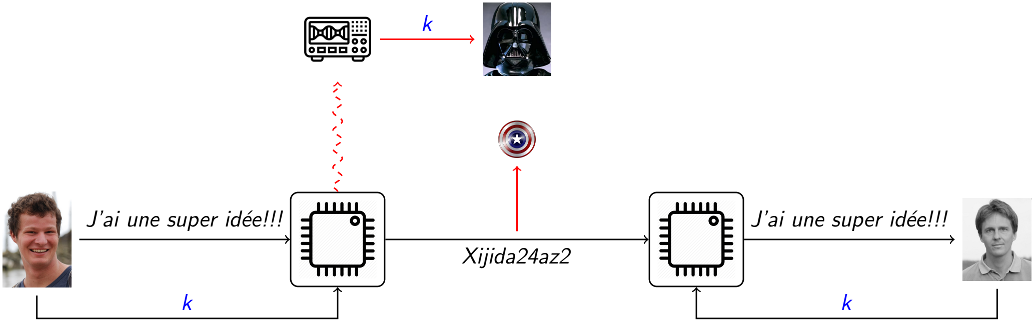
Worst-case side-channel security: from evaluation of countermeasures to new designs



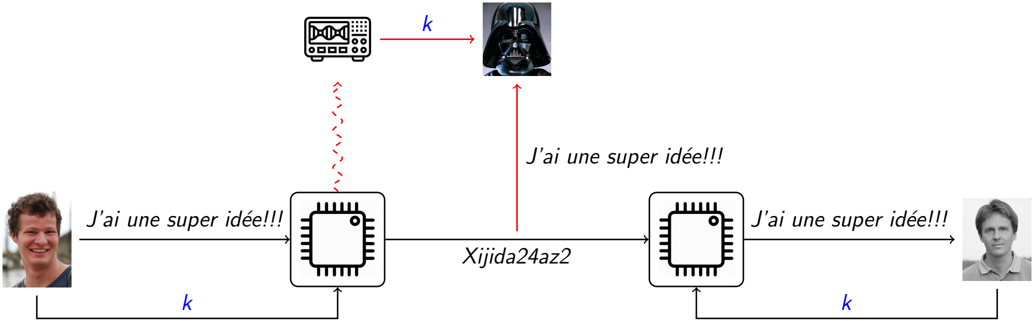
Worst-case side-channel security: from evaluation of countermeasures to new designs



Worst-case side-channel security: from evaluation of countermeasures to new designs



Worst-case side-channel security: from evaluation of countermeasures to new designs



Essayons une analogie

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 2268 ?

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 2268 ?
- 6628 ?

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 2268 ?
- 6628 ?
- 8826 ?

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 2268 ?
- 6628 ?
- 8826 ?

$\rightarrow$ Il teste une à une chacune des possibilités, avec uniquement des 2, des 6 ou des 8.

Essayons une analogie

- La sécurité: le digipass.
- La clé: un code à 4 chiffres.

Un adversaire devine la clé:

- 2268 ?
- 6628 ?
- 8826 ?

$\rightarrow$ Il teste une à une chacune des possibilités, avec uniquement des 2, des 6 ou des 8.
$\rightarrow$ Moins de sécurité.

Essayons une analogie

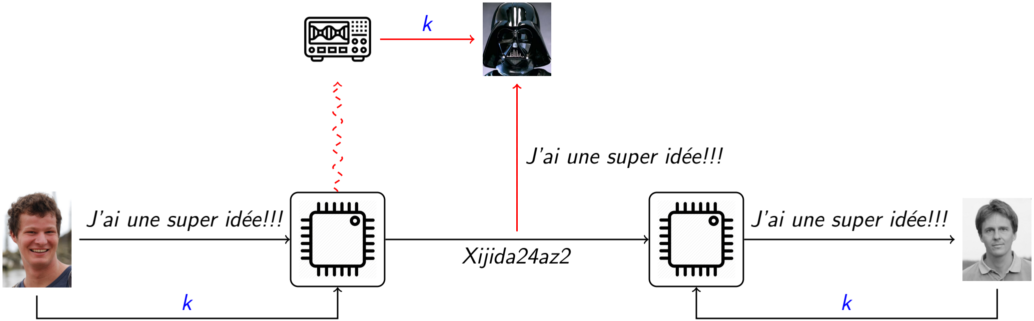
- La sécurité: le digipass.
- La clé: un code à 4 chiffres.
- Side-Channel: le chocolat.

Un adversaire devine la clé:

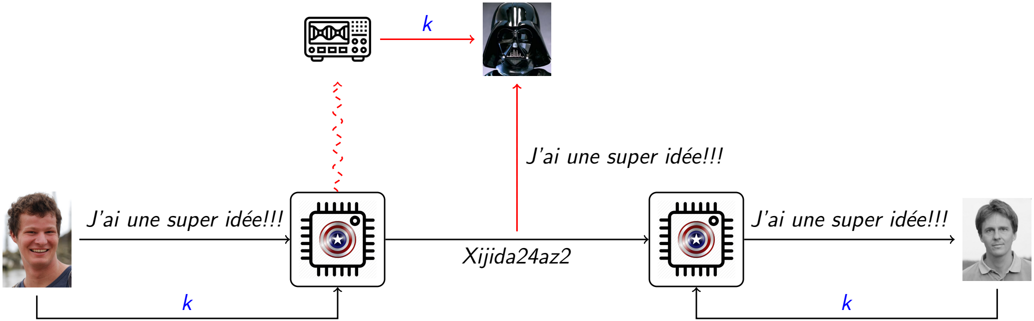
- 2268 ?
- 6628 ?
- 8826 ?

$\rightarrow$ Il teste une à une chacune des possibilités, avec uniquement des 2, des 6 ou des 8.
$\rightarrow$ Moins de sécurité.

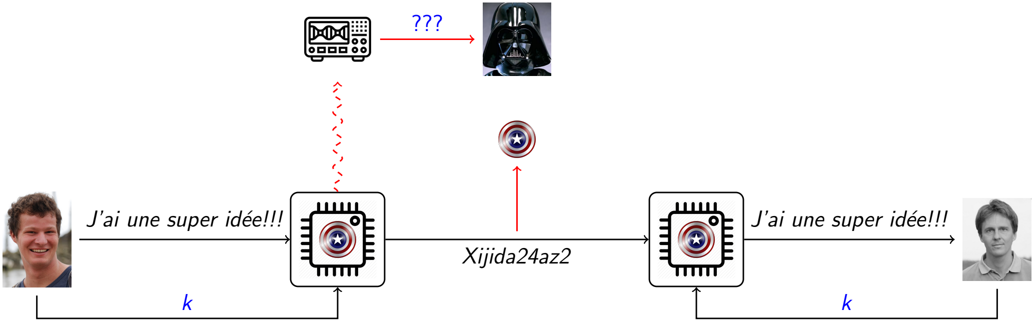
Worst-case side-channel security: from evaluation of countermeasures to new designs



Worst-case side-channel security: from evaluation of countermeasures to new designs



Worst-case side-channel security: from evaluation of countermeasures to new designs



Essayons une analogie

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.

$+$

$=$

Essayons une analogie

$+$

$=$

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure:...

Essayons une analogie

$+$

$=$

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure:... se laver les mains!

Essayons une analogie

+

=

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure:... se laver les mains!

Un adversaire devine la clé:

Essayons une analogie

+

=

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure:... se laver les mains!

Un adversaire devine la clé:

- 1028 ?

Essayons une analogie

+

=

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure:... se laver les mains!

Un adversaire devine la clé:

- 1028 ?
- 5742 ?

Essayons une analogie

+

=

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure:... se laver les mains!

Un adversaire devine la clé:

- 1028 ?
- 5742 ?
- 0743 ?

Essayons une analogie

+

=

- La sécurité: le digipass.
- Side-channel: → sécurité réduite.
- Contre-mesure:... se laver les mains!

Un adversaire devine la clé:

- 1028 ?
- 5742 ?
- 0743 ?

→ Pas d'autre solution que de tester une à une chacune des combinaisons.

Essayons une analogie

+

=

- La sécurité: le digipass.
- Side-channel: → sécurité réduite.
- Contre-mesure:... se laver les mains!

Un adversaire devine la clé:

- 1028 ?
- 5742 ?
- 0743 ?

→ Pas d'autre solution que de tester une à une chacune des combinaisons.
→ Retour de la sécurité!

Worst-case side-channel security: from evaluation of countermeasures to new designs

Vador va attaquer la contre-mesure ()!

Worst-case side-channel security: from evaluation of countermeasures to new designs

Vador va attaquer la contre-mesure ()!

Worst-case side-channel security: from evaluation of countermeasures to new designs

Vador va attaquer la contre-mesure ( )!

Worst-case side-channel security: from evaluation of countermeasures to new designs

Vador va attaquer la contre-mesure ()!

Mon travail:

- Évaluer la résistance de avant une éventuelle attaque.

Worst-case side-channel security: from evaluation of countermeasures to new designs

Vador va attaquer la contre-mesure ()!

Mon travail:

- Évaluer la résistance de avant une éventuelle attaque.

Une possibilité:

- Reproduire l'attaque dans mon laboratoire.

Worst-case side-channel security: from evaluation of countermeasures to new designs

Vador va attaquer la contre-mesure ()!

Mon travail:

- Évaluer la résistance de  avant une éventuelle attaque.

Une possibilité:

- Reproduire l'attaque dans mon laboratoire.

Essayons une analogie

Essayons une analogie

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure: se laver les mains.
- Évaluation: Reproduire l'attaque.

Essayons une analogie

- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure: se laver les mains.
- Évaluation: Reproduire l'attaque.

Se mettre à la place de l'adversaire: quelle stratégie pourrait-il adopter ?

Essayons une analogie

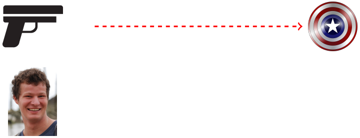
- La sécurité: le digipass.
- Side-channel: $\rightarrow$ sécurité réduite.
- Contre-mesure: se laver les mains.
- Évaluation: Reproduire l'attaque.

Se mettre à la place de l'adversaire: quelle stratégie pourrait-il adopter ?

- Utiliser une loupe pour détecter des traces de chocolat ?
- Fermer le robinet pour empêcher le lavage des mains ?

Worst-case side-channel security: from evaluation of countermeasures to new designs

Challenge:



Worst-case side-channel security: from evaluation of countermeasures to new designs

Challenge: Quid si Vador a une attaque que je ne sais pas reproduire dans mon laboratoire ?

Worst-case side-channel security: from evaluation of countermeasures to new designs

Challenge: Quid si Vador a une attaque que je ne sais pas reproduire dans mon laboratoire ?

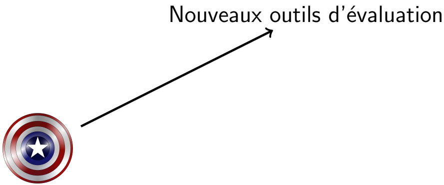
J'ai travaillé là dessus ! Comment faire ?

1. Faire une analyse sur une petite , et extrapoler pour des plus grosses.
2. Se baser sur des preuves mathématiques, et vérifier leurs hypothèses.

Worst-case side-channel security: from evaluation of countermeasures to new designs

Worst-case side-channel security: from evaluation of countermeasures to new designs

Worst-case side-channel security: from evaluation of countermeasures to new designs



Worst-case side-channel security: from evaluation of countermeasures to new designs

Nouveaux outils d'évaluation

La résistance est trop petit !!!

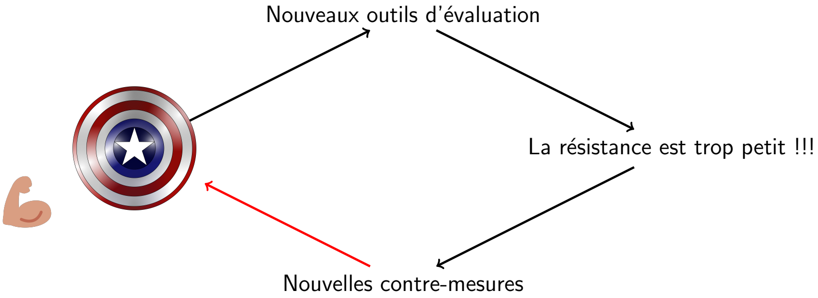
Worst-case side-channel security: from evaluation of countermeasures to new designs

Nouveaux outils d'évaluation

La résistance est trop petit !!!

Nouvelles contre-mesures

Worst-case side-channel security: from evaluation of countermeasures to new designs



Content

How to get SCA security with masking ?

Design goes in two steps:

1. Proofs in abstract models.
2. Use reductions to concrete (noisy) security.

How to get SCA security with masking ?

Design goes in two steps:

1. Proofs in abstract models.
2. Use reductions to concrete (noisy) security.

With masking:

1. Randomize the computation,
2. More randomness (shares) $\rightarrow$ more security.
3. More noise $\rightarrow$ more security

How to get SCA security with masking ?

Design goes in two steps:

1. Proofs in abstract models.
2. Use reductions to concrete (noisy) security.

With masking:

1. Randomize the computation,
2. More randomness (shares) $\rightarrow$ more security.
3. More noise $\rightarrow$ more security

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares

How to get SCA security with masking ?

Design goes in two steps:

1. Proofs in abstract models.
2. Use reductions to concrete (noisy) security.

With masking:

1. Randomize the computation,
2. More randomness (shares) $\rightarrow$ more security.
3. More noise $\rightarrow$ more security

Two physical conditions:

1. **Independence**: ensures d.
2. **Noise**: small enough MI.

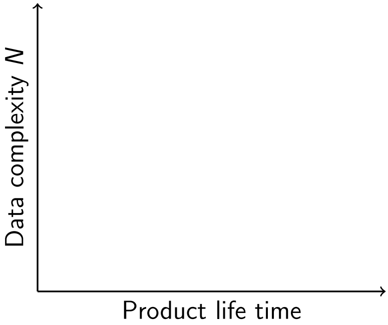
Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

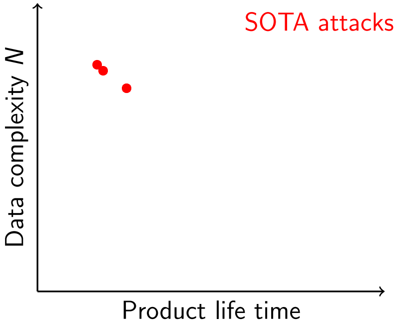
Number of shares

Part I & II: goal of security evaluations



Timeline for secure implementations:

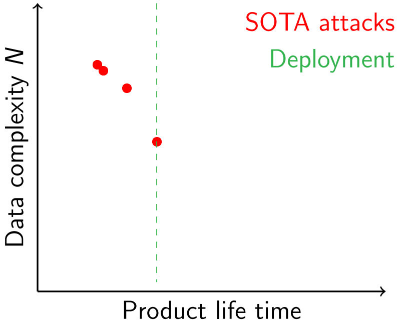
Part I & II: goal of security evaluations



Timeline for secure implementations:

- Apply current SOTA attacks.
- SOTA improvements with time.

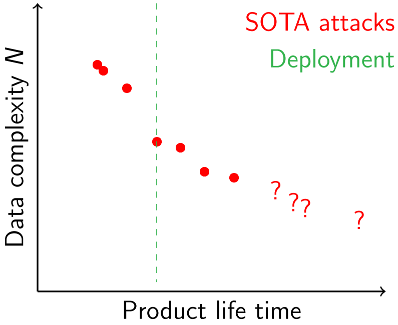
Part I & II: goal of security evaluations



Timeline for secure implementations:

- Apply current SOTA attacks.
- SOTA improvements with time.
- Deploy the implementation.

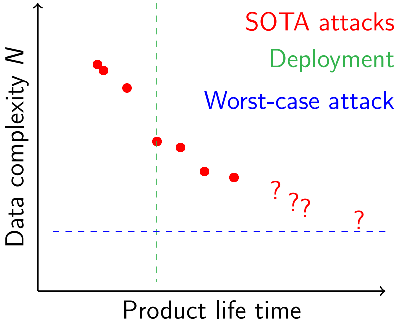
Part I & II: goal of security evaluations



Timeline for secure implementations:

- Apply current SOTA attacks.
- SOTA improvements with time.
- Deploy the implementation.
- Attack still improves.

Part I & II: goal of security evaluations



Timeline for secure implementations:

- Apply current SOTA attacks.
- SOTA improvements with time.
- Deploy the implementation.
- Attack still improves.

How to anticipate future SOTA with **worst-case** attacks ?

Ingredients for (close-to) worst-case evaluations

Evaluator benefits from open-source (during profiling/learning):

- Knowledge of the source code.
- Knowledge of the randomness $\mathcal{R}$.
- Measurement setup optimization.
- Exploit as much as possible the information from leakages.

Ingredients for (close-to) worst-case evaluations

Evaluator benefits from open-source (during profiling/learning):

- Knowledge of the source code.
- Knowledge of the randomness $\mathcal{R}$.
- Measurement setup optimization.
- Exploit as much as possible the information from leakages.

Why profiling with $\mathcal{R}$?

- Simpler profiling and easier interpretation.
- Provide design guidelines.

Ingredients for (close-to) worst-case evaluations

Evaluator benefits from open-source (during profiling/learning):

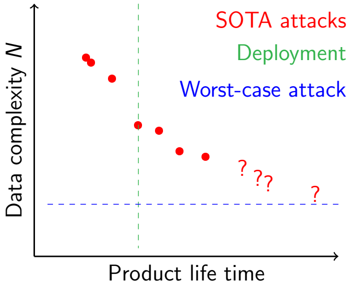
- Knowledge of the source code.
- Knowledge of the randomness $\mathcal{R}$.
- Measurement setup optimization.
- Exploit as much as possible the information from leakages.

Why profiling with $\mathcal{R}$?

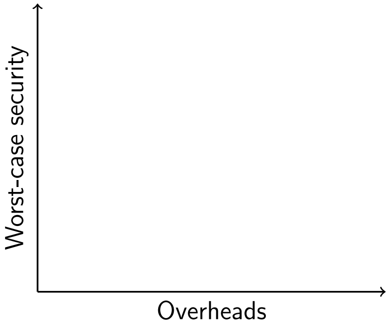
- Simpler profiling and easier interpretation.
- Provide design guidelines.

Why worst considering ?

- Short term: gap vs profiling w/o $\mathcal{R}$.
- Long term: gap expected to vanish.



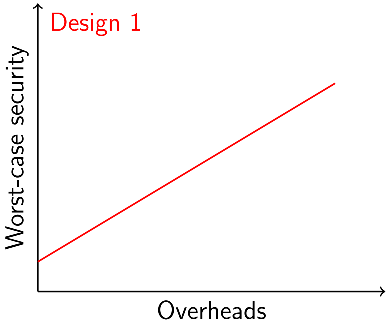
Part III: goal of designer



How to compare (secure) implementations?

- Security vs. performance curves.

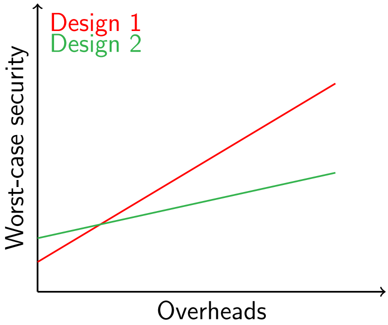
Part III: goal of designer



How to compare (secure) implementations?

- Security vs. performance curves.

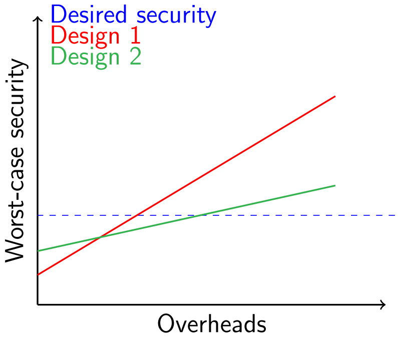
Part III: goal of designer



How to compare (secure) implementations?

- Security vs. performance curves.

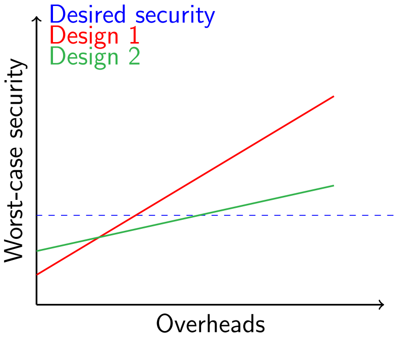
Part III: goal of designer



How to compare (secure) implementations?

- Security vs. performance curves.
- Define security target.

Part III: goal of designer



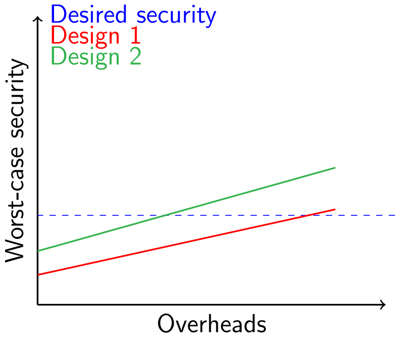
How to compare (secure) implementations?

- Security vs. performance curves.
- Define security target.

Challenges:

- What performance metric ?
- What platform / context ?

Part III: goal of designer



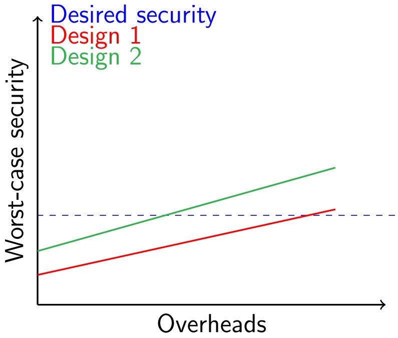
How to compare (secure) implementations?

- Security vs. performance curves.
- Define security target.

Challenges:

- What performance metric ?
- What platform / context ?

Part III: goal of designer



How to compare (secure) implementations?

- Security vs. performance curves.
- Define security target.

Challenges:

- What performance metric ?
- What platform / context ?

How & When to combine countermeasures ?

Content

Proof-based: general principle (Part I)

Masking countermeasure requires:

1. **Independence**: ensures d.
2. **Noise**: small enough MI.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares

Proof-based: general principle (Part I)

Masking countermeasure requires:

1. **Independence**: ensures d.
2. **Noise**: small enough MI.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share Number of shares

Proof-based evaluation:

Proof-based: general principle (Part I)

Masking countermeasure requires:

1. **Independence**: ensures d.
2. **Noise**: small enough MI.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares

Proof-based evaluation:

1. **Leakage detection:** convinces that independence holds.

Proof-based: general principle (Part I)

Masking countermeasure requires:

1. **Independence**: ensures d.
2. **Noise**: small enough MI.

$$\overbrace{N}^{\text{Data complexity}} \geq \frac{c}{\underbrace{\mathrm{MI}(X_i;L)}_{\text{Information per share}}{}^{\overbrace{d}^{\text{Number of shares}}}}$$

Proof-based evaluation:

1. **Leakage detection:** convinces that independence holds.
2. **Information estimation:** ensures small MI.

Proof-based: general principle (Part I)

Masking countermeasure requires:

1. **Independence**: ensures d.
2. **Noise**: small enough MI.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares

Proof-based evaluation:

1. **Leakage detection:** convinces that independence holds.
2. **Information estimation:** ensures small MI.
3. Estimation of attack complexity N.

Proof-based: independence (Chap. 3)

Leakage detection:

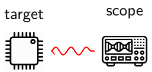
- Test for independence between statistical order of the leakage and secret data.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares



Proof-based: independence (Chap. 3)

Leakage detection:

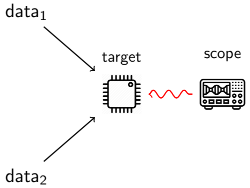
- Test for independence between statistical order of the leakage and secret data.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^{d}}$$

Information per share

Number of shares



Proof-based: independence (Chap. 3)

Leakage detection:

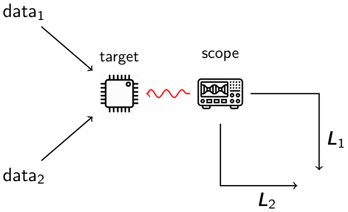
- Test for independence between statistical order of the leakage and secret data.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares



Proof-based: independence (Chap. 3)

Leakage detection:

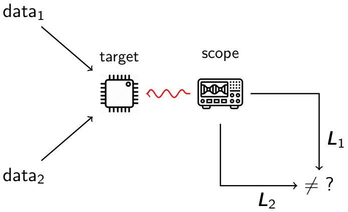
- Test for independence between statistical order of the leakage and secret data.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^{d}}$$

Information per share

Number of shares



Proof-based: independence (Chap. 3)

Leakage detection:

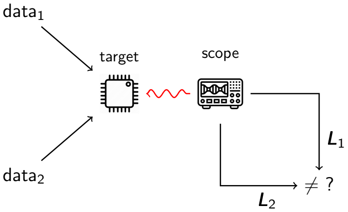
- Test for independence between statistical order of the leakage and secret data.
- **cons**: qualitative, false positive/negative, badly scales with d.

Data complexity

$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^{d}}$$

Information per share

Number of shares



Proof-based: independence (Chap. 3)

Leakage detection:

- Test for independence between statistical order of the leakage and secret data.
- **cons**: qualitative, false positive/negative, badly scales with d.

Data complexity

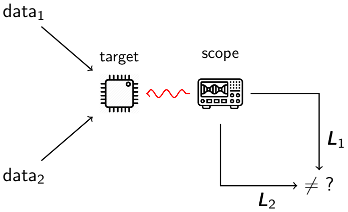
$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares

Contribution:

- Multivariate statistical tests:
  - + Faster detection.
  - + Better control of false negative.



Proof-based: independence (Chap. 3)

Leakage detection:

- Test for independence between statistical order of the leakage and secret data.
- **cons**: qualitative, false positive/negative, badly scales with d.

Data complexity

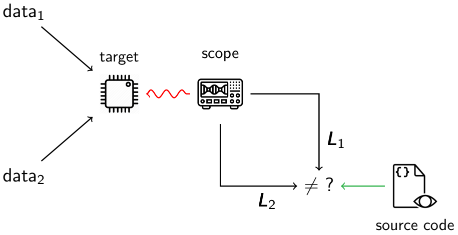
$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^{d}}$$

Information per share

Number of shares

Contribution:

- Multivariate statistical tests:
  - + Faster detection.
  - + Better control of false negative.
- Knowledge of source useful.



Proof-based: noise quantification (Chap. 4)

Data complexity

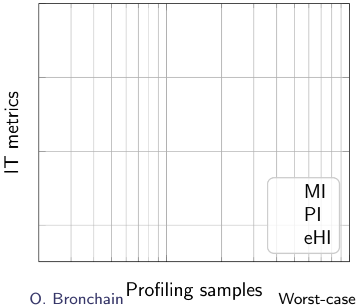
$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares

THEOREM

$\geq$ $\geq$



Proof-based: noise quantification (Chap. 4)

Data complexity

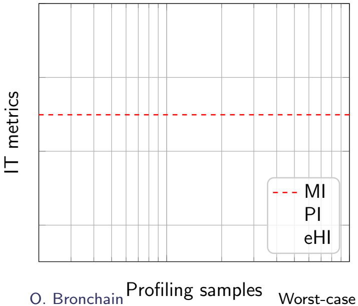
$$N \geq \frac{c}{\mathrm{MI}(X_i;L)^d}$$

Information per share

Number of shares

Theorem

$$\geq \mathrm{MI}(X_i;L) \geq$$

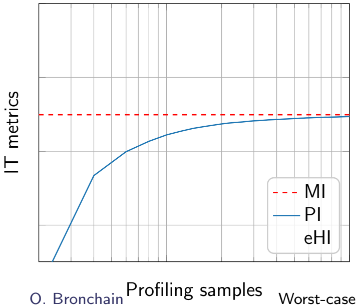


Proof-based: noise quantification (Chap. 4)

$$\underbrace{N}_{\text{Data complexity}} \geq \frac{c}{\underbrace{\mathrm{MI}(X_i;L)}_{\text{Information per share}}^{\overbrace{d}^{\text{Number of shares}}}}$$

Theorem

$$\geq \mathrm{MI}(X_i;L) \geq \mathrm{PI}(X_i;L)$$



Proof-based: noise quantification (Chap. 4)

Data complexity

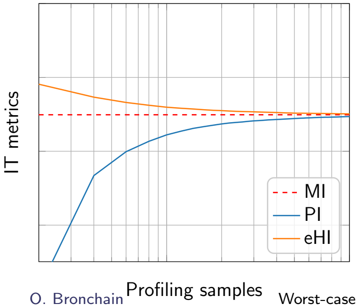
$$N \geq \frac{c}{\textcolor{red}{\mathrm{MI}(X_i;L)}^d}$$

Information per share

Number of shares

Theorem

$$\mathrm{eHI}(X_i;L) \geq \textcolor{red}{\mathrm{MI}(X_i;L)} \geq \mathrm{PI}(X_i;L)$$



Proof-based: noise quantification (Chap. 4)

Data complexity

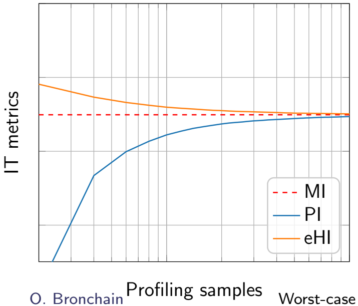
$$N \geq \frac{c}{\color{red}{\mathrm{MI}(X_i;L)}^d}$$

Information per share

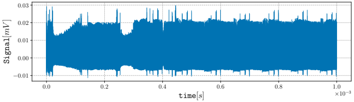
Number of shares

Theorem

$$\mathrm{eHI}(X_i;L) \geq \color{red}{\mathrm{MI}(X_i;L)} \geq \mathrm{PI}(X_i;L)$$



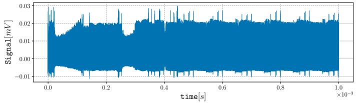
Attack-based: General profiling methodology (Part II)



Goal:

- Learn leakage distribution for secret x.

Attack-based: General profiling methodology (Part II)



Goal:

- Learn leakage distribution for secret x.

How ?

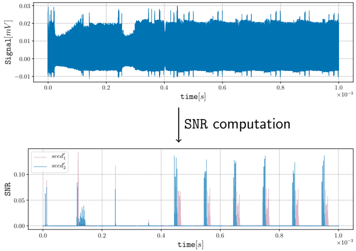
Attack-based: General profiling methodology (Part II)

Goal:

- Learn leakage distribution for secret x.

How ?

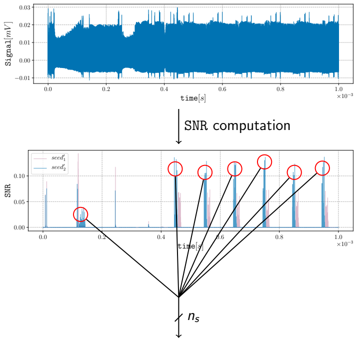
1. Compute SNR for variable x.



Attack-based: General profiling methodology (Part II)

Goal:

- Learn leakage distribution for secret x.



How ?

1. Compute SNR for variable x.
2. Extract Point-of-Interests (POIs).

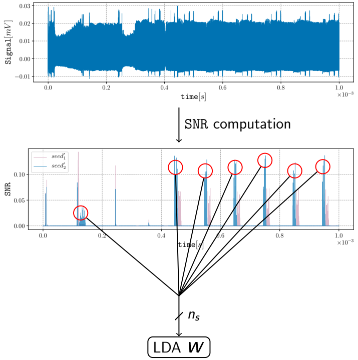
Attack-based: General profiling methodology (Part II)

Goal:

- Learn leakage distribution for secret x.

How ?

1. Compute SNR for variable x.
2. Extract Point-of-Interests (POIs).
3. Project to linear subspace $\boldsymbol{W}$.



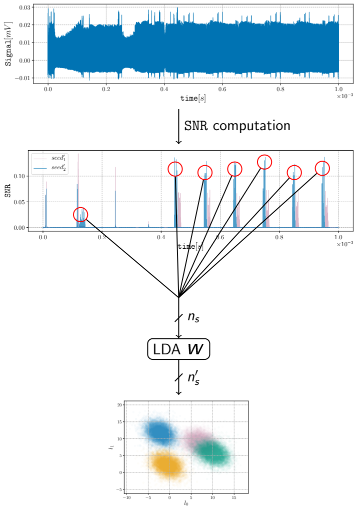
Attack-based: General profiling methodology (Part II)

Goal:

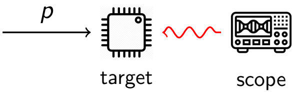
- Learn leakage distribution for secret x.

How ?

1. Compute SNR for variable x.
2. Extract Point-of-Interests (POIs).
3. Project to linear subspace $\boldsymbol{W}$.
4. Approximate distributions with Gaussian assumption.

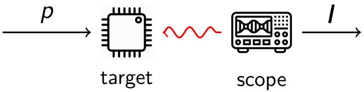


Attack-based: General attack methodology (Part II)



How ?

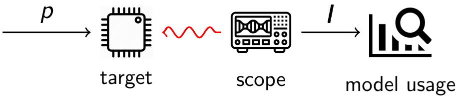
Attack-based: General attack methodology (Part II)



How ?

1. Record traces $\boldsymbol{l}$.

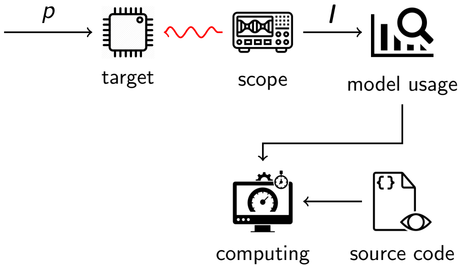
Attack-based: General attack methodology (Part II)



How ?

1. Record traces $\boldsymbol{l}$.
2. Gain information about x's with model.

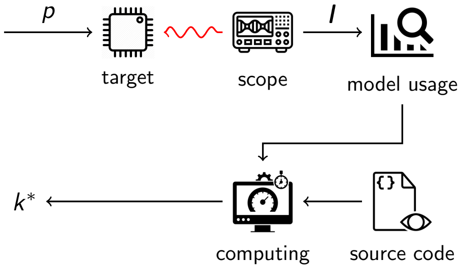
Attack-based: General attack methodology (Part II)



How ?

1. Record traces $\boldsymbol{l}$.
2. Gain information about x's with model.
3. Process information.

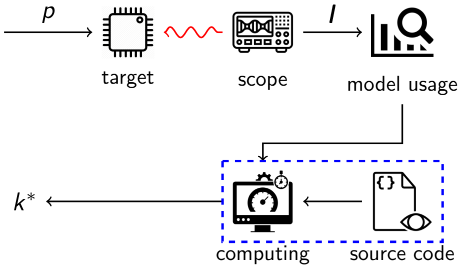
Attack-based: General attack methodology (Part II)



How ?

1. Record traces $\boldsymbol{l}$.
2. Gain information about x's with model.
3. Process information.
4. Output a key guess k^*.

Attack-based: General attack methodology (Part II)



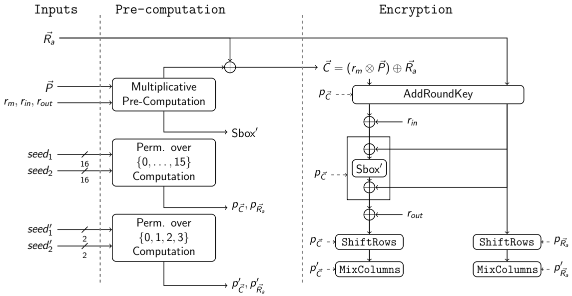
How ?

1. Record traces $\boldsymbol{l}$.
2. Gain information about x's with model.
3. Process information.
4. Output a key guess k^*.

Efficient processing requires:

- Trade-off between information exploitation and computational cost.
- Careful case-specific analysis.

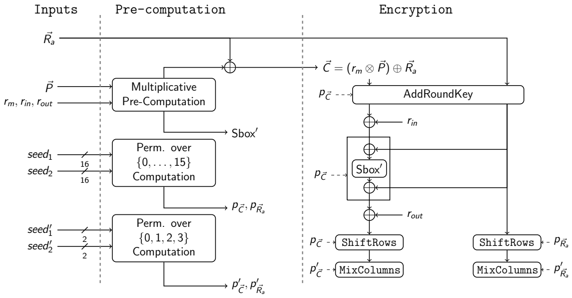
Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



Implementation has:

- Multiplicative masking.
- Boolean masking.
- Shuffling.

Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



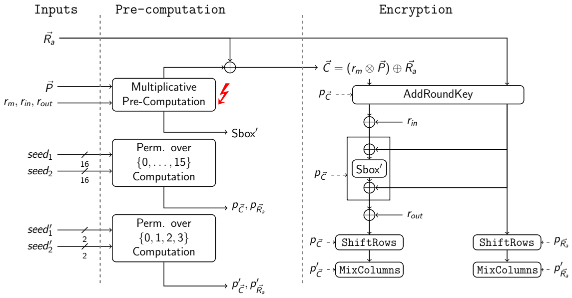
Implementation has:

- ~~Multiplicative masking~~.
- Boolean masking.
- ~~Shuffling~~.

Tricks:

- Attack countermeasures independently.

Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



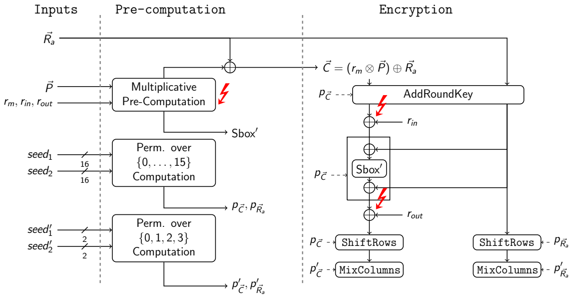
Implementation has:

- ~~Multiplicative masking~~.
- Boolean masking.
- ~~Shuffling~~.

Tricks:

- Attack countermeasures independently.
- Identify easier attack paths.

Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



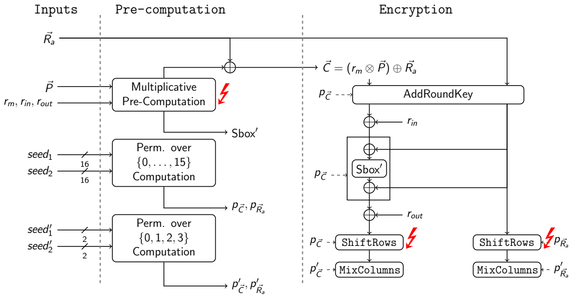
Implementation has:

- ~~Multiplicative masking~~.
- Boolean masking.
- ~~Shuffling~~.

Tricks:

- Attack countermeasures independently.
- Identify easier attack paths.

Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



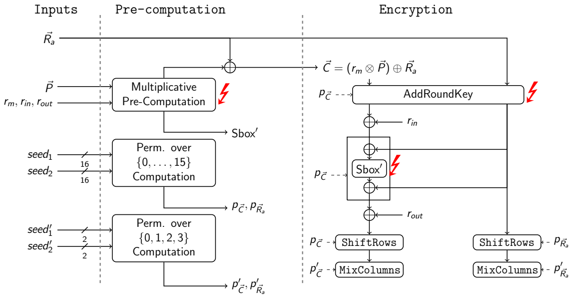
Implementation has:

- ~~Multiplicative masking~~.
- Boolean masking.
- ~~Shuffling~~.

Tricks:

- Attack countermeasures independently.
- Identify easier attack paths.

Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



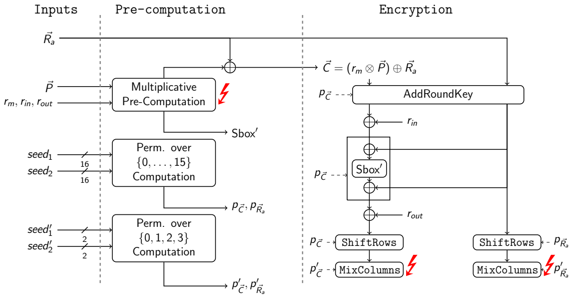
Implementation has:

- ~~Multiplicative masking~~.
- Boolean masking.
- ~~Shuffling~~.

Tricks:

- Attack countermeasures independently.
- Identify easier attack paths.

Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



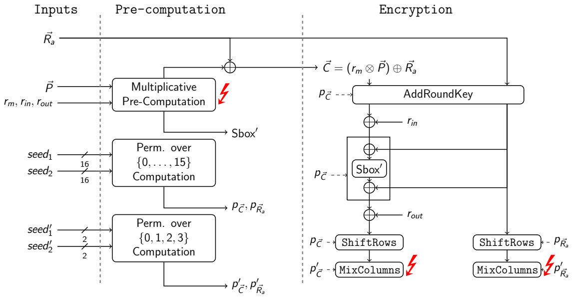
Implementation has:

- ~~Multiplicative masking~~.
- Boolean masking.
- ~~Shuffling~~.

Tricks:

- Attack countermeasures independently.
- Identify easier attack paths.

Attack-based: ANSSI's AES (32-bit MCU) (Chap. 6)



Implementation has:

- ~~Multiplicative masking~~.
- Boolean masking.
- ~~Shuffling~~.

Tricks:

- Attack countermeasures independently.
- Identify easier attack paths.

Summary of the results:

- (Close-to) worst-case attacks succeed in less than 1,000 traces.
- No known attacks in black-box settings.
- Low noise on 32-bit software.

Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Contributions:

- Using SASCA.
- With dedicated factor graphs.
- Reducing computational complexity.

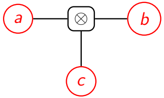
Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Contributions:

- Using SASCA.
- With dedicated factor graphs.
- Reducing computational complexity.



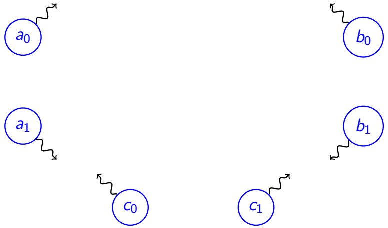
Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Contributions:

- Using SASCA.
- With dedicated factor graphs.
- Reducing computational complexity.



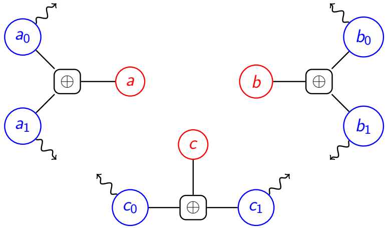
Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Contributions:

- Using SASCA.
- With dedicated factor graphs.
- Reducing computational complexity.



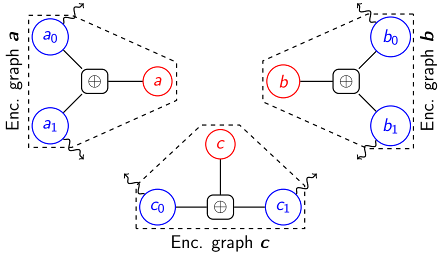
Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Contributions:

- Using SASCA.
- With dedicated factor graphs.
- Reducing computational complexity.



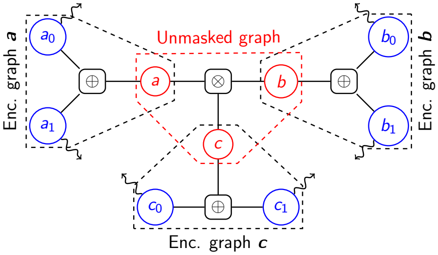
Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Contributions:

- Using SASCA.
- With dedicated factor graphs.
- Reducing computational complexity.



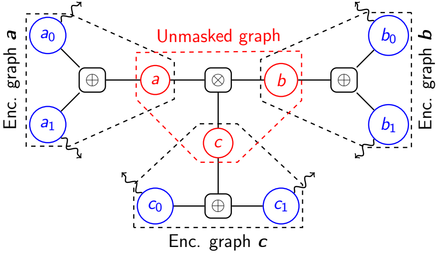
Attack-based: Bitslice masking (32-bit MCU) (Chap. 7)

Implementation has:

- Bitslice boolean masking with large d.

Contributions:

- Using SASCA.
- With dedicated factor graphs.
- Reducing computational complexity.



Summary of the results:

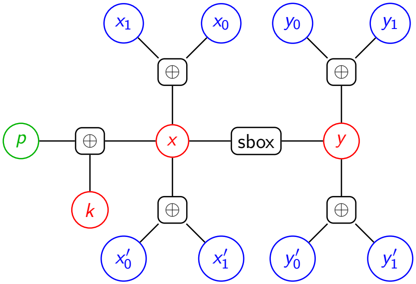
- Able to break 8 shares versions of lightweight ciphers.
- Low noise on 32-bit software.

Attack-based: ASCAD (8-bit MCU) (Chap. 5)

Implementation has:

- 2-shares table-based boolean masking.
- 8-bit MCU.

Attack-based: ASCAD (8-bit MCU) (Chap. 5)



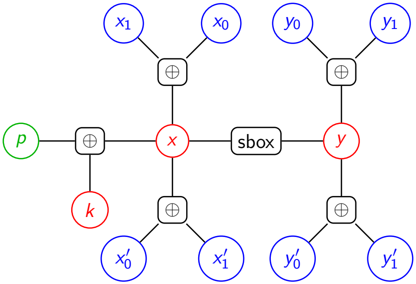
Implementation has:

- 2-shares table-based boolean masking.
- 8-bit MCU.

Tricks:

- Same as for bitslice masked software.
- Exploiting full leakage traces.

Attack-based: ASCAD (8-bit MCU) (Chap. 5)



Implementation has:

- 2-shares table-based boolean masking.
- 8-bit MCU.

Tricks:

- Same as for bitslice masked software.
- Exploiting full leakage traces.

Results summary:

- Recover full key with a single (raw) trace.
- Thanks to $\mathcal{R}$: Better offline & online attack performances.

SCALib: an open-source library [1] (Part II)

Welcome to SCALib

pypi package 0.3.3 docs passing

The Side-Channel Analysis Library (SCALib) is a Python package that contains state-of-the-art tools for side-channel evaluation. It focuses on providing efficient implementations of analysis methods widely used by the side-channel community and maintaining a flexible and simple interface.

[1] Developped with Gaëtan Cassiers

SCALib: an open-source library [1] (Part II)

Welcome to SCALib

pypi package 0.3.3 docs passing

The Side-Channel Analysis Library (SCALib) is a Python package that contains state-of-the-art tools for side-channel evaluation. It focuses on providing efficient implementations of analysis methods widely used by the side-channel community and maintaining a flexible and simple interface.

What is SCALib?

- Python: `pip install scalib`.
- Multiple tools for SCA.
- Optimized for single / multiple thread(s) (Rust back-end).

[1] Developped with Gaëtan Cassiers

SCALib: an open-source library [1] (Part II)

Welcome to SCALib

pypi package 0.3.3 docs passing

The Side-Channel Analysis Library (SCALib) is a Python package that contains state-of-the-art tools for side-channel evaluation. It focuses on providing efficient implementations of analysis methods widely used by the side-channel community and maintaining a flexible and simple interface.

What is SCALib?

- Python: `pip install scalib`.
- Multiple tools for SCA.
- Optimized for single / multiple thread(s) (Rust back-end).

Developed for this thesis to:

- Signal-to-Noise Ratio (SNR).
- Build LDA + Gaussian templates.
- SASCA.
- Key rank estimation.

[1]Developped with Gaëtan Cassiers

Content

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

$\overbrace{a^0}$ $\overbrace{b^0}$ $\overbrace{c^0}$

$\underbrace{a^1}$ $\underbrace{b^1}$ $\underbrace{c^1}$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

a^0

$\#$ random perm:0

θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

a^0

a^1

$\#$ random perm:0

θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

a^0 b^0

a^1

\# random perm:0

θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

a^0 b^0

a^1 b^1

random perm:0

θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

$\widehat{a^0}$ $\widehat{b^0}$ $\widehat{c^0}$ # random perm:0

$\underbrace{a^1}$ $\underbrace{b^1}$

θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

$\overset{\frown}{a^0}$ $\overset{\frown}{b^0}$ $\overset{\frown}{c^0}$ # random perm:0

$\underset{\smile}{a^1}$ $\underset{\smile}{b^1}$ $\underset{\smile}{c^1}$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

$\overbrace{a^0}$ $\overbrace{b^0}$ $\overbrace{c^0}$ # random perm:0

$\underbrace{a^1}$ $\underbrace{b^1}$ $\underbrace{c^1}$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

random perm:1

θ : [ ]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

$a^0 \quad b^0 \quad c^0$ # random perm:0

$a^1 \quad b^1 \quad c^1$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

random perm:1

θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

$\overset{\frown}{a^0}$ $\overset{\frown}{b^0}$ $\overset{\frown}{c^0}$ # random perm:0

$\underset{\smile}{a^1}$ $\underset{\smile}{b^1}$ $\underset{\smile}{c^1}$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

$\overset{\frown}{b^0}$ # random perm:1

θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

b^0 # random perm:1

b^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 # random perm:1

b^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 # random perm:1

a^1 b^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:1

a^1 b^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

$\overset{\frown}{a^0}$ $\overset{\frown}{b^0}$ $\overset{\frown}{c^0}$ # random perm:0

$\underset{\smile}{a^1}$ $\underset{\smile}{b^1}$ $\underset{\smile}{c^1}$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

$\overset{\frown}{a^0}$ $\overset{\frown}{b^0}$ $\overset{\frown}{c^0}$ # random perm:1

$\underset{\smile}{a^1}$ $\underset{\smile}{b^1}$ $\underset{\smile}{c^1}$ θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:1

a^1 b^1 c^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

---

random perm:d

θ : [ ]

Shuffling shares:

- Access i-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:1

a^1 b^1 c^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

---

random perm:d

θ : [2 0 1]

Shuffling shares:

- Access i-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

$a^0 \quad b^0 \quad c^0$ # random perm:0

$a^1 \quad b^1 \quad c^1$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

$a^0 \quad b^0 \quad c^0$ # random perm:1

$a^1 \quad b^1 \quad c^1$ θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

---

c^0 # random perm:d

θ : [2 0 1]

Shuffling shares:

- Access i-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:1

a^1 b^1 c^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

---

a^0 c^0 # random perm:d

θ : [2 0 1]

Shuffling shares:

- Access i-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:1

a^1 b^1 c^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:d

θ : [2 0 1]

Shuffling shares:

- Access i-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

$a^0 \quad b^0 \quad c^0$ # random perm:0

$a^1 \quad b^1 \quad c^1$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

$a^0 \quad b^0 \quad c^0$ # random perm:1

$a^1 \quad b^1 \quad c^1$ θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

$a^0 \quad b^0 \quad c^0$ # random perm:d

θ : [1 2 0]

Shuffling shares:

- Access i-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

$\overbrace{a^0}$ $\overbrace{b^0}$ $\overbrace{c^0}$ # random perm:0

$\underbrace{a^1}$ $\underbrace{b^1}$ $\underbrace{c^1}$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

$\overbrace{a^0}$ $\overbrace{b^0}$ $\overbrace{c^0}$ # random perm:1

$\underbrace{a^1}$ $\underbrace{b^1}$ $\underbrace{c^1}$ θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

$\overbrace{a^0}$ $\overbrace{b^0}$ $\overbrace{c^0}$ # random perm:d

$\underbrace{b^1}$ θ : [1 2 0]

Shuffling shares:

- Access i-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? ($d = 2$, $\eta = 3$) (Part III)

$\overset{\frown}{a^0}$ $\overset{\frown}{b^0}$ $\overset{\frown}{c^0}$ # random perm:0

$\underset{\smile}{a^1}$ $\underset{\smile}{b^1}$ $\underset{\smile}{c^1}$ θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

$\overset{\frown}{a^0}$ $\overset{\frown}{b^0}$ $\overset{\frown}{c^0}$ # random perm:1

$\underset{\smile}{a^1}$ $\underset{\smile}{b^1}$ $\underset{\smile}{c^1}$ θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

$\overset{\frown}{a^0}$ $\overset{\frown}{b^0}$ $\overset{\frown}{c^0}$ # random perm:d

$\underset{\smile}{b^1}$ $\underset{\smile}{c^1}$ θ : [1 2 0]

Shuffling shares:

- Access *i*-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

How to masking + shuffling ? $(d = 2, \eta = 3)$ (Part III)

a^0 b^0 c^0 # random perm:0

a^1 b^1 c^1 θ : [0 1 2]

No shuffling:

- Access shares sequentially.
- Security: $\mathcal{O}(\frac{1}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:1

a^1 b^1 c^1 θ : [1 0 2]

Shuffling tuples:

- Access tuples randomly.
- Security: $\mathcal{O}(\frac{\eta}{\mathrm{MI}^d})$

---

a^0 b^0 c^0 # random perm:d

a^1 b^1 c^1 θ : [1 2 0]

Shuffling shares:

- Access *i*-th shares randomly.
- Security: $\mathcal{O}(\frac{\eta^d}{\mathrm{MI}^d})$

Exploring design space: when to mask+shuffle ? (Part III)

Exploring design space: when to mask+shuffle ? (Part III)

Explored parameters:

- MI: per share.
- $\#AND$: Number of bits to protect.
- b_s: bitslicing size.
- r: randomness cost.

Exploring design space: when to mask+shuffle ? (Part III)

Explored parameters:

- MI: per share.
- $\#AND$: Number of bits to protect.
- b_s: bitslicing size.
- r: randomness cost.

Cycles vs. security comparison:

- For r, $\#AND$ and MI.
- ■: masking is faster.
- □: masking + shuffling is faster.

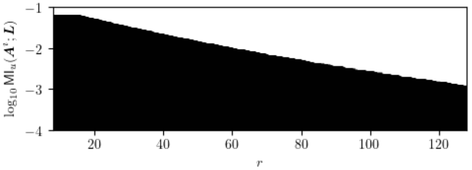


Figure: #AND=128

Exploring design space: when to mask+shuffle ? (Part III)

Explored parameters:

- MI: per share.
- $\#AND$: Number of bits to protect.
- b_s: bitslicing size.
- r: randomness cost.

Cycles vs. security comparison:

- For r, $\#AND$ and MI.
- ■: masking is faster.
- □: masking + shuffling is faster.

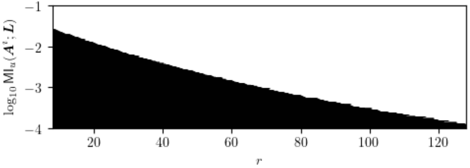


Figure: #AND=256

Exploring design space: when to mask+shuffle ? (Part III)

Explored parameters:

- MI: per share.
- $\#AND$: Number of bits to protect.
- b_s: bitslicing size.
- r: randomness cost.

Cycles vs. security comparison:

- For r, $\#AND$ and MI.
- ■: masking is faster.
- □: masking + shuffling is faster.

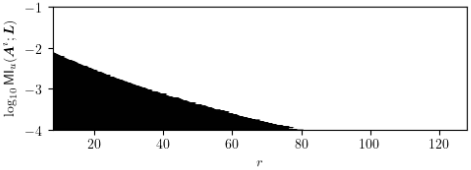


Figure: #AND=512

Exploring design space: when to mask+shuffle ? (Part III)

Explored parameters:

- MI: per share.
- $\#AND$: Number of bits to protect.
- b_s: bitslicing size.
- r: randomness cost.

Cycles vs. security comparison:

- For r, $\#AND$ and MI.
- ■: masking is faster.
- □: masking + shuffling is faster.

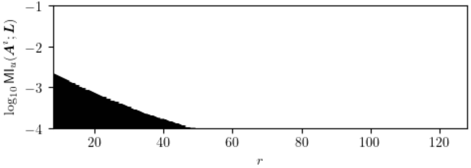


Figure: #AND=1024

Exploring design space: when to mask+shuffle ? (Part III)

Explored parameters:

- MI: per share.
- $\#AND$: Number of bits to protect.
- b_s: bitslicing size.
- r: randomness cost.

Trends for shuffling + masking:

+ large $\#AND$.
+ expensive randomness r.
- large noise.

Cycles vs. security comparison:

- For r, $\#AND$ and MI.
- ■: masking is faster.
- □: masking + shuffling is faster.

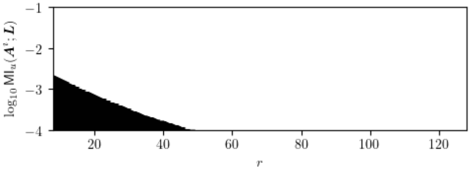


Figure: #AND=1024

Content

Content of the thesis

Three parts answering different questions:

Content of the thesis

Three parts answering different questions:

- **Part I:** *Proof-based evaluations*
  - How to improve leakage detection in open-source ?
  - How to estimate noise / information ?

Evaluation

Content of the thesis

Three parts answering different questions:

- **Part I:** *Proof-based evaluations*
  - How to improve leakage detection in open-source ?
  - How to estimate noise / information ?
- **Part II:** *Attack-based evaluations*
  - How to ease evaluation software implementations ?
  - What is the security of current implementations ?

Evaluation

Content of the thesis

Three parts answering different questions:

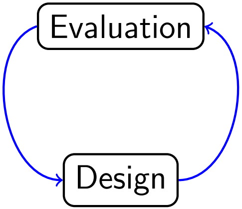
- **Part I:** *Proof-based evaluations*
  - How to improve leakage detection in open-source ?
  - How to estimate noise / information ?
- **Part II:** *Attack-based evaluations*
  - How to ease evaluation software implementations ?
  - What is the security of current implementations ?
- **Part III:** *New designs*
  - How to combine countermeasures ?
  - Is it effective / useful ? In what context ?

$\boxed{\text{Design}}$

Content of the thesis

Three parts answering different questions:

- **Part I:** *Proof-based evaluations*
  - How to improve leakage detection in open-source ?
  - How to estimate noise / information ?
- **Part II:** *Attack-based evaluations*
  - How to ease evaluation software implementations ?
  - What is the security of current implementations ?
- **Part III:** *New designs*
  - How to combine countermeasures ?
  - Is it effective / useful ? In what context ?



Both **evaluation** & **design** are linked.

Conclusion

Takeaways for the field:

Conclusion

Takeaways for the field:

- Lack of noise in low-cost MCU:
  - Hard to have effective countermeasures.
  - Masking at higher-order is needed.

Conclusion

Takeaways for the field:

- Lack of noise in low-cost MCU:
  - Hard to have effective countermeasures.
  - Masking at higher-order is needed.
- How to reach good cost vs. security ?
  - Amortizing cost of masking at the mode level.
  - Based on SPA security.

Conclusion

Takeaways for the field:

- Lack of noise in low-cost MCU:
  - Hard to have effective countermeasures.
  - Masking at higher-order is needed.
- How to reach good cost vs. security ?
  - Amortizing cost of masking at the mode level.
  - Based on SPA security.
- New open-source evaluation tools:
  - Ease multivariate quantified security analysis.
  - Better reflecting actual SCA security in countermeasure papers.

Conclusion

Takeaways for the field:

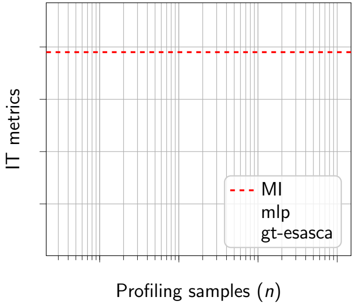
- Lack of noise in low-cost MCU:
  - Hard to have effective countermeasures.
  - Masking at higher-order is needed.
- How to reach good cost vs. security ?
  - Amortizing cost of masking at the mode level.
  - Based on SPA security.
- New open-source evaluation tools:
  - Ease multivariate quantified security analysis.
  - Better reflecting actual SCA security in countermeasure papers.

Thanks!

SCALib sources: https://github.com/simple-crypto/SCALib
SCALib documentation: https://scalib.readthedocs.io/en/latest/
CTF2020 Attacks: https://github.com/obronchain/BS21_ches2020CTF

Backup slides

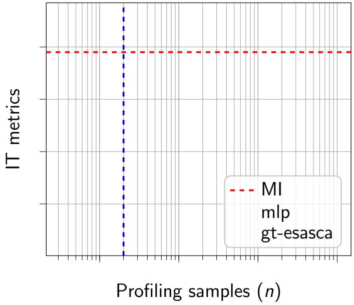
Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)



Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)

Profiling complexity:

$$\mathrm{PI}_n^{f_1} \leq \mathrm{PI}_n^{f_2}$$



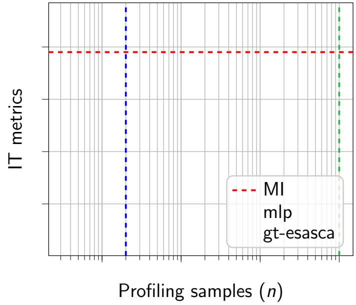
Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)

Profiling complexity:

$$\mathrm{PI}_n^{f_1} \leq \mathrm{PI}_n^{f_2}$$

Online complexity:

$$\mathrm{PI}_\infty^{f_1} \leq \mathrm{PI}_\infty^{f_2} \leq \mathrm{MI}$$



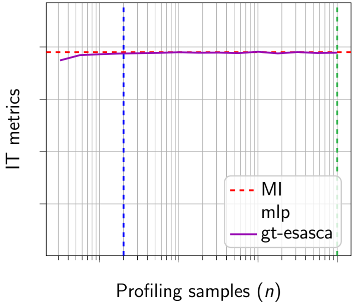
Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)

Profiling complexity:

$$\mathrm{PI}_n^{f_1} \leq \mathrm{PI}_n^{f_2}$$

Online complexity:

$$\mathrm{PI}_\infty^{f_1} \leq \mathrm{PI}_\infty^{f_2} \leq \mathrm{MI}$$



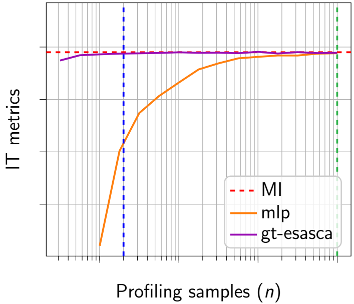
Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)

Profiling complexity:

$$\mathrm{PI}_n^{f_1} \leq \mathrm{PI}_n^{f_2}$$

Online complexity:

$$\mathrm{PI}_\infty^{f_1} \leq \mathrm{PI}_\infty^{f_2} \leq \mathrm{MI}$$



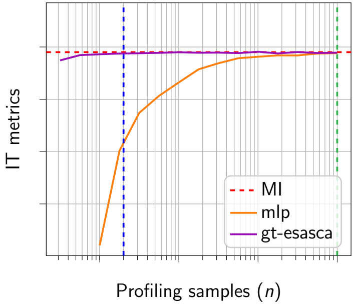
Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)

Profiling complexity:

$$\mathrm{PI}_n^{f_1} \leq \mathrm{PI}_n^{f_2}$$

Online complexity:

$$\mathrm{PI}_\infty^{f_1} \leq \mathrm{PI}_\infty^{f_2} \leq \mathrm{MI}$$



Close-to worst-case attacks:

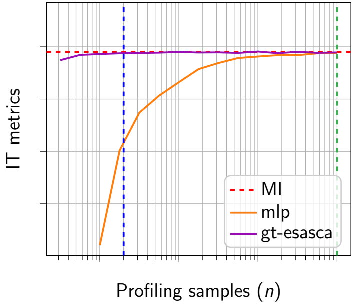
Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)

Profiling complexity:

$$\mathrm{PI}_n^{f_1} \leq \mathrm{PI}_n^{f_2}$$

Online complexity:

$$\mathrm{PI}_\infty^{f_1} \leq \mathrm{PI}_\infty^{f_2} \leq \mathrm{MI}$$



Close-to worst-case attacks:

- $\mathrm{PI}_n^{mlp} \leq \mathrm{PI}_n^{gt-esasca}$.

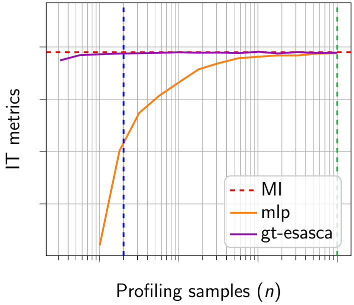
Attack-based: How to compare adv. (f_1 vs. f_2)? (Chap. 5)

Profiling complexity:

$$\mathrm{PI}_n^{f_1} \leq \mathrm{PI}_n^{f_2}$$

Online complexity:

$$\mathrm{PI}_\infty^{f_1} \leq \mathrm{PI}_\infty^{f_2} \leq \mathrm{MI}$$



Close-to worst-case attacks:

- $\mathrm{PI}_n^{mlp} \leq \mathrm{PI}_n^{gt-esasca}$.
- $\mathrm{PI}_\infty^{mlp} \stackrel{?}{=} \mathrm{PI}_\infty^{gt-esasca}$.