TELECOM Paris

IP PARIS

Méthodes pour la modélisation des injections de fautes électromagnétiques

JAIF 2021
23/09/2021

Oualid Trabelsi, Laurent Sauvage, Jean-Luc Danger
(Travaux financés par le projet CSAFE+)

Institut Mines-Télécom

Plan

Plan

Contexte

Aujourd'hui ..

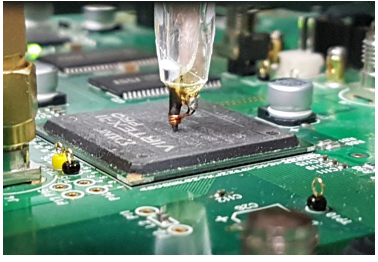

Besoin de fonction de sécurité :

- authentification
- chiffrement
- etc.

$\Rightarrow$ Attention aux attaques sur les implémentations

Contexte

Perturbation par rayonnement électromagnétique



- Injection de fautes électromagnétiques (EMFI) :
  - non invasive
  - localité
  - coût

Positionnement

Questions scientifiques et contributions

1 Modèles de fautes générés par EMFI
- quels sont les modèles de fautes au niveau bit/instruction ?
- comment identifier les éléments vulnérables de l'architecture d'un MCU ?
- quelle est la durée temporelle des fautes (transitoire, persistante . . .) ?

$\Rightarrow$ 3 méthodes génériques pour l'analyse des différentes vulnérabilités

Positionnement

Questions scientifiques et contributions

1. Modèles de fautes générés par EMFI
   - quels sont les modèles de fautes au niveau bit/instruction ?
   - comment identifier les éléments vulnérables de l'architecture d'un MCU ?
   - quelle est la durée temporelle des fautes (transitoire, persistante . . .) ?

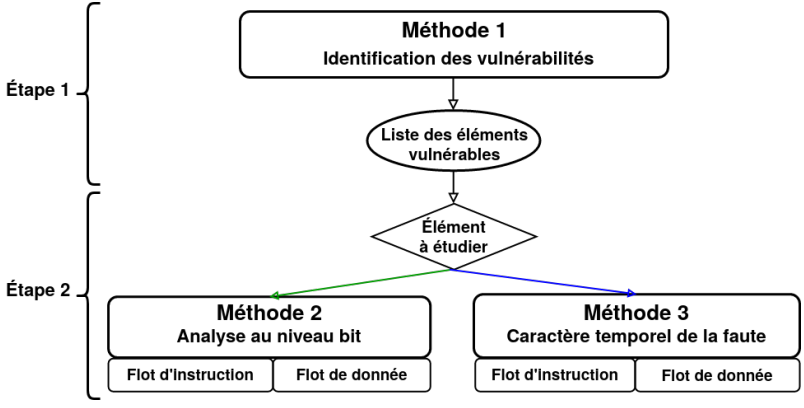
   $\Rightarrow$ 3 méthodes génériques pour l'analyse des différentes vulnérabilités

2. Efficacité de contre-mesures face à ces modèles de fautes

   $\Rightarrow$ analyse de la vulnérabilité et proposition d'améliorations

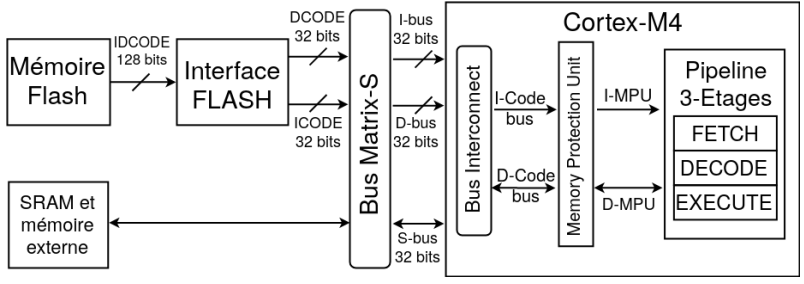
Plan

Démarche d'analyse sur MCU



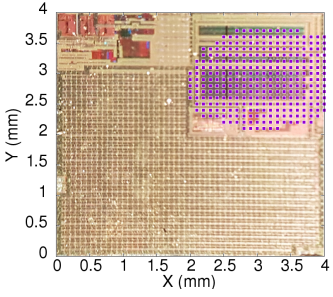
Cible MCU d'étude

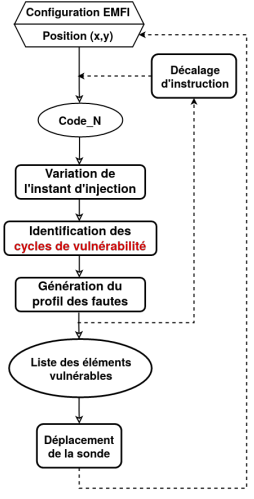
Architecture du STM32F407



- Chaque élément peut être mis en faute
- Observation des registres processeur uniquement

Localisation spatiale des éléments vulnérables

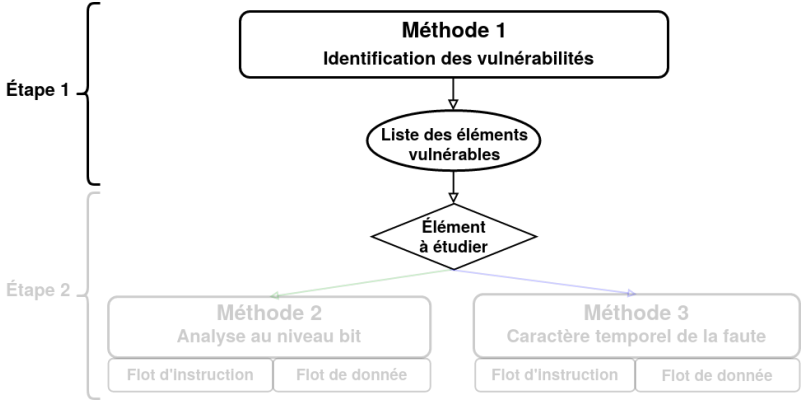




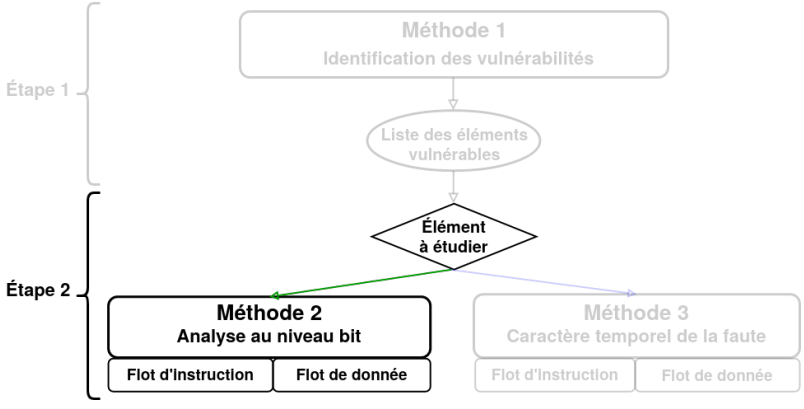
Sur le flot d'instruction et de donnée :

- même élément vulnérable (interface mémoire)
- mêmes positions d'impact

Démarche d'analyse sur MCU



Démarche d'analyse sur MCU



Quel est le modèle de faute au niveau bit ?

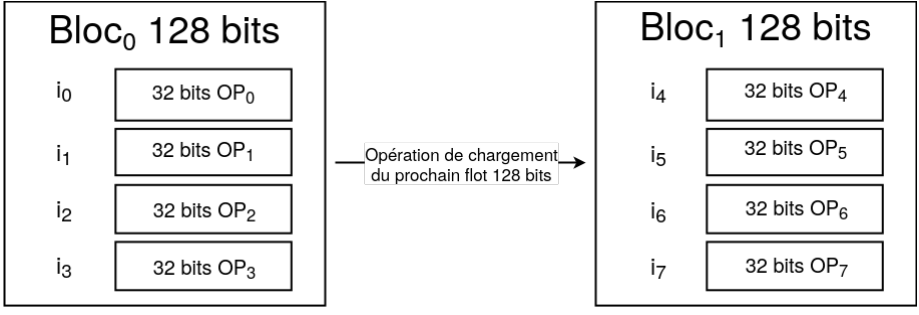
Conditions d'observation des modèles de fautes

bit_{N-1}	bit_N	$bit_{N'}$	**bit-set**	**bit-reset**	**bit-flip**	**no-sampling**
0	0	1	✓	–	✓	–
0	1	0	–	✓	✓	✓
1	0	1	✓	–	✓	✓
1	1	0	–	✓	✓	–

Valider un modèle de faute $\rightarrow$ Confirmer toutes les observations

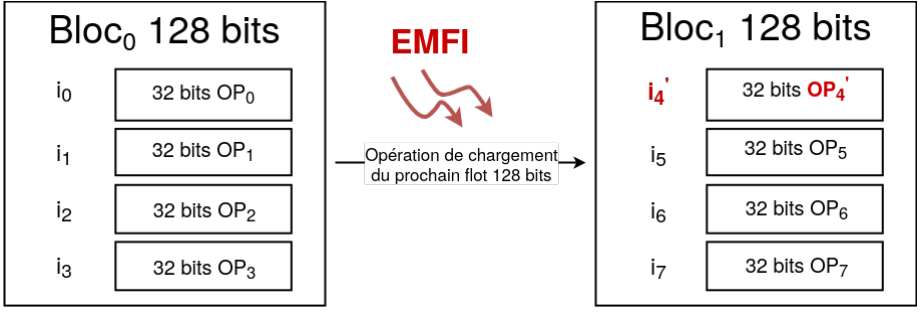
Quel est le modèle de faute au niveau bit ?

Adaptation de l'opcode pour l'observation des modèles de fautes



Quel est le modèle de faute au niveau bit ?

Adaptation de l'opcode pour l'observation des modèles de fautes



instruction fautée $\rightarrow$ valeur binaire de l'opcode altérée

Quel est le modèle de faute au niveau bit ?

Identification des modèles de fautes - flot d'instruction

```
/* séquence bit-reset */
op    | instruction
/* bloc0 8 x 16 bits */
0 0000    MOVS R0,R0
1 0000    MOVS R0,R0
2 0000    MOVS R0,R0
3 0000    MOVS R0,R0
4 0000    MOVS R0,R0
5 0000    MOVS R0,R0
6 0000    MOVS R0,R0
7 0000    MOVS R0,R0
/* bloc1 8 x 16 bits */
8 3FFF    SUBS R7,#0xFF
9 3FFF    SUBS R7,#0xFF
10 3FFF   SUBS R7,#0xFF
11 3FFF   SUBS R7,#0xFF
12 3FFF   SUBS R7,#0xFF
13 3FFF   SUBS R7,#0xFF
14 3FFF   SUBS R7,#0xFF
15 3FFF   SUBS R7,#0xFF
```

⇓

fautes observées

```
/* séquence bit-reset */
op    | instruction
/* bloc0 8 x 16 bits */
0 3FFF    SUBS R7,#0xFF
1 3FFF    SUBS R7,#0xFF
2 3FFF    SUBS R7,#0xFF
3 3FFF    SUBS R7,#0xFF
4 3FFF    SUBS R7,#0xFF
5 3FFF    SUBS R7,#0xFF
6 3FFF    SUBS R7,#0xFF
7 3FFF    SUBS R7,#0xFF
/* bloc1 8 x 16 bits */
8 3FFF    SUBS R7,#0xFF
9 3FFF    SUBS R7,#0xFF
10 3FFF   SUBS R7,#0xFF
11 3FFF   SUBS R7,#0xFF
12 3FFF   SUBS R7,#0xFF
13 3FFF   SUBS R7,#0xFF
14 3FFF   SUBS R7,#0xFF
15 3FFF   SUBS R7,#0xFF
```

⇓

fautes observées

⇒ Modèle de faute bit-reset sur le flot d'instruction

Quel est le modèle de faute au niveau bit ?

Identification des modèles de fautes - flot de donnée

```
/* séquence bit-set */
/* Init. données */
.equ d1, 0x00000000
.equ d2, 0x00000000
.equ d3, 0x00000000
.equ d4, 0x00000000
.equ d6, 0x00000000
.equ d7, 0x00000000
.equ d8, 0x00000000
.equ d9, 0x00000000
/*   Seq Trig Haut  */
0 LDR.w   R0,=d1
1 LDR.w   R5,=d6
2 LDM.w   R0!,{R1-R4}
3 LDM.w   R5!,{R6-R9}
```

⇓

fautes observées

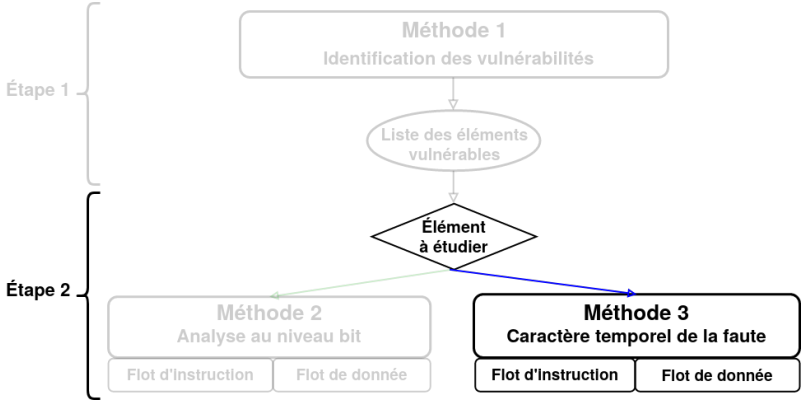
```
/* séquence bit-set */
/* Init. données */
.equ d1, 0xFFFFFFFF
.equ d2, 0xFFFFFFFF
.equ d3, 0xFFFFFFFF
.equ d4, 0xFFFFFFFF
.equ d6, 0x00000000
.equ d7, 0x00000000
.equ d8, 0x00000000
.equ d9, 0x00000000
/*   Seq Trig Haut  */
0 LDR.w   R0,=d1
1 LDR.w   R5,=d6
2 LDM.w   R0!,{R1-R4}
3 LDM.w   R5!,{R6-R9}
```

⇓

fautes observées

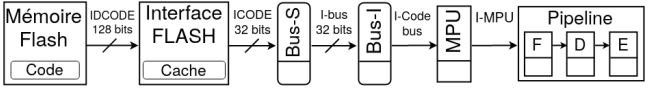
$\Rightarrow$ Modèle de faute bit-set sur le flot de donnée

Démarche d'analyse sur MCU

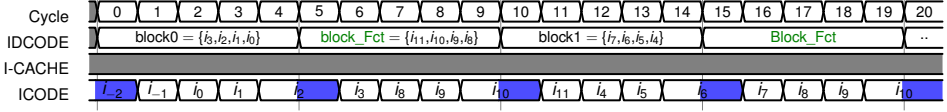


Quel est l'impact temporel des fautes ?

Méthode 3 - Principe d'analyse

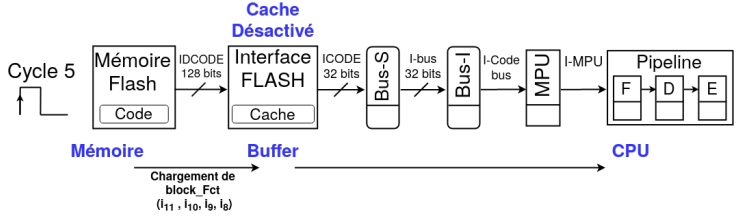


I-Cache désactivé :

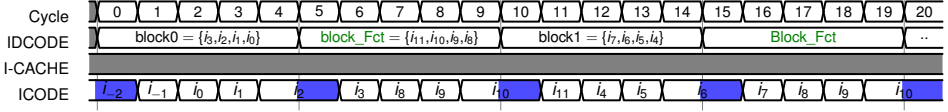


Quel est l'impact temporel des fautes ?

Méthode 3 - Principe d'analyse

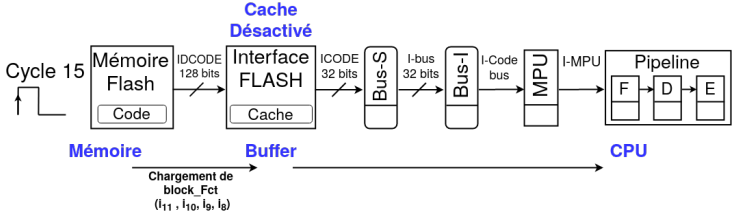


I-Cache désactivé :

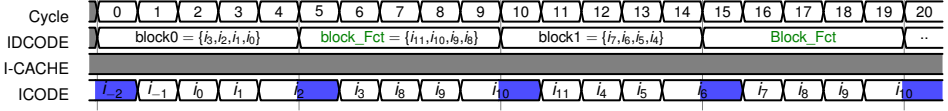


Quel est l'impact temporel des fautes ?

Méthode 3 - Principe d'analyse

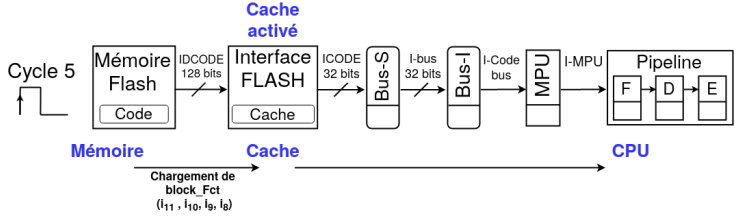


I-Cache désactivé :

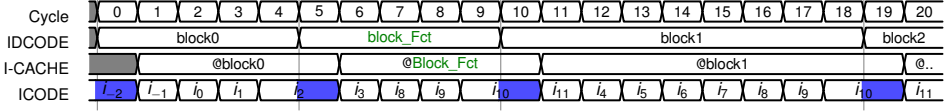


Quel est l'impact temporel des fautes ?

Méthode 3 - Principe d'analyse

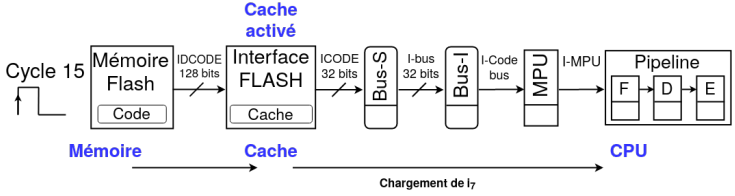


I-Cache activé :

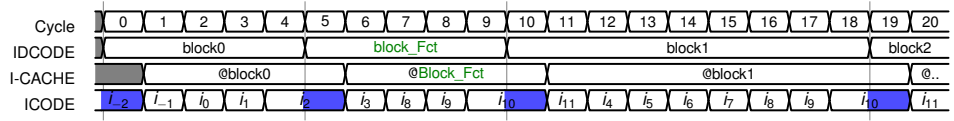


Quel est l'impact temporel des fautes ?

Méthode 3 - Principe d'analyse

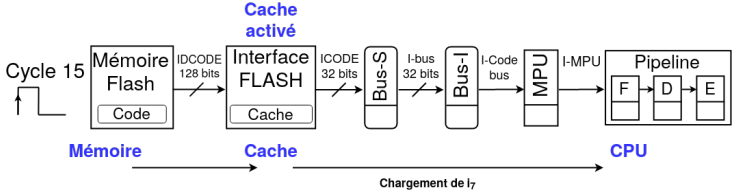


I-Cache activé :

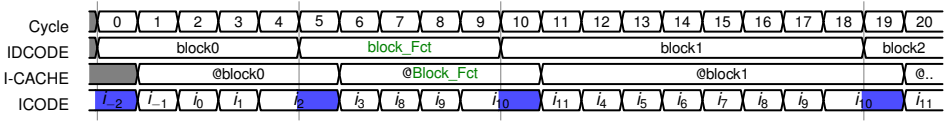


Quel est l'impact temporel des fautes ?

Méthode 3 - Principe d'analyse



I-Cache activé :



- Impact sur la ligne de I-Cache :
  - faute de lecture ?
  - faute d'écriture (faute semi-persistante) ?

Quel est l'impact temporel des fautes ?

Résultat Méthode 3 - Flot d'instruction

```
/*  Trig Haut   */
..
/* Cfg. I-Cache */
..
/*  Inst. NOP   */
..
```

```
/* Bloc 0 */
0 BL.w    Fct
1 MOV.w   R4,R1
2 MOV.w   R5,R2
3 MOV.w   R6,R3
```

```
/* Bloc 1 */
4 BL.w    Fct
5 MOV.w   R7,R1
6 MOV.w   R8,R2
7 MOV.w   R9,R3
```

```
/*   Fct   */
8  ADD.w   R1,R1,#0x1
9  ADD.w   R2,R2,#0x2
10 ADD.w   R3,R3,#0x4
11 NOP
12 BX      LR
```

```
/* Inst. NOP */
..
/* Trig Bas  */
..
```

Registre	Valeur Initiale	1er Appel			2nd Appel		
		Attendu	Résultat	Modèle	Attendu	Résultat	Modèle
R1	0x00011000	0x00011001	0x40811000	Rempl.	0x00011002	0x40811000	Rempl.
R2	0x00033000	0x00033002	0x00033002	–	0x00033004	0x00033004	–
R3	0x00077000	0x00077004	0x00077000	Saut	0x00077008	0x00077000	Saut

- 1ère et 2nd exécutions fautées → Corruptions durant l'écriture sur le cache

  ⇒ faute semi-persistante

Quel est l'impact temporel des fautes ?

Résultat Méthode 3 - Flot d'instruction

```
/*  Trig Haut   */
..
/* Cfg. I-Cache */
..
/*  Inst. NOP   */
..
```

```
/* Bloc 0 */
0 BL.w    Fct
1 MOV.w   R4,R1
2 MOV.w   R5,R2
3 MOV.w   R6,R3
```

```
/* Bloc 1 */
4 BL.w    Fct
5 MOV.w   R7,R1
6 MOV.w   R8,R2
7 MOV.w   R9,R3
```

```
/*   Fct   */
8  ADD.w   R1,R1,#0x1
9  ADD.w   R2,R2,#0x2
10 ADD.w   R3,R3,#0x4
11 NOP
12 BX      LR
```

```
/* Inst. NOP */
..
/* Trig Bas  */
..
```

Registre	Valeur Initiale	1er Appel			2nd Appel		
		Attendu	Résultat	Modèle	Attendu	Résultat	Modèle
R1	0x00011000	0x00011001	0x40811000	Rempl.	0x00011002	0x40811000	Rempl.
R2	0x00033000	0x00033002	0x00033002	–	0x00033004	0x00033004	–
R3	0x00077000	0x00077004	0x00077000	Saut	0x00077008	0x00077000	Saut

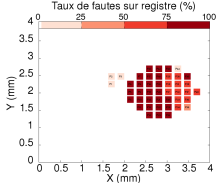
- 1ère et 2^{nd} exécutions fautées → Corruptions durant l'écriture sur le cache

⇒ faute semi-persistante (idem pour le cache de donnée)

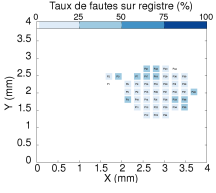
Distribution des fautes dans un bloc 128 bits

Impact du paramètre spatial sur le nombre d'instructions en faute

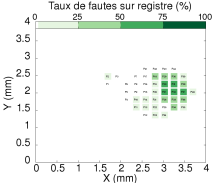
Une instruction



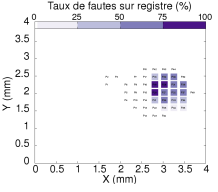
Deux instructions

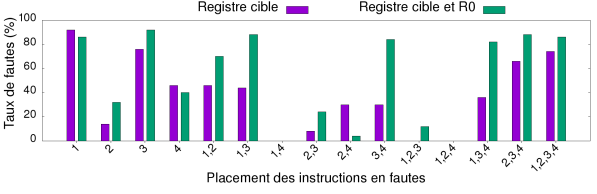


Trois instructions



Quatre instructions





Généralisation de la démarche d'analyse

Application des méthodes sur un autre MCU

Caractérisation		STM32F407	SAM4C16
Élements vulnérables		interface mémoire	interface mémoire
Modèles niveau bit	instruction	*bit-reset*	*bit-reset*
	donnée	*bit-set*	*bit-reset*
Caractère temporel	cache désactivé	transitoire	transitoire
	cache activé	semi-persistent	semi-persistent

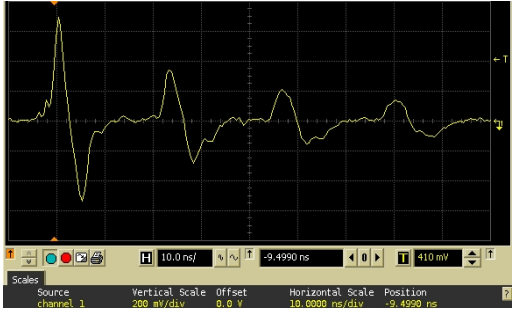
Résultats de caractérisation entre deux plateformes EMFI

Plateforme Avtech vs Keysight - cible STM32F4

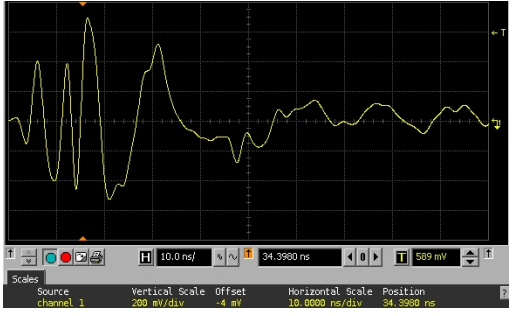
Caractérisation		Plateforme EMFI P_{Av}	Plateforme EMFI P_{Ke}
Élements vulnérables		interface mémoire	interface mémoire
Modèles niveau bit	instruction	*bit-reset*	*bit-set*
	donnée	*bit-set*	*bit-set*
Caractère temporel	cache désactivé	transitoire	transitoire
	cache activé	semi-persistent	semi-persistent

Résultats de caractérisation entre deux plateformes EMFI

Forme d'onde générée par P_{Av}



Forme d'onde générée par P_{Ke}



Plan

Contre-mesures logicielles

État de l'art

- Majorité est basée sur :
  - redondance algorithmique
  - redondance de tour (AES)
- Résistent à un seul tir

- Contre-mesures à redondance d'instruction (Barenghi *et al.*, Moro *et al.*)
  - conçues pour résister à un seul saut d'instruction
  - efficacité contre plusieurs sauts proches (même bloc) ?

Contre-mesures logicielles par redondance d'instruction

Barenghi *et al.* [1]

1. Duplication d'instruction + détection de faute
2. Triplication d'instruction + détection et correction de faute
3. Calcul de parité (donnée) + détection

Contre-mesures logicielles par redondance d'instruction

Duplication d'instruction - Analyse de résistance

```
/* inst. cible */
LDR.w R4,[R7]
/* Duplication */
---- bloc0 ----
0 LDR.w R4,[R7]
1 LDR.w R12,[R7]
2 CMP.w R12,R4
3 BNE.w <erreur>
```

Contre-mesures logicielles par redondance d'instruction

Duplication d'instruction - Analyse de résistance

```
/* inst. cible */
LDR.w R4,[R7]
/* Duplication */
---- bloc0 ----
0 LDR.w R4,[R7]
1 LDR.w R12,[R7]
2 CMP.w R12,R4
3 BNE.w <erreur>
```

```
/* Contre-mesure Améliorée */
---- bloc0 ----
0 inst. libre
1 inst. libre
2 inst. libre
3 LDR.w R4,[R7]
---- bloc1 ----
4 LDR.w R12,[R7]
5 CMP.w R12,R4
6 BNE.w <erreur>
7 inst. libre
```

Méthode d'amélioration :

- Répartition de la duplication sur plusieurs blocs
- Coût nul

Contre-mesures logicielles par redondance d'instruction

Duplication d'instruction - Analyse de résistance

```
/* inst. cible */
LDR.w R4,[R7]
/* Duplication */
---- bloc0 ----
LDR.w R4,[R7]
LDR.w R12,[R7]
CMP.w R12,R4
BNE.w <erreur>
```

```
/* Contre-mesure Améliorée */
---- bloc0 ----
inst. libre
inst. libre
inst. libre
LDR.w R4,[R7]
---- bloc1 ----
LDR.w R12,[R7]
CMP.w R12,R4
BNE.w <erreur>
inst. libre
```

				Injection de fautes Chargement bloc0
Duplication	**Registre**	**Valeur Initiale**	**Attendu**	**Résultat**
sans amélioration	R4	0x0FF00000	0xF010010F	0x0FF00000
	R12	0x03300000	0xF010010F	0x03300000
	R9 (erreur)	0x00000000	0x00000000	0x00000000
			Faute	(non détecté)

Méthode d'amélioration :

- Répartition de la duplication sur plusieurs blocs
- Coût nul

Contre-mesures logicielles par redondance d'instruction

Duplication d'instruction - Analyse de résistance

```
/* inst. cible */
LDR.w R4,[R7]
/* Duplication */
---- bloc0 ----
0 LDR.w R4,[R7]
1 LDR.w R12,[R7]
2 CMP.w R12,R4
3 BNE.w <erreur>
```

```
/* Contre-mesure Améliorée */
---- bloc0 ----
0 inst. libre
1 inst. libre
2 inst. libre
3 LDR.w R4,[R7]
---- bloc1 ----
4 LDR.w R12,[R7]
5 CMP.w R12,R4
6 BNE.w <erreur>
7 inst. libre
```

			Injection de fautes Chargement bloc0	
Duplication	**Registre**	**Valeur Initiale**	**Attendu**	**Résultat**
sans amélioration	R4	0x0FF00000	0xF010010F	0x0FF00000
	R12	0x03300000	0xF010010F	0x03300000
	R9 (erreur)	0x00000000	0x00000000	0x00000000
			Faute	(non détecté)
avec amélioration	R4	0x0FF00000	0xF010010F	0x0FF00000
	R12	0x03300000	0xF010010F	0xF010010F
	R9 (erreur)	0x00000000	0x00000000	0x5A5A5A5A
			Faute	détectée

Méthode d'amélioration :

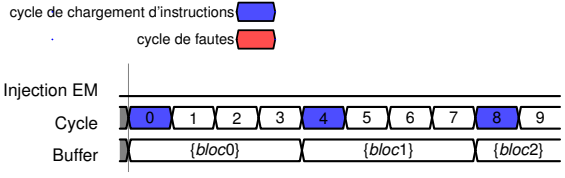
- Répartition de la duplication sur plusieurs blocs
- Coût nul

Contre-mesures logicielles

- Répartition de la duplication sur plusieurs blocs
  $\Rightarrow$ Inciter un attaquant à produire plus d'une injection
- Saut de 300 instructions successives en laser [2]
- Peut-on générer des sauts successifs dans le temps en EM ?

Tirs multiples dans le temps

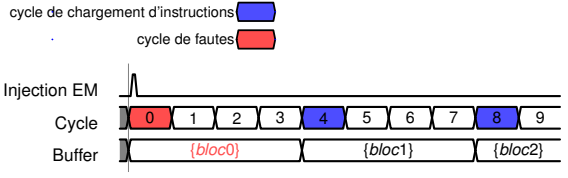
Méthode




```
/* 320 instructions */
#### bloc 0 ####
0   ADD R1,#1
1   ADD R1,#1
2   ADD R1,#1
3   ADD R1,#1
#### bloc 1 ####
1   ADD R1,#1
2   ADD R1,#1
3   ADD R1,#1
4   ADD R1,#1
...
...
#### bloc 80 ####
316 ADD R1,#1
317 ADD R1,#1
318 ADD R1,#1
319 ADD R1,#1
#### Résultat R1 = 320 ####
```

Tirs multiples dans le temps

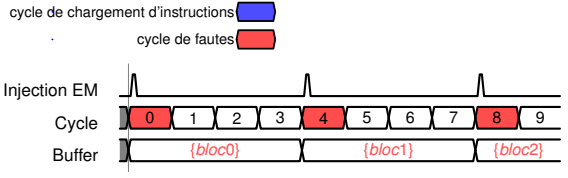
Méthode




```
/* 320 instructions */
#### bloc 0 ####
0   NOP
1   NOP
2   NOP
3   NOP
#### bloc 1 ####
1   ADD  R1,#1
2   ADD  R1,#1
3   ADD  R1,#1
4   ADD  R1,#1
...
...
#### bloc 80 ####
316 ADD  R1,#1
317 ADD  R1,#1
318 ADD  R1,#1
319 ADD  R1,#1
#### Résultat R1 = 316 ####
```

Tirs multiples dans le temps

Méthode



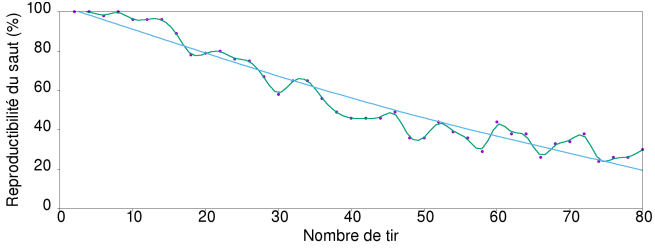

```
/* 320 instructions */
#### bloc 0 ####
0 NOP
1 NOP
2 NOP
3 NOP
#### bloc 1 ####
4 NOP
5 NOP
6 NOP
7 NOP
...
...
#### bloc 80 ####
316 NOP
317 NOP
318 NOP
319 NOP
#### Résultat R1 = 0 ####
```

- Impulsions successives à fréquence fixe $F_{pulse} = \frac{F_{sys}}{4}$
  $\rightarrow$ Cibler les cycles de chargement d'instructions

Tirs multiples dans le temps

Résultat tir multiple - cible SAMD21

- $F_{pulse} \simeq \frac{8MHz}{4} = 1.9936MHz$



Nb de tir	2	10	20	30	40	50	60	70	80
Nb de saut	8	40	80	120	160	200	240	280	320
Taux (%)	100	96	79	58	46	36	44	34	30

Plan

Conclusions

1- Trois Méthodes pour l'analyse des vulnérabilités sur MCU :
   - → Vulnérabilité de l'interface mémoire.
   - → Modèle de faute au niveau bit dépend de l'architecture/plateforme EM.
   - → Fautes semi-persistantes sur le cache d'instruction/donnée.
   - → Fautes de 4 instructions 32 bits successives (voir plus en 16 bits).

Conclusions

1- Trois Méthodes pour l'analyse des vulnérabilités sur MCU :
   - → Vulnérabilité de l'interface mémoire.
   - → Modèle de faute au niveau bit dépend de l'architecture/plateforme EM.
   - → Fautes semi-persistantes sur le cache d'instruction/donnée.
   - → Fautes de 4 instructions 32 bits successives (voir plus en 16 bits).
2- Analyse de vulnérabilité des contre-mesures à redondance d'instruction.
   - → Peuvent être améliorées pour une meilleur robustesse
   - → Vulnérables contre les sauts multiples dans le temps

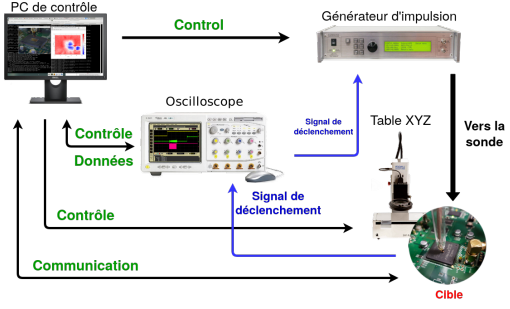
Perspectives

- Étendre la démarche d'analyse sur MCU à d'autres architectures :
  - Plusieurs niveaux de cache
  - Multi-coeurs
  - blocs cryptographiques
- Amélioration des contre-mesures vis-à-vis :
  - sauts multiples dans le temps
  - fautes semi-persistantes sur le cache

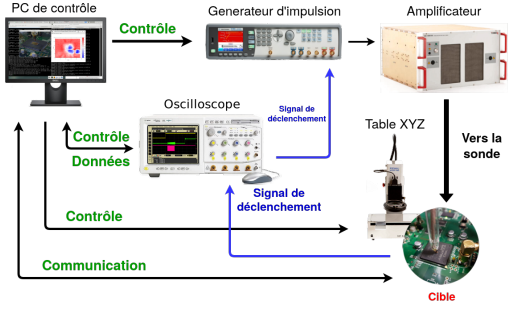
Q&A Time !

Plateforme EM Avtech vs Keysight

Plateforme Avtech P_{Av}

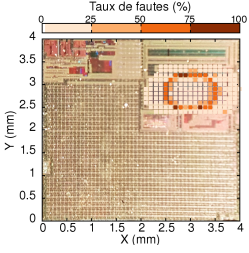


Plateforme Keysight P_{Ke}

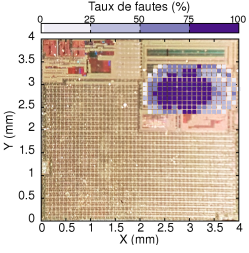


Distribution des fautes dans un bloc 128 bits

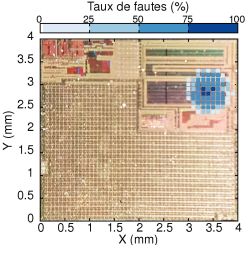
Impact du paramètre spatial sur le nombre de données en faute



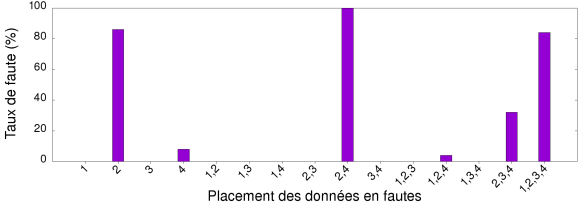
Une donnée altérée



Deux données altérées



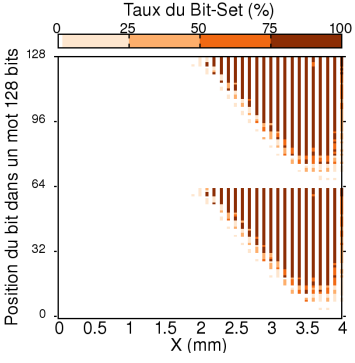
Quatre données altérées



Impact sur les données non-volatile

Résultats du balayage linéaire

X (mm)	Ligne de Données 128 bits							
	d4		d3		d2		d1	
	127 96		95 64		63 32		31 0	
1.9	80000000		00000000		00000000		00000000	
2.0	F0000000		00000000		F0000000		00000000	
2.1	C0000000		00000000		C0000000		00000000	
2.2	F8000000		00000000		F8000000		00000000	
2.3	FF800000		00000000		FF000000		00000000	
2.4	FFF80000		00000000		FFF00000		00000000	
2.5	FFFE0000		00000000		FFFC0000		00000000	
2.6	FFFFF000		00000000		FFFFE000		00000000	
2.7	FFFFFE00		00000000		FFFFFE00		00000000	
2.8	FFFFFFF0		00000000		FFFFFFE0		00000000	
2.9	FFFFFFFF		F8000000		FFFFFFFF		F0000000	
3.0	FFFFFFFF		E0000000		FFFFFFFF		80000000	
3.1	FFFFFFFF		F8000000		FFFFFFFF		F0000000	
3.2	FFFFFFFF		FFFF0000		FFFFFFFF		FFFE0000	
3.3	FFFFFFFF		FFF80000		FFFFFFFF		FFF00000	
3.4	FFFFFFFF		FFFFF800		FFFFFFFF		FFFFF000	
3.5	FFFFFFFF		FFFFFC00		FFFFFFFF		FFFFFC00	
3.6	FFFFFFFF		FFFFFE00		FFFFFFFF		FFFFFC00	
3.7	FFFFFFFF		FFFFE000		FFFFFFFF		FFFFC000	
3.8	FFFFFFFF		FFFF8000		FFFFFFFF		FFFF0000	
3.9	FFFFFFFF		FFFF8000		FFFFFFFF		FFFF0000	

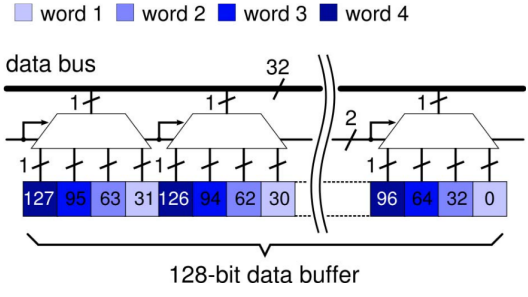


Positions des fautes bit-set en fonction des coordonnées de X

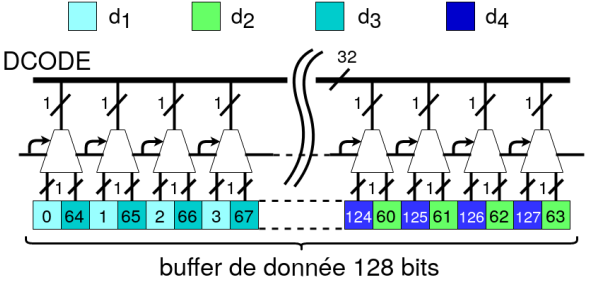
Modèle de faute bit-set validé
(modèle bit-set non observé)

Impact sur les données non-volatile

Hypothèse sur l'architecture



Hypothèse de l'architecture sur SAM3X8E [3]



Hypothèse de l'architecture sur STM32F4

Contre-mesures logicielles par redondance d'instruction

Moro *et al.* [4]

- idempotente $\rightarrow$ duplication d'instruction
- non-idempotente $\rightarrow$ décomposition en instructions idempotentes avec duplication
- specifique $\rightarrow$ séquence de remplacement avec duplication

Contre-mesures logicielles

Cas des instructions non-idempotentes - Analyse de résistance aux sauts multiples

```
/* Non-idempotente */
LDR.w R1,[R1]
/* Remplacement */
---- bloc0 ----
0 MOV.w R4,R1
1 MOV.w R4,R1
2 LDR.w R1,[R4]
3 LDR.w R1,[R4]
```

Contre-mesures logicielles

Cas des instructions non-idempotentes - Analyse de résistance aux sauts multiples

```
/* Non-idempotente */
LDR.w R1,[R1]
```

```
/* Remplacement */
---- bloc0 ----
0 MOV.w R4,R1
1 MOV.w R4,R1
2 LDR.w R1,[R4]
3 LDR.w R1,[R4]
```

```
/* Amélioration */
---- bloc0 ----
0 MOV.w R4,R1
1 inst. libre
2 inst. libre
3 inst. libre
---- bloc1 ----
4 MOV.w R4,R1
5 inst. libre
6 inst. libre
7 inst. libre
---- bloc2 ----
8 LDR.w R1,[R4]
9 inst. libre
10 inst. libre
11 inst. libre
---- bloc3 ----
12 LDR.w R1,[R4]
13 inst. libre
14 inst. libre
15 inst. libre
```

Contre-mesures logicielles

Cas des instructions non-idempotentes - Analyse de résistance aux sauts multiples

```
/* Non-idempotente */
LDR.w R1,[R1]
/* Remplacement */
---- bloc0 ----
0 MOV.w R4,R1
1 MOV.w R4,R1
2 LDR.w R1,[R4]
3 LDR.w R1,[R4]
```

```
/* Amélioration */
---- bloc0 ----
0 MOV.w R4,R1
1 inst. libre
2 inst. libre
3 inst. libre
---- bloc1 ----
4 MOV.w R4,R1
5 inst. libre
6 inst. libre
7 inst. libre
---- bloc2 ----
8 LDR.w R1,[R4]
9 inst. libre
10 inst. libre
11 inst. libre
---- bloc3 ----
12 LDR.w R1,[R4]
13 inst. libre
14 inst. libre
15 inst. libre
```

				Injection de faute durant le chargement		
				bloc0	bloc1	bloc2
Non-Idempotente	**Registre**	**Valeur Initiale**	**Attendue**	**Résultat**	**Résultat**	**Résultat**
sans amélioration	R4	0x0FF00000	0x01000360	0x0FF00000	–	–
	R1	0x01000360	0xF010010F	0x01000360	–	–
			Faute	(non protégée)	–	–

Contre-mesures logicielles

Cas des instructions non-idempotentes - Analyse de résistance aux sauts multiples

```
/* Non-idempotente */
LDR.w R1,[R1]
/* Remplacement */
---- bloc0 ----
0 MOV.w R4,R1
1 MOV.w R4,R1
2 LDR.w R1,[R4]
3 LDR.w R1,[R4]
```

```
/* Amélioration */
---- bloc0 ----
0 MOV.w R4,R1
1 inst. libre
2 inst. libre
3 inst. libre
---- bloc1 ----
4 MOV.w R4,R1
5 inst. libre
6 inst. libre
7 inst. libre
---- bloc2 ----
8 LDR.w R1,[R4]
9 inst. libre
10 inst. libre
11 inst. libre
---- bloc3 ----
12 LDR.w R1,[R4]
13 inst. libre
14 inst. libre
15 inst. libre
```

				Injection de faute durant le chargement		
				bloc0	bloc1	bloc2
Non-Idempotente	**Registre**	**Valeur Initiale**	**Attendue**	**Résultat**	**Résultat**	**Résultat**
sans amélioration	R4	0x0FF00000	0x01000360	0x0FF00000	–	–
	R1	0x01000360	0xF010010F	0x01000360	–	–
			Faute	(non protégée)	–	–
avec amélioration	R4	0x0FF00000	0x01000360	0x01000360	0x01000360	0x01000360
	R1	0x01000360	0xF010010F	0xF010010F	0xF010010F	0xF010010F
			Faute	protégée	protégée	protégée

References I

[1] A. BARENGHI et al. “Countermeasures against fault attacks on software implemented AES : effectiveness and cost”. In : *Proceedings of the 5th Workshop on Embedded Systems Security, WESS 2010*. 2010.

[2] J-M. DUTERTRE et al. “Experimental Analysis of the Laser-Induced Instruction Skip Fault Model”. In : *Secure IT Systems - 24th Nordic Conference, NordSec*. 2019.

[3] A. MENU et al. “Precise Spatio-Temporal Electromagnetic Fault Injections on Data Transfers”. In : *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. 2019, p. 1–8.

[4] N. MORO et al. “Formal verification of a software countermeasure against instruction skip attacks”. In : *Journal of Cryptographic Engineering* (2014).