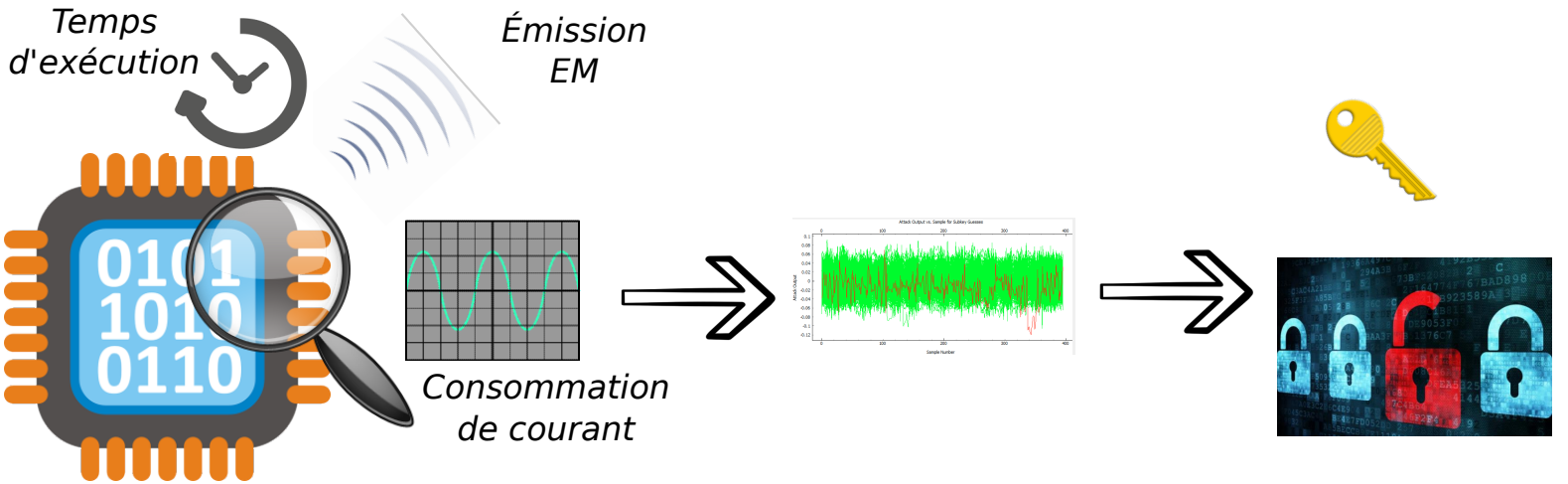
Analyse du code binaire masqué contre les SCA

Workshop Prosecco

I. Ben El Ouahma, Q. Meunier, K. Heydemann, E. Encrenaz

6 Novembre 2018, Paris, France

Attaques par canaux cachés

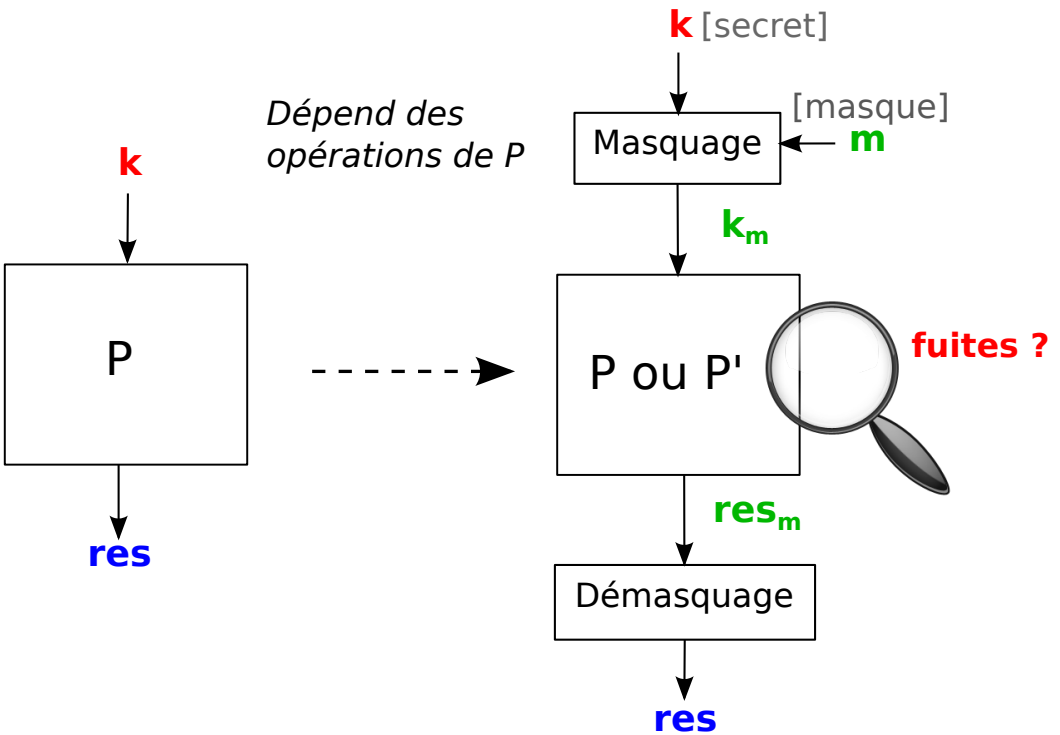


Canaux cachés

Traces

Analyse statistique
pour retrouver la clef

Masquage 1/2



À l'ordre d : l'observation de d calculs intermédiaires ne révèle pas le secret k

Masquage 2/2

Propriétés

- Très dépendant de l'algorithme
- Au niveau logiciel, ajouté dans le source code
- En pratique, programmes masqués souvent écrits directement en assembleur par des experts

Quelles garanties ?

- Simulation d'attaques
- Revue de code après compilation car celle-ci peut compromettre le masquage
- Méthodes formelles

Travaux existants

- [Bayrak,CHES13] Vérification SAT de *sensibilité* : une opération sur un secret doit dépendre d'un masque
  - ✓ Bas niveau : LLVM IR
  - × Propriété de sécurité insuffisante
- [Eldib,TACAS14] Vérification SMT de *masquage parfait* : calculs intermédiaires indépendants statistiquement des secrets
  - ✓ Bonne propriété de sécurité
  - × Programmes C booléens
  - × Explosion combinatoire de l'énumération
- [Barthe,Eurocrypt15] *t-non-interference* : distribution de probabilité conjointe de t calculs intermédiaires est indépendante des secrets
  - ✓ Bonne propriété de sécurité
  - ✓ Efficace
  - × Niveau algorithmique

But de ce travail

Objectif : Étudier l'analyse de robustesse de programmes masqués au niveau assembleur ou binaire, de façon la plus automatisée possible

Proposer une méthode d'analyse de la robustesse face aux attaques de type side-channel :

- De programmes masqués au premier ordre
- Au niveau assembleur
- Dans le *value-based model* : le résultat de l'instruction fuit
- Avec une approche symbolique qui infère le type de distribution de l'expression d'une instruction
- Contrainte : le chemin d'exécution doit être connu statiquement

Plan

Types de distribution

- *Random Uniform* (`RUD`)
- *Unknown* (`UKD`)
- *Constant* (`CST`)
- *(Statistically) Independent from Secrets* (`ISD`): même distribution pour toutes les valeurs du secret.

`e = (k` $\oplus$ `m1) & m2`

k	m_1	m_2	e	
0	0	0	0	$P(e=0\|k=0)=\frac{3}{4}$
	0	1	0	
	1	0	0	$P(e=1\|k=0)=\frac{1}{4}$
	1	1	1	
1	0	0	0	$P(e=0\|k=1)=\frac{3}{4}$
	0	1	1	
	1	0	0	$P(e=1\|k=1)=\frac{1}{4}$
	1	1	0	

`e'= (k` $\oplus$ `m1) & m1`

e'	
0	$P(e'=0\|k=0)=\frac{1}{2}$
0	
1	$P(e'=1\|k=0)=\frac{1}{2}$
1	
0	$P(e'=0\|k=1)=1$
0	
0	$P(e'=1\|k=1)=0$
0	

Dépendances

$(k \oplus m)\ \&\ m = (RUD)\ \&\ RUD = ISD \Longrightarrow$ faux

- Nécessaire de mémoriser les dépendances *structurelles*

$(k \oplus m)\ \&\ m = (RUD\{m,k\})\ \&\ RUD\{m\} = UKD$

Types garantis sans fuite

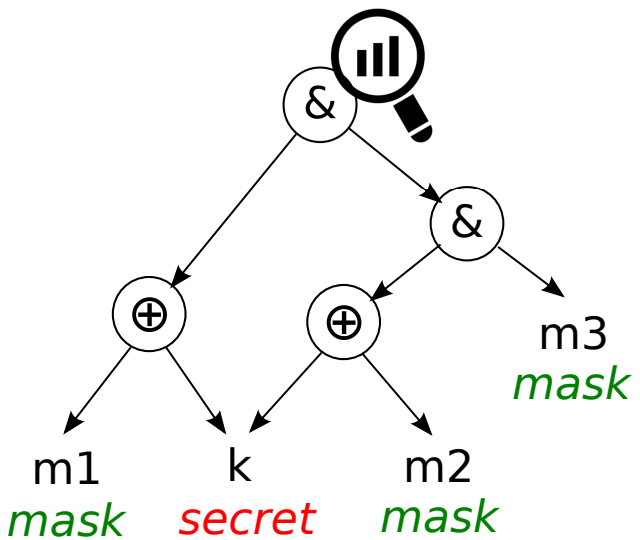
- e∼RUD
- e∼ISD
- e∼UKD sans dépendance structurelle sur un secret

Type potentiellement vulnérable

- e∼UKD avec dépendance structurelle sur un secret

Schéma de vérification

```
    # r0 ← k; r1 ← m1; r2 ← m2; r3 ← m3
1   eor r4, r0, r1   # k ⊕ m1
2   eor r5, r0, r2   # k ⊕ m2
3   and r5, r5, r3   # (k ⊕ m2) & m3
4   and r5, r5, r4   # (k ⊕ m1) & ((k ⊕ m2) & m3)
```



Arbre d'expression de la dernière instruction

La distribution de la racine est elle statistiquement indépendante de k ?

- Étiqueter les variables d'entrée selon leur type
- Combinaison ascendante de types de distribution utilisant des règles d'inférence

Masques dominants

- expression **e** = **e'** $\oplus$ **m** / **e** = **e'** + **m** mod 2^n
- **m**$\sim$**RUD**
- **m** n'apparaît pas dans **e'**

$\Longrightarrow$ **e**$\sim$**RUD** et **m** est un masque dominant de **e**.

Pour chaque expression, 2 ensembles : $\texttt{dom}_{\oplus}\texttt{(e)}$, $\texttt{dom}_{+}\texttt{(e)}$
Exemples:

- $\texttt{dom}_{\oplus}\texttt{((k + m1) } \oplus \texttt{ (k } \oplus \texttt{ m1 } \oplus \texttt{ m2)) = m2}$
- $\texttt{dom}_{+}\texttt{((k + m1) } \oplus \texttt{ 0) = dom}_{+}\texttt{(k + m1) = m1}$

Autres règles d'inférence

- Pour $\oplus$, $+$ mod 2^n
- Pour AND, OR
- Distribution des accès SBox = distribution de l'expression d'accès (propriété de permutation)

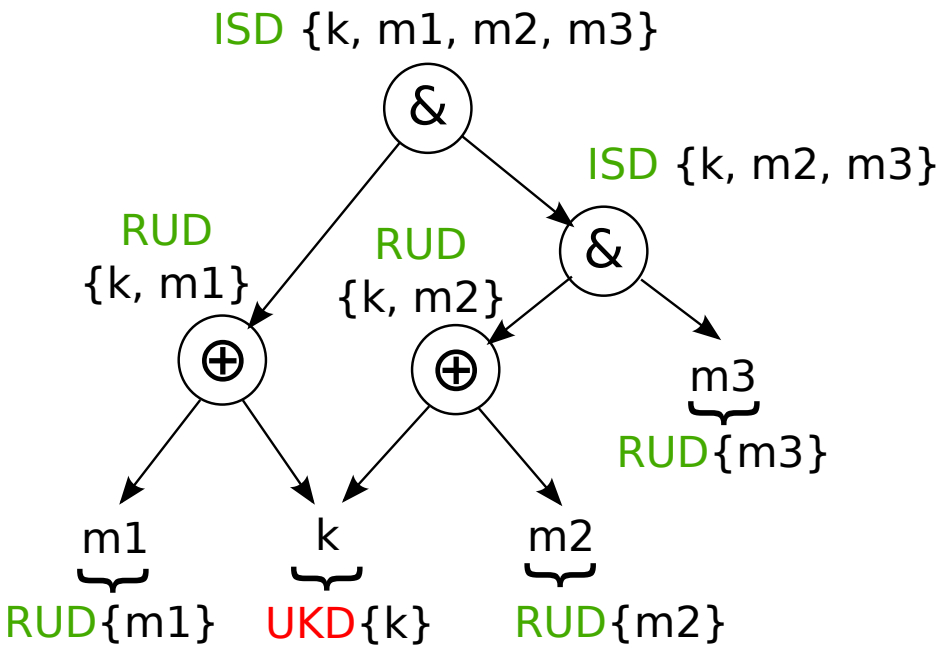
Règle Disjoint

- Deux expressions `u`$\sim$`ISD` et `v`$\sim$`ISD`
- Aucun masque en commun entre `u` et `v`

$\Longrightarrow$ `(u op v)`$\sim$`ISD` pour tout opérateur binaire `op`

Exemple d'analyse de distribution

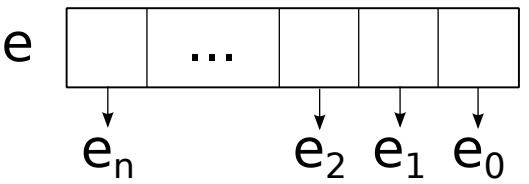
Dernière instruction *i4* : $(\mathtt{k} \oplus \mathtt{m}_1)\ \&\ ((\mathtt{k} \oplus \mathtt{m}_2)\ \&\ \mathtt{m}_3)$



▷ *i4* est statistiquement indépendante de $\mathtt{k}$

Analyse au niveau bit

Quand l'analyse précédente ne permet pas de conclure :
$\Longrightarrow$ Décomposition de l'expression par bit



▷ cas 1 :

$e_n \quad \cdots \quad e_2 \; e_1 \; e_0$
$\downarrow \qquad\qquad \downarrow \; \downarrow \; \downarrow$
$m_n \qquad\quad m_2 \; m_1 \; m_0$

$e_i \sim$RUD et masques dominants différents pour chaque e_i

▷ cas 2 :

$\underbrace{e_n \cdots e_{i+1}}_{\text{CST}} \; \underbrace{e_i}_{\text{ISD}} \; \underbrace{e_{i-1} \cdots e_0}_{\text{CST}}$

Concaténation d'un bit ISD et de bits CST

▷ cas 3 :

$\underbrace{e_n \cdots}_{\text{CST}} \; \underbrace{e_i}_{\text{ISD}} \; \underbrace{\cdots}_{\text{CST}} \; \underbrace{e_i}_{\text{ISD}} \; \underbrace{\cdots e_0}_{\text{CST}}$

Expression identique de deux bits ISD et concaténation avec des bits CST

Exemple dans la fonction mix columns de l'AES (première ronde) :

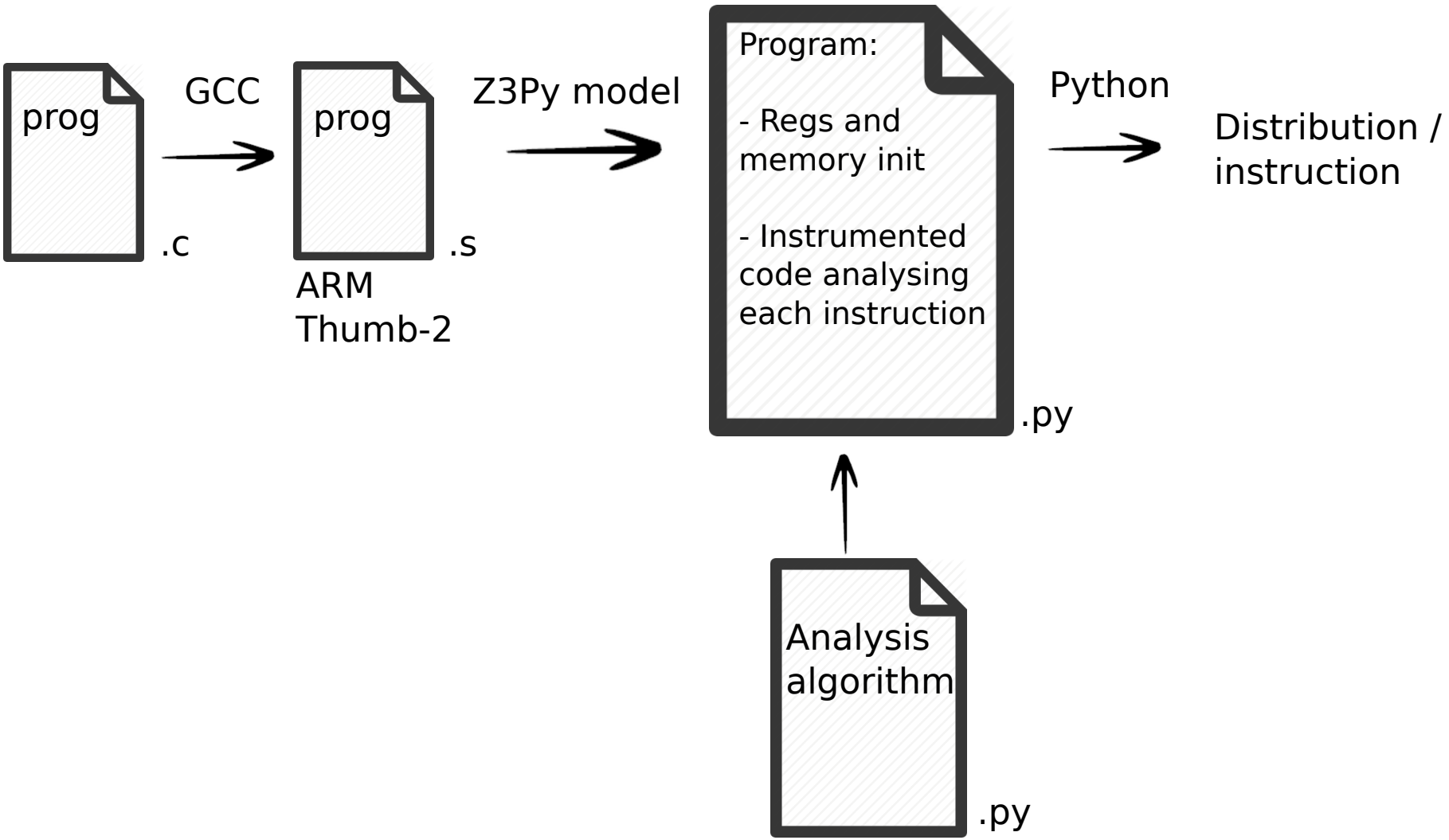
```
e = (LSR(mt1 ⊕ mp ⊕ sbox5, 7) ⊕ LSR(mt2 ⊕ mp ⊕ sbox10, 7)) +
   ((LSR(mt1 ⊕ mp ⊕ sbox5, 7) ⊕ LSR(mt2 ⊕ mp ⊕ sbox10, 7)) << 1)
```

$b_7 = mt1_7 \oplus \cancel{mp_7} \oplus sbox5_7 \oplus mt2_7 \oplus \cancel{mp_7} \oplus sbox10_7$

$e \Longrightarrow 0000\ 00b_7b_7 \Longrightarrow$ ISD

Plan

Implémentation



Benchmarks

Programme	#Inst ASM	Taille en bits des données	#Masques	#Secrets	Robuste dans réf
Programmes C booléens					
P6 [Eldib,TACAS14]	8	1	3	3	×
Masked Chi [Eldib,TACAS14]	8	1	2	3	✓
Algorithmes de conversion de masquage booléen/arithmétique					
Goubin Conversion [Goubin01]	8	4	2	1	✓
Coron Conversion [Coron15]	37	4	3	1	✓
Algorithmes cryptographiques					
Masked AES ronde 1 [Herbst06]	422	8	6	16 + 16	✓
Simon TI ronde 1 [Shahverdi17]	15	32	5	3 + 2	✓

Résultats d'analyse

Vérification avec *Enum-C* : programme C qui énumère les combinaisons de valeurs (masques et secrets) et cherche 2 valeurs d'un secret qui ont des distributions différentes

Programme	# RUD	# ISD	# CST	# UKD	Temps	Enum-C # fuites
P6	6	2	0	0	<1s	0
Chi masqué	2	2	0	4	<1s	4
Conversion Goubin	5	0	0	3	<1s	0
Conversion Coron	14	10	0	13	2s	7
Masked AES ronde 1	302	0	120	0	22s	-
Simon TI ronde 1	7	4	1	3	8.5s	-

Analyse par bit

Programme	#RUD$_m$	#RUD$_b$	#RUD total	#ISD$_m$	#ISD$_b$	#ISD total	#CST	#UKD$_m$	#UKD$_b$	#UKD total
P6	6	0	6	2	0	2	0	0	0	0
Masked Chi	2	0	2	2	0	2	0	4	4	4
Conv. Goubin	0	0	5	0	0	0	0	3	3	3
Conv. Coron	10	4	14	6	4	10	0	21	13	13
AES masqué ronde 1	222	80	302	0	0	0	120	80	0	0
Simon TI ronde 1	7	0	7	0	3	3	1	7	4	4

- Algorithme de Coron & Simon TI : conclusion pour environ 40% d'instructions UKD
- AES masqué : plus aucune instruction UKD

Plan

Bilan

- Proposition d'une méthode d'analyse de distribution pour des programmes en assembleur masqués à l'ordre 1
- Confirmation expérimentale du besoin de vérifier la robustesse au niveau assembleur

Perspectives 1/2

À court terme :

- **Automatisation** en cours
  - pour traiter du code binaire en entrée
  - de la modélisation Z3Py
  - pour repérer automatiquement les accès SBox
- **Comparaison** avec la méthode de Barthe *et. al*
  - application au niveau assembleur
  - temps et précision de l'analyse

Perspectives 2/2

À moyen terme :

- **Programmes analysés :** autres algorithmes avec des t-tables
- **Raffinement** des règles d'inférence et de l'analyse par bit
- **Extension** à d'autres modèles de fuite comme le *transition-based model*, et au masquage à l'ordre supérieur

Références

[Bayrak,CHES13] Ali Galip Bayrak, Francesco Regazzoni, David Novo, Paolo Ienne. Sleuth: Automated Verification of Software Power Analysis Countermeasures. *CHES 2013: 293-310*

[Eldib,TACAS14] Hassan Eldib, Chao Wang, Patrick Schaumont. SMT-Based Verification of Software Countermeasures against Side-Channel Attacks. *TACAS 2014: 62-77*

[Barthe,Eurocrypt15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub. Verified Proofs of Higher-Order Masking. *EUROCRYPT (1) 2015: 457-485*

[Goubin01] Louis Goubin. A sound method for switching between boolean and arithmetic masking. In Cryptographic Hardware and Embedded SystemsCHES 2001, pages 3–15. Springer, 2001.

[Coron15] Jean-Sébastien Coron, Johann GroBschadl, Mehdi Tibouchi, and Praveen Kumar Vadnala. Conversion from arithmetic to boolean masking with logarithmic complexity. In International Workshop on Fast Software Encryption, pages 130–149. Springer, 2015.

[Herbst06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An aes smart card implementation resistant to power analysis attacks. In ACNS, volume 3989, pages 239–252. Springer, 2006.

[Shahverdi17] Aria Shahverdi, Mostafa Taha, and Thomas Eisenbarth. Lightweight side channel resistance. Threshold implementations of simon. IEEE Transactions on Computers, 66(4):661–671, 2017.

Merci pour votre attention.