EURECOM
Sophia Antipolis

(L')Embarqué entre Confiance et Défiance ?

Aurélien Francillon

SSTIC, Rennes Juin 2013

Pour ignorer cette publicité, cliquez ici [x]

Aurélien Francillon

- **Dans le passé**
  - Ingénieur puis thèse a l'INRIA Grenoble
  - ETH Zürich *(post doc)*
- **Maitre de conférences a EURECOM**
  - Depuis Sept. 2011

- **Eurecom**
  - Petite école d'ingénieurs a Sophia Antipolis
  - Membre de l'institut mines-télécom
- **2/3 d'étudiants étrangers**
- **A un nouveau diplôme cette année**
  - Diplôme d'ingénieur de spécialisation (CTI)

EURECOM
Sophia Antipolis

(L')Embarqué entre Confiance et Défiance ?

Aurélien Francillon

SSTIC, Rennes Juin 2013

Les questions que je me suis posé !

- **Pourquoi ce titre ?**
- **Pourquoi l'embarqué ?**
- **Comment faire confiance dans un système embarqué?**
- **Ils sont partout !**
- **Comment font ils pour faire ca de manière sécurisée?!?**
- **Peut on leur faire confiance?**
- **Pourquoi faire un système sécurisé?**

Pourquoi ce titre ?

Je me suis embarqué entre attaques et contre mesures

Mon but est donc

- **De raconter quelques histoires sur le sujet**
  - « Pot pourri »
- **Avec un point de vue assez personnel**
- **Sans trop redire des choses que vous aurez entendu ailleurs**
  - J'espère!

POURQUOI L'EMBARQUÉ

Pourquoi l'embarqué?

- Hors cartes a puces, l'idée d'attaques ciblées sur des systèmes embarqués était assez tirée par les cheveux
  - Ca a pas mal changé ces derniers temps…
- Curiosité face a des objets opaques
- Souvent petit, moins complexe
- Souvent aucune sécurité ☺

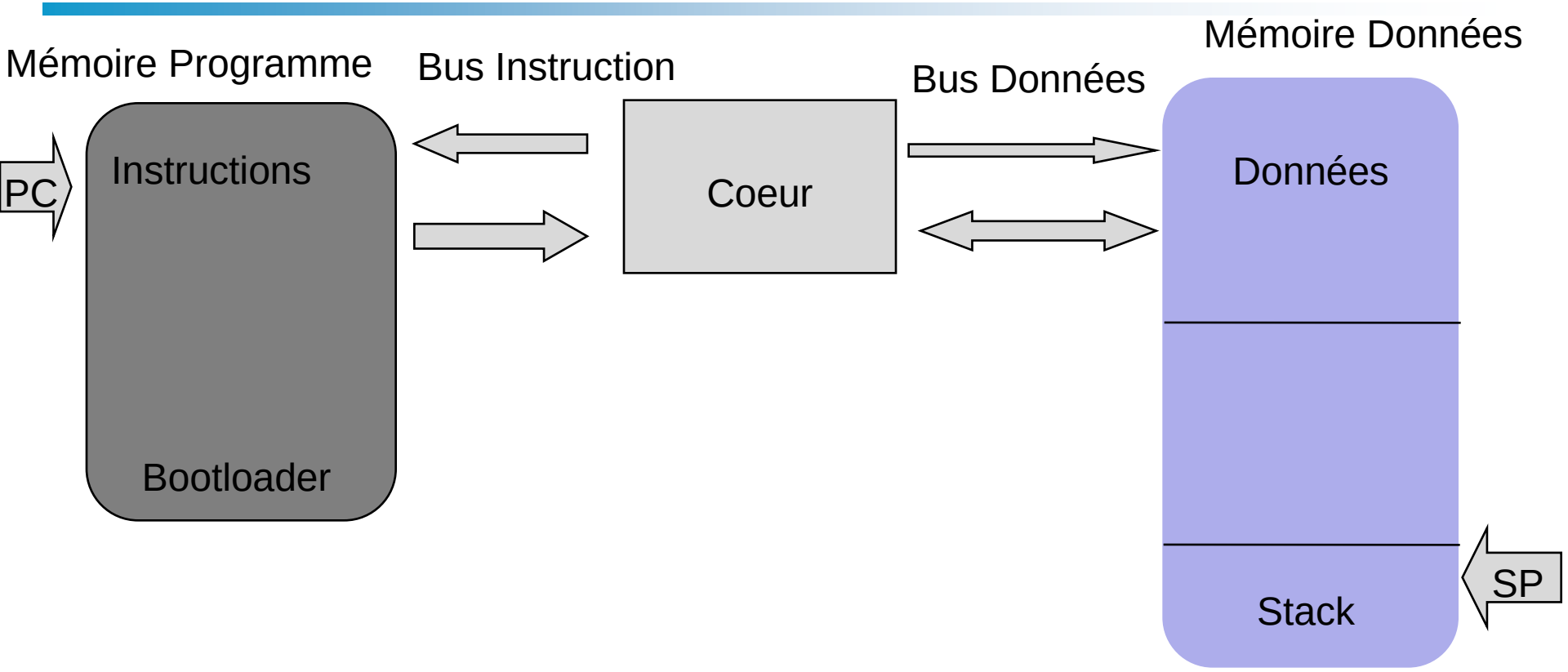
Qu'est ce qu'il fait ce système embarqué?

- **Déjà difficile avec programmes « normaux »**
  - unix/windows
- **Mais c'est pire quand :**
  - Il n'y a pas d'interface utilisateur
  - C'est du code dans un asm que l'on connait pas forcément
  - Qui tourne sur un OS inconnu ?
  - Sur on processeur dont on a pas le datasheet
  - Quand on (ou l'utilisateur) ne sait même pas que le système est la?
- **Est-ce que l'on peut faire confiance a cette boite noire?**

Comment il marche ce système?

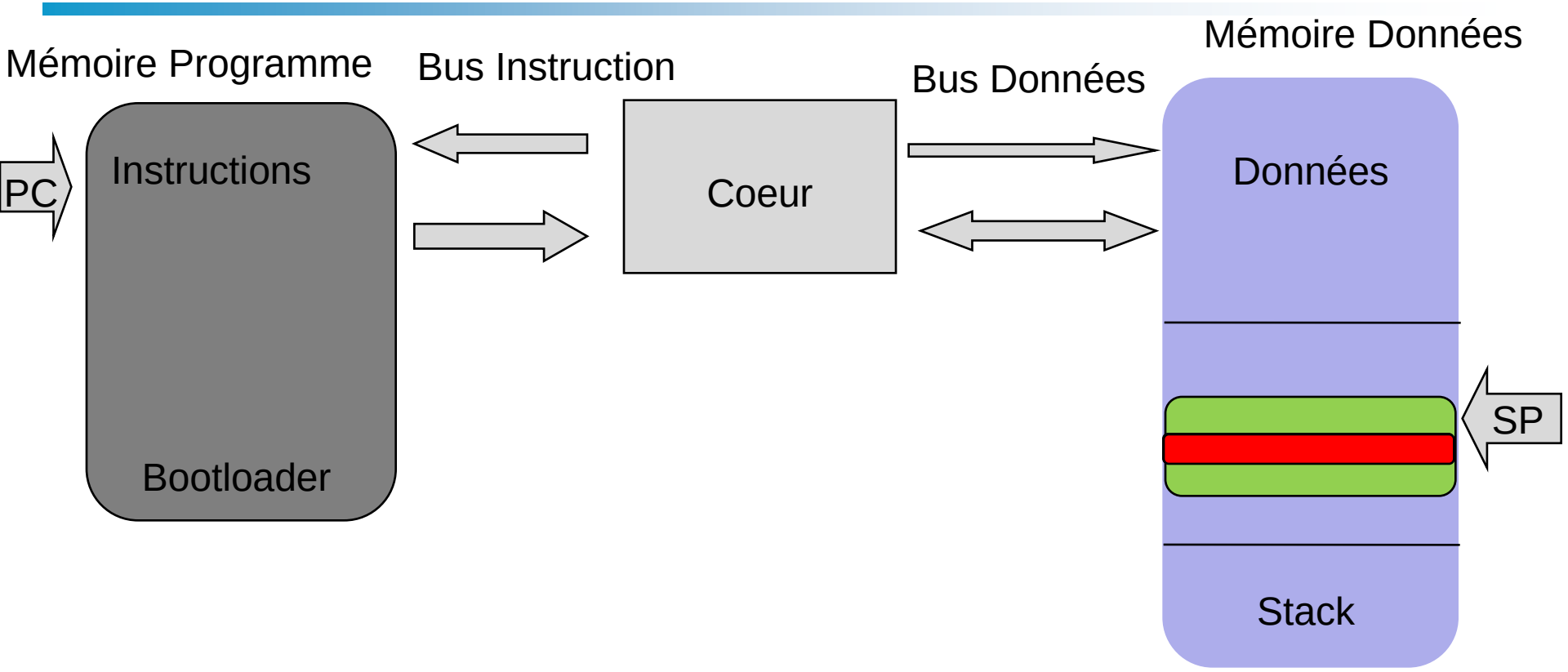
COMMENT EST ELLE EST ORGANISÉE CETTE MÉMOIRE?

Architecture Harvard



- **Un AVR…**
- **Les mémoires données et instructions sont indépendantes**

Injection de Code et Architecture Harvard

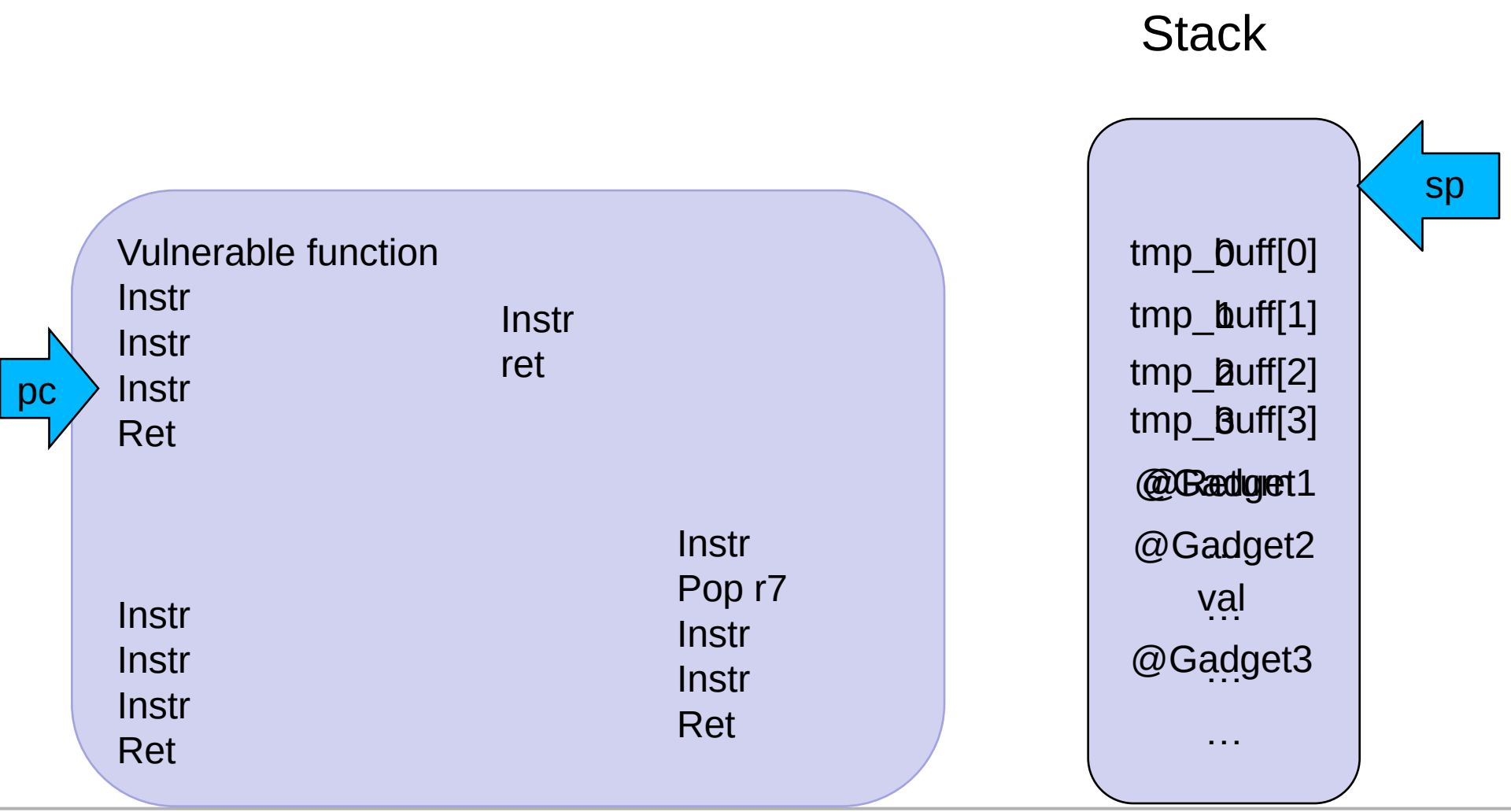


- **Les mémoires données et instructions sont séparées**
- **Peut on faire une injection de code classique ? Non…**

Return Oriented Programming

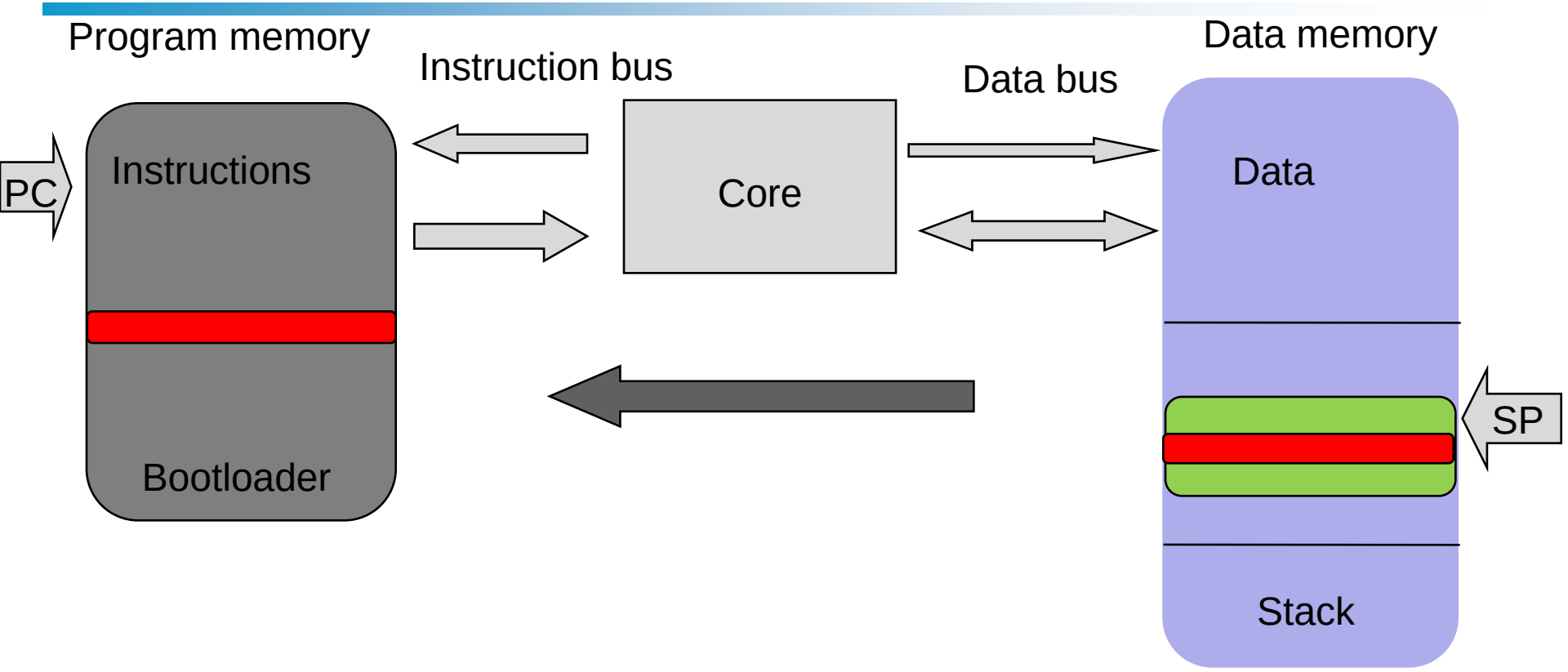
- **Généralisation des attaques “return to libc”**
  - Shacham CCS 2007 on the x86
- **Exécute du code déjà présent en mémoire programme**
  - « gadgets »
- **Ces gadgets peuvent être exécutés en séquence pour réaliser n’importe quelle « action »**
  - Si un jeu de gadgets “Turing Complet” est disponible

Return Oriented Programming (en 2 slides!)



EURECOM

Code Injection in Harvard Architecture
[ACM CCS 2008]



- **Démonstration que l'architecture Harvard architecture ne protège pas contre l'injection de code…**
- **Beaucoup d'astuces utilisées pour faire fonctionner ca…**

EURECOM

Comment faire confiance dans un système embarqué?

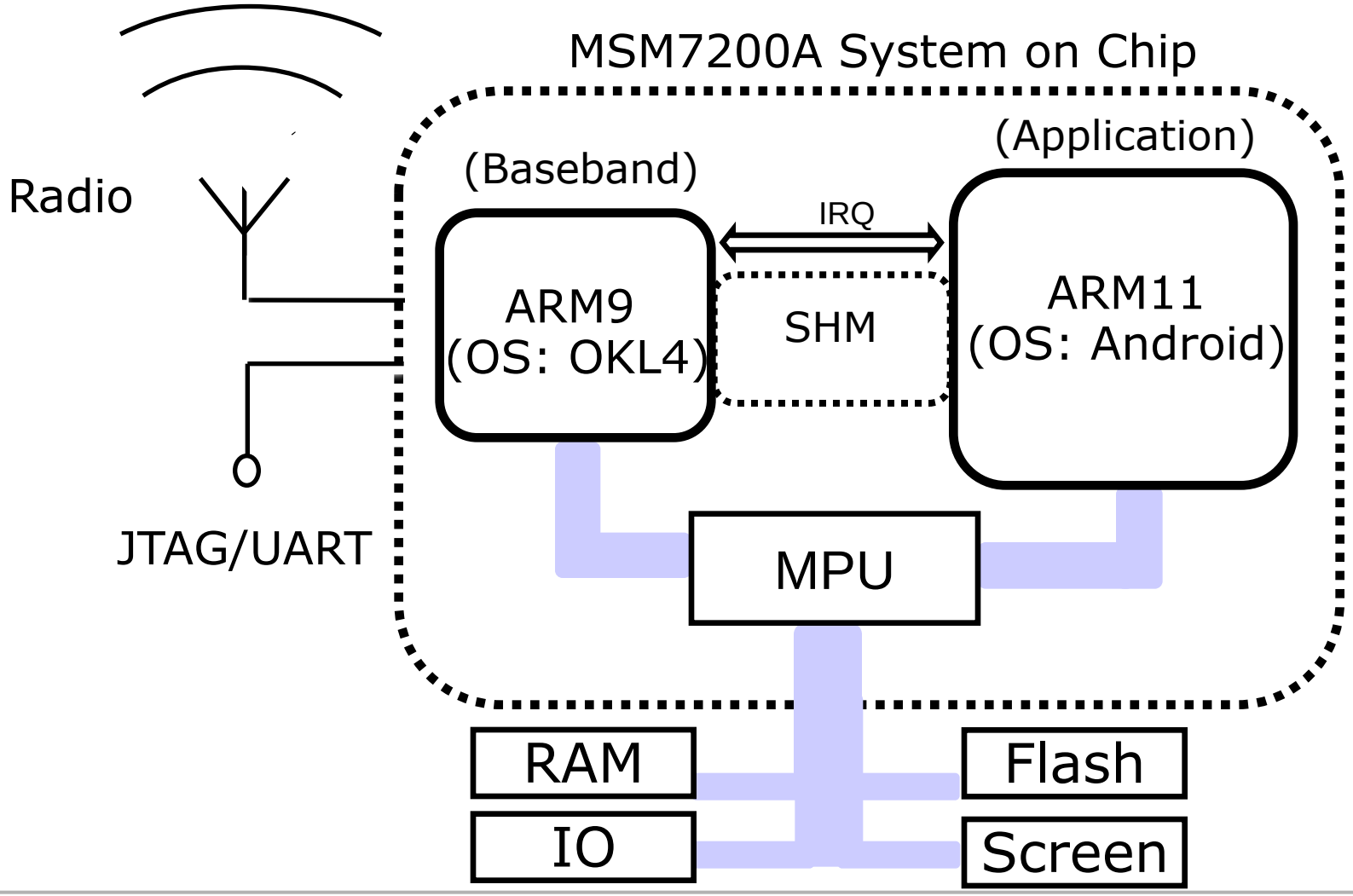
HTC G1

- **Similaire au système sur puce qui a été décrit dans**

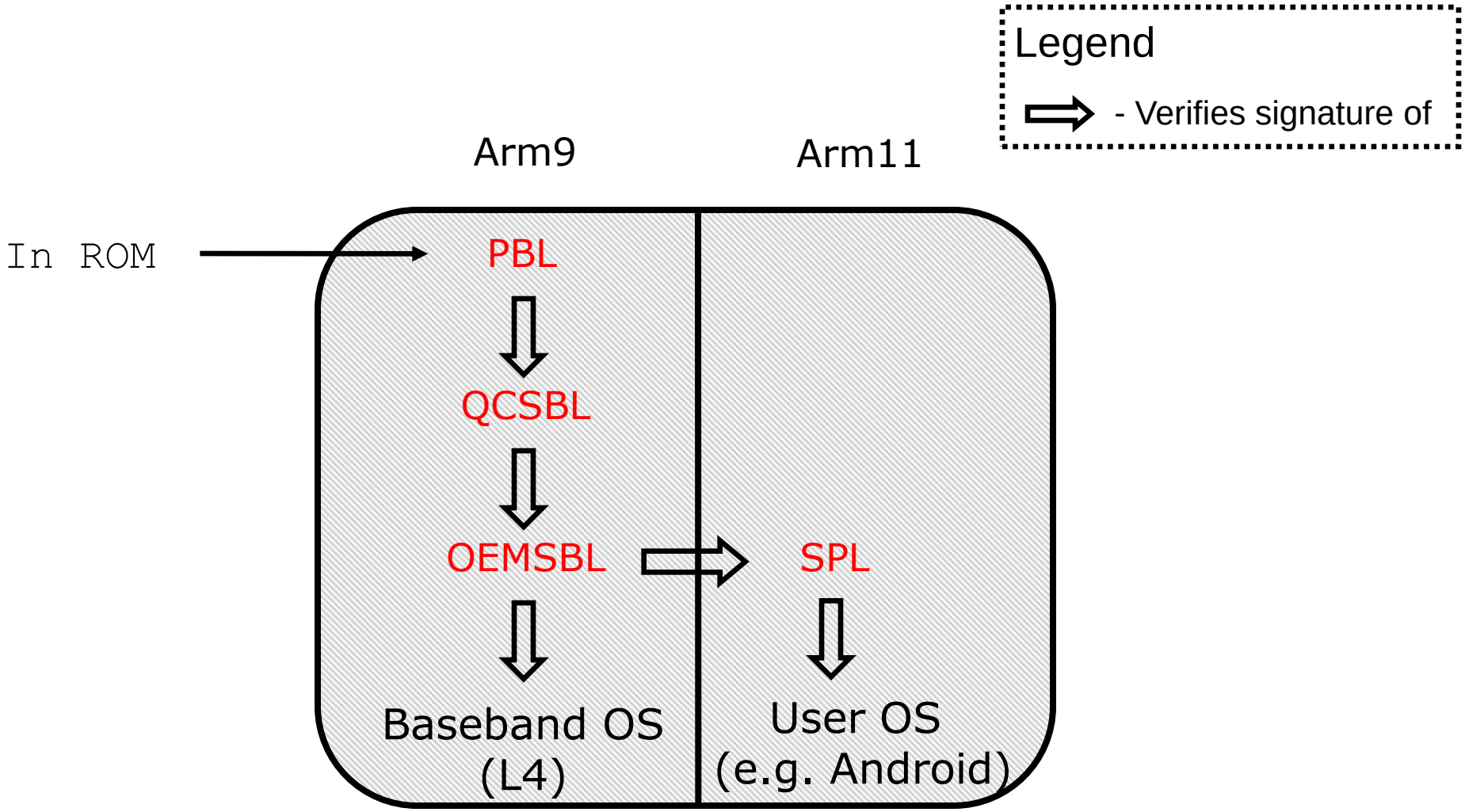
  « Rétroconception et débogage d'un baseband Qualcomm » Guillaume Delugré, SSTIC 2012

- **Mais dédié aux smartphones (v.s. clefs USB/3G)**

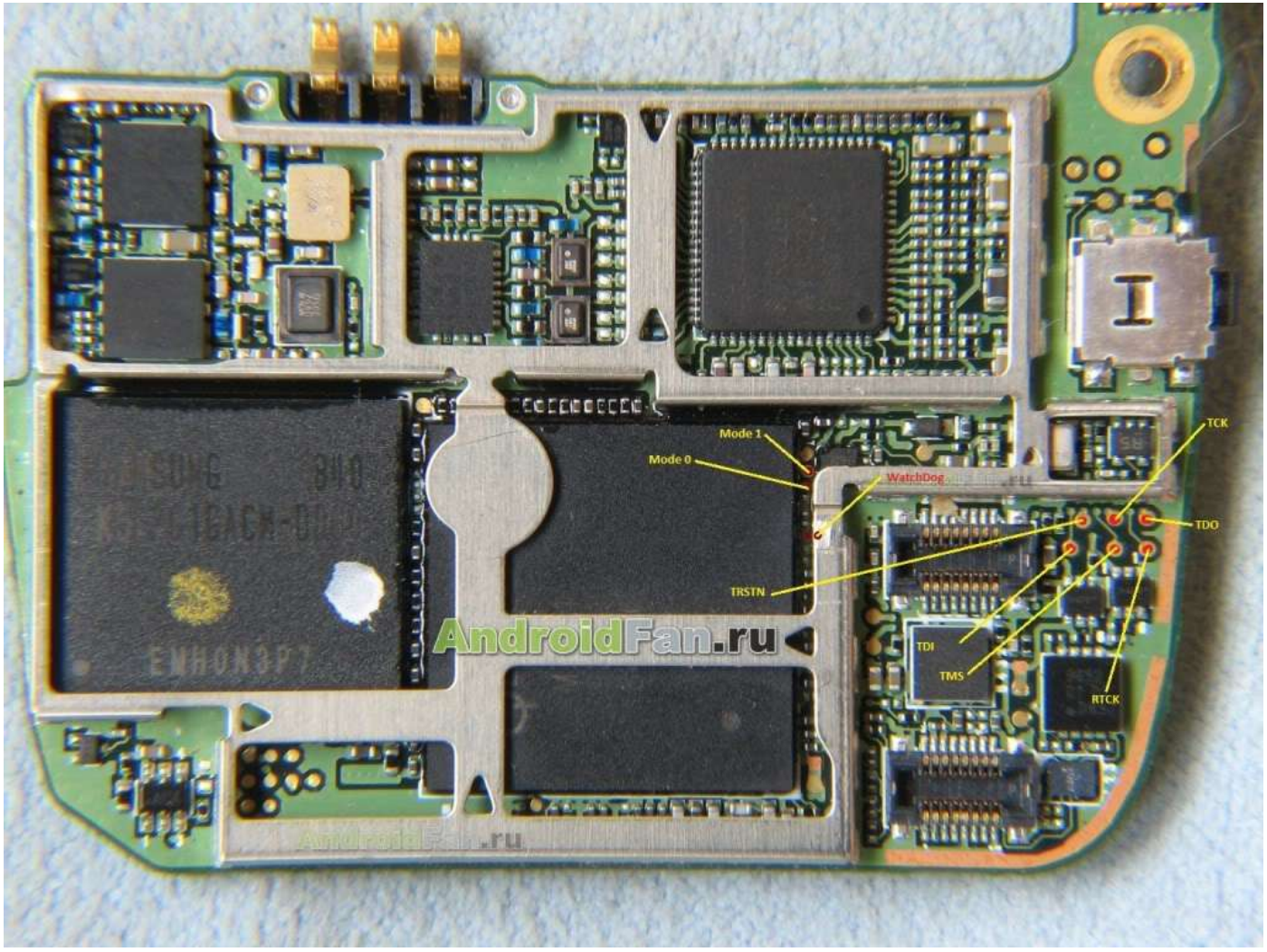
Architecture du HTC G1



Secure Boot


Legend
- Verifies signature of
Arm9
Arm11
In ROM
PBL
QCSBL
OEMSBL
SPL
Baseband OS
(L4)
User OS
(e.g. Android)

Avec un peu de chance... JTAG !



Avec L. Malisa

Le bootloader en mask ROM …

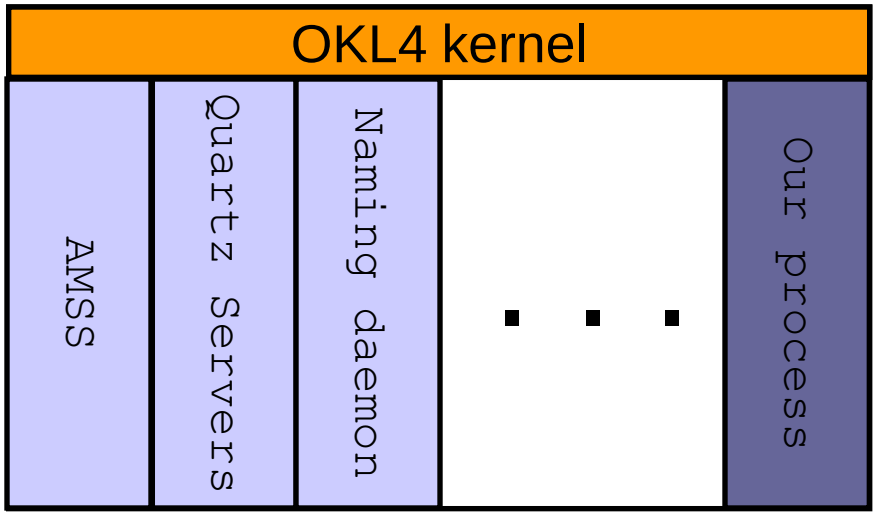
```
ficates                         ; CODE XREF: sub_FFFF24D4+28↓p
                                ; sub_FFFF2608+30↓p ...
    STMFD   SP!, {R4-R6,LR}
    LDR     R6, =0x8000605C ; address of CA Certificate in use
    MOV     R5, R0
    LDR     R1, [R6,#4]
    MOV     R0, #0
    STR     R1, [R5]
    LDR     R2, [R6]
    CMP     R2, #1              ; if cert == #1 ?
    MOVEQ   R0, #0              ; return 0
    LDMEQFD SP!, {R4-R6,PC}
    CMP     R1, #0
    MOVNE   R0, #1
    LDMNEFD SP!, {R4-R6,PC}
    MOV     R2, #0xB8000000
    LDR     R1, [R2,#0x950]
    AND     R1, R1, #0x1C0000 ; Bits 20:18 COM_GOV_SEL
                              ; Three fuses for majority vote encoding: 0 = Commercial, 1 =>
                              ; Government
    MOV     R1, R1,LSR#18
    CMP     R1, #3
    CMPNE   R1, #5
    CMPNE   R1, #6
    CMPNE   R1, #7
    LDREQ   R0, =certificate_GOV ; if 3/5/6/7 use certificate for government
    BEQ     loc_FFFF2434        ; store ROOT certificate  address
    LDR     R1, [R2,#0x938] ; SEC_BOOT_MODE
    TST     R1, #1
    BEQ     loc_FFFF243C
```

Backdoor ?

- **1 certificat => 1 gouvernement ?**
- **Une backdoor ?**
  - Accès physique pour changer les fuses ?
  - Le SoC est probablement aussi vendu pour construire des téléphones certifiés/durcis
- **Mais le « gouvernement » peut quand même reprogrammer n'importe quel téléphone avec son propre baseband...**

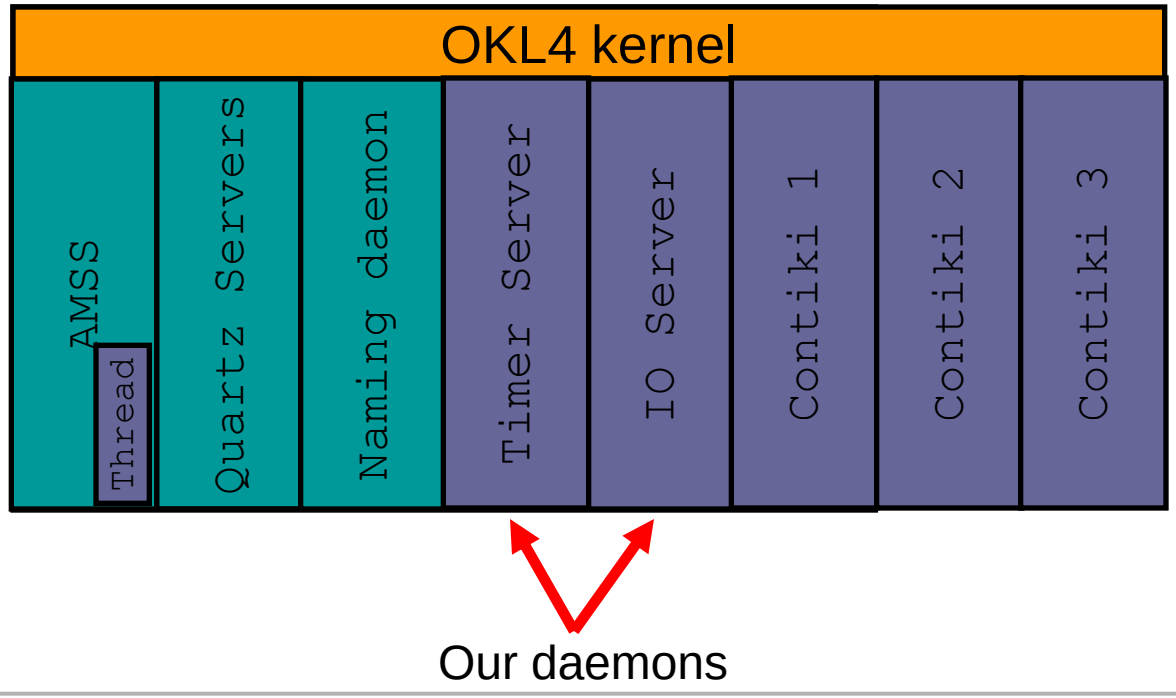
Encore faudrait il pouvoir modifier l'image !

- **Le baseband tourne sur un micro noyau L4: OKL4**
- **Dont des versions antérieures sont disponibles en open source…**
- **OKL4 peut être utilisé comme un hyperviseur**



Preuve de concept

- **Preuve de concept**
  - Il y a une fenêtre entre la vérification de la signature et la désactivation du JTAG !
  - Injection de plusieurs instances de Contiki OS dans un baseband



Linux on the Baseband

```
[MSM_STDOUT] icache lockdown register = 0
[MSM_STDOUT] domain pairs: (0, 1)
[MSM_STDOUT] Initialising scheduler...
[MSM_STDOUT] Switching to idle thread
[MSM_STDOUT] system has 9 interrupts
[MSM_STDOUT] Initializing root servers
[MSM_STDOUT] root-servers: utcb_area: cff00100 (64KB), kip_area: ff000000 (0KB)
[MSM_STDOUT] creating root server (000a4001)
[MSM_STDOUT] Linux version 2.6.10-arm (milkfairy@sunbird) (gcc version 3.4.4) #26 Mon Apr 11 1
3:48:58 CEST 2011
[MSM_STDOUT] Kernel memory ranges:
[MSM_STDOUT]   0: 0xb0380000-0xb03ef000 (111 pages)
[MSM_STDOUT]   total 111 pages
[MSM_STDOUT] Built 1 zonelists
[MSM_STDOUT] Kernel command line: igms_name=ramdisk root=/dev/igms0
[MSM_STDOUT] PID hash table entries: 32 (order: 5, 512 bytes)
[MSM_STDOUT] Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
[MSM_STDOUT] Inode-cache hash table entries: 512 (order: -1, 2048 bytes)
[MSM_STDOUT] Memory: 348k/4100k available (1161k kernel code, 3724k reserved, 234k data, 56k i
nit)
```

ILS SONT PARTOUT !

Ceci n'est pas un PC

Mais une combinaison de systèmes embarqués

En particulier on a regardé un Disque Dur

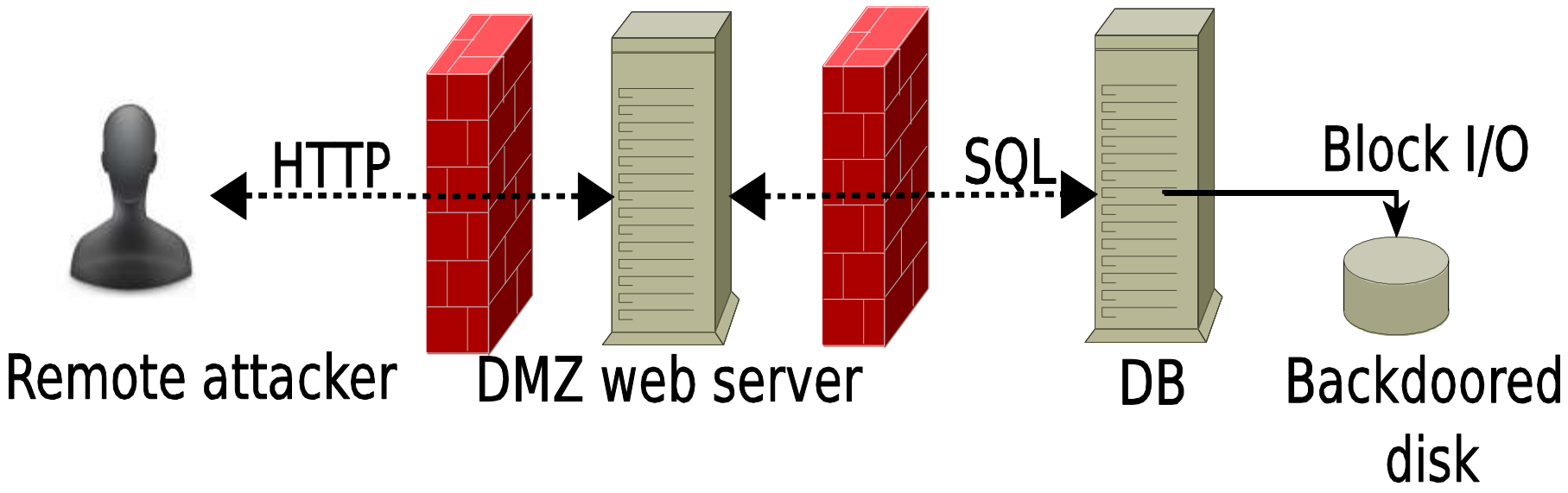
- **Est un système embarqué a part entière**
  - Pas mal de choses sont faites en logiciel !
- **Que peut il se passer si un firmware modifié est installé ?**
  - Ils ne sont pas signés
  - Peuvent être mis à jour grâce a un simple utilitaire …

Avec J. Zaddach A. Kurmus, etc …

Apres « un peu » d'instrumentation et de reverse

Avec J. Zaddach A. Kurmus, etc …

Une backdoor dans le disque pour exfiltrer des données



Avec J. Zaddach A. Kurmus, etc …

MAIS COMMENT FONT ILS POUR FAIRE CA DE MANIÈRE SÉCURISÉE ??!?

Une histoire vraie ☺

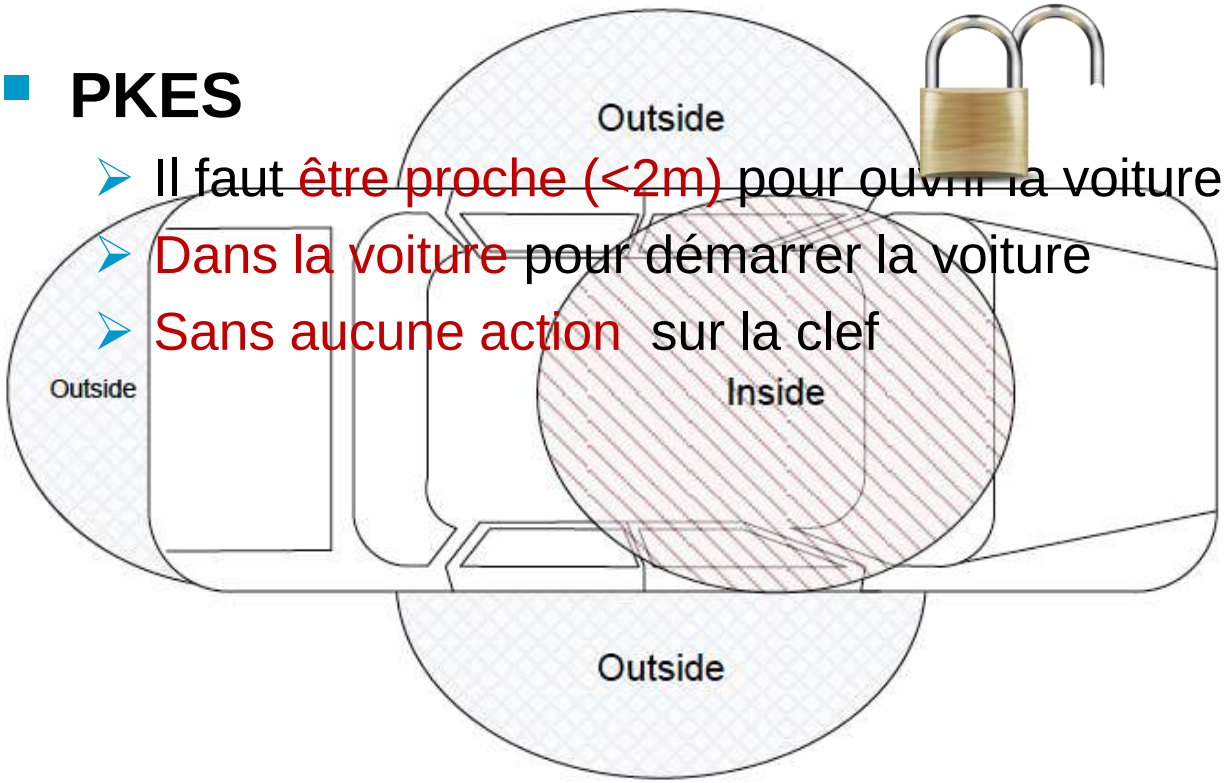
- **Groupe syssec @ ETH zurich**
  - années de recherche dans le domaine de la sécurité wireless
    - Distance bounding: Mesure de manière sécurisée de la distance

- **Srdjan achète une voiture avec une « clef sans fil passive »**
  - Comment ca marche ?

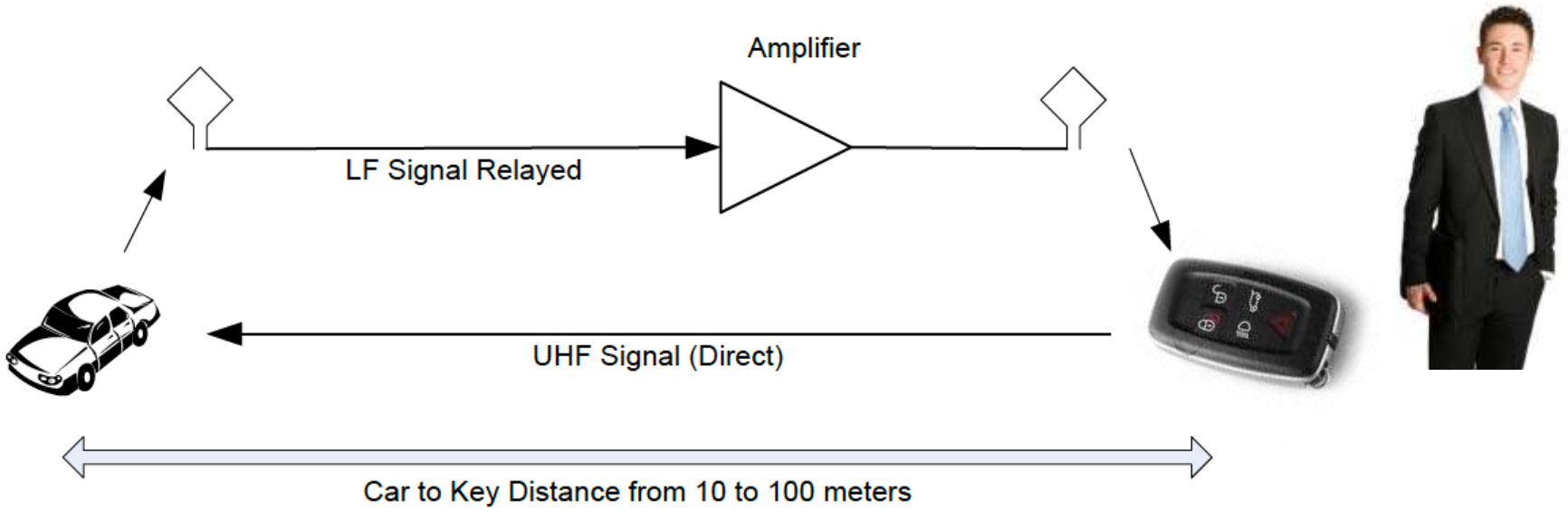
- **On regarde sa voiture d'un peu plus près …**

Relay on Passive Keyless Entry and Start
[NDSS 11]

- **PKES**
  - Il faut être proche (<2m) pour ouvrir la voiture
  - Dans la voiture pour démarrer la voiture
  - Sans aucune action sur la clef



Relay: Over a Cable
[NDSS 11]



- **La clef croit être proche si elle reçoit le signal de la voiture**
- **Mais les signaux peuvent être relayés…**
- **Par câble ou en relai sans fils**
- **Au niveau physique, très rapide**

Relay: Analysis on 10 Models
[NDSS 11]

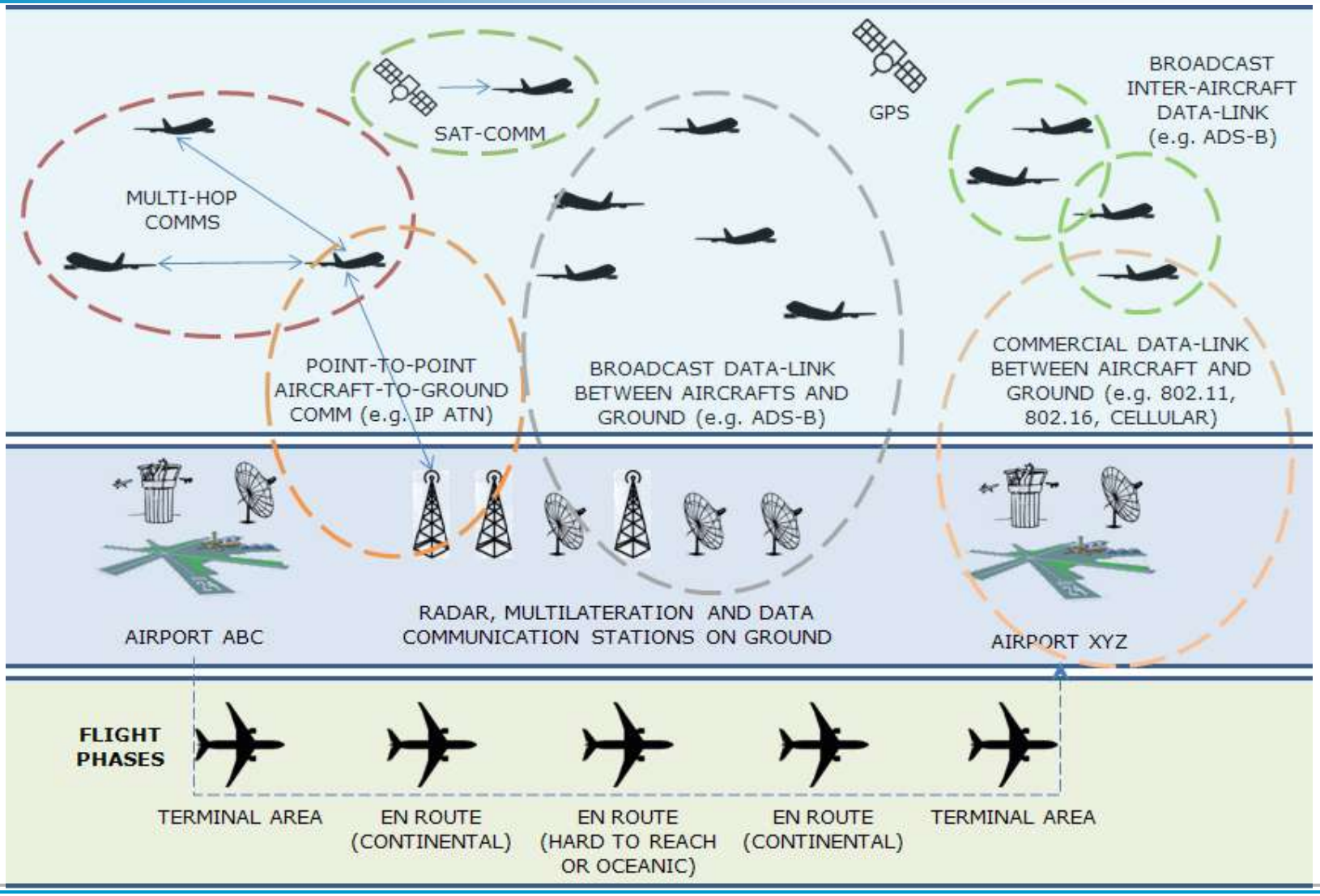
- **Analyses de plusieurs modèles**
  - 10 modèles de 8 fabricants
  - Tous utilisent la même techno (LF/UHF)

- **Aucun avec le même protocole**
- **Certains avec des messages longs**
  - Crypto forte ?

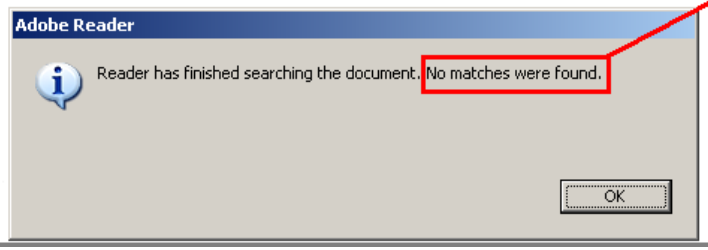
COMMENT FONT ILS POUR FAIRE CA DE MANIÈRE SÉCURISÉ ??!? (BIS)

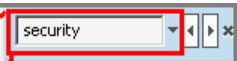
ADS-B : Ghost is in the Air (Traffic)

[Costin, Francillon, Blackhat 2012]



Minimum Operational Performance Standards for Universal Access Transceiver (UAT) Automatic Dependent Surveillance - Broadcast (ADS-B)





PEUT ON LEUR FAIRE CONFIANCE?

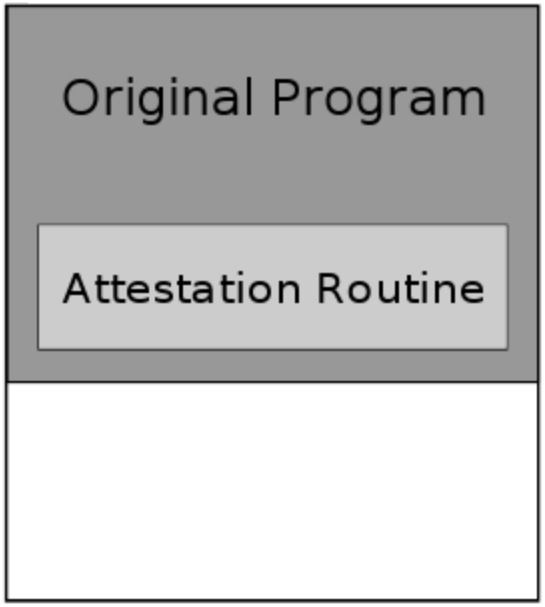
Vérifier le logiciel avec du logiciel?

- **« Software-based attestation »**
- **Motivation:**
  - Vérifier, a distance, n'importe quel système
  - Sans support dédié (TPM…)
- **Vérifier du logiciel avec du logiciel**
  - Possible ?

EURECOM

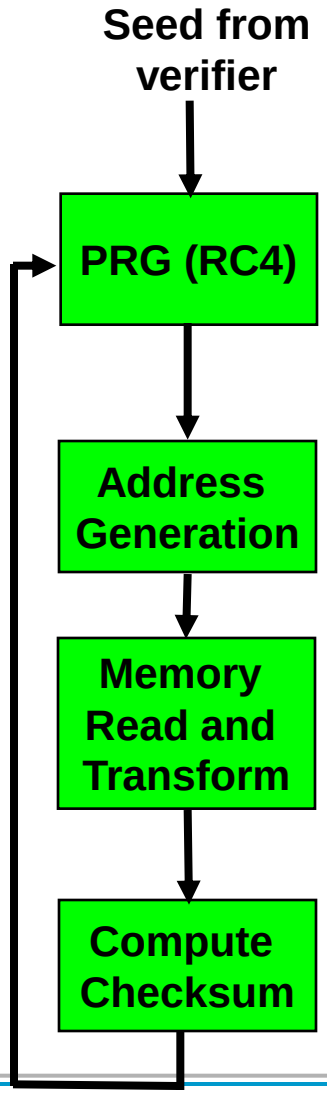
Software-based Attestation Temps

- **Plusieurs approches**
- **Ex. en utilisant le temps d'exécution**
  - Attestation optimisée pour être faite en temps constant
  - Modifications code détectées par un délai supplémentaire
  - Access mémoire aléatoires
  - …
- **Contraintes de temps fortes:**
  - Un seul hop
  - Communications fiables



SWATT Assembly Code

NB: This is just an example I'm not expecting you to understand the code



Generate ith member of random sequence using RC4

zh = 2	ldi zh, 0x02
r15 = *(x++)	ld r15, x+
yl = yl + r15	add yl, r15
zl = *y	ld zl, y
*y = r15	st y, r15
*x = r16	st x, r16
zl = zl + r15	add zl, r15
zh = *z	ld zh, z

Generate 16-bit memory address

zl = r6	mov zl, r6

Load byte from memory and compute transformation

r0 = *z	lpm r0, z
r0 = r0 xor r13	xor r0, r13
r0 = r0 + r4	add r0, r4

Incorporate output of hash into checksum

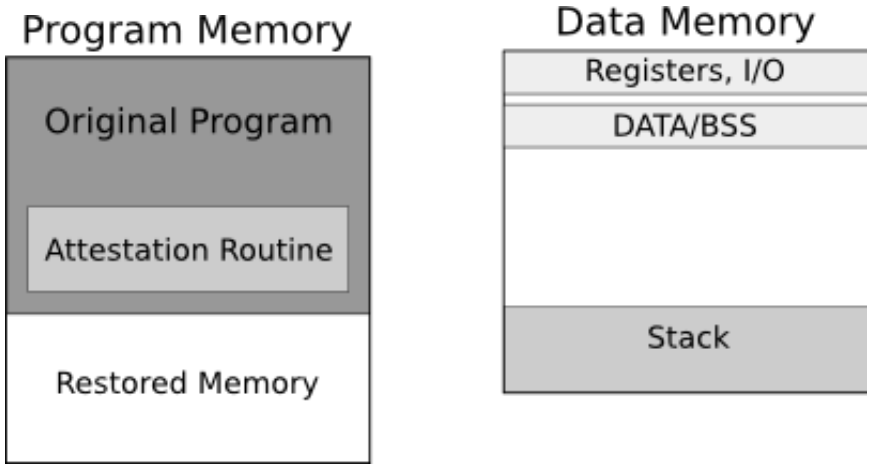
r7 = r7 + r0	add r7, r0
r7 = r7 << 1	lsl r7
r7 = r7 + carry_bit	adc r7, r5
r4 = zh	mov r4, zh

This slide is © A. Perrig

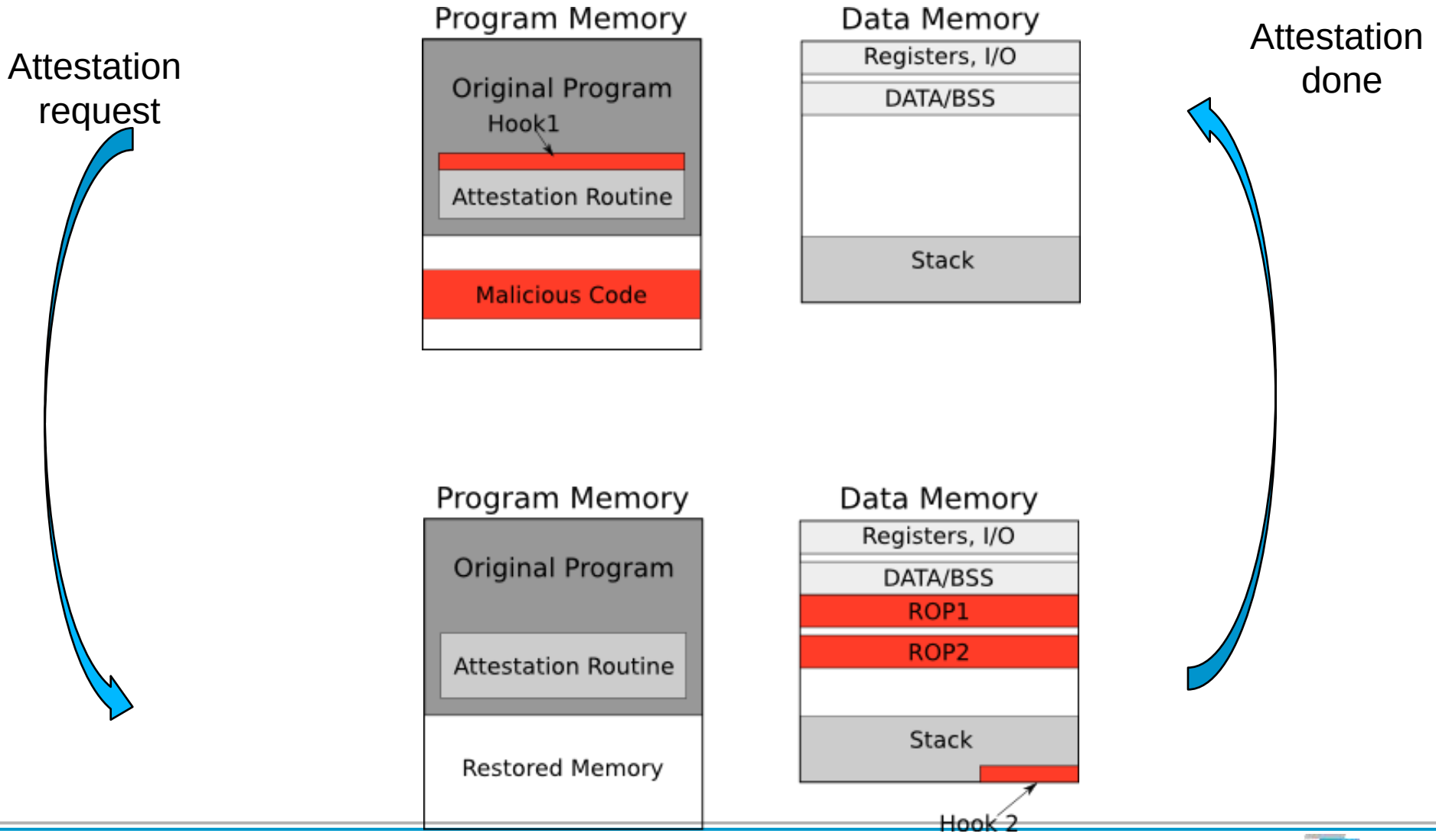
Attacks on timing based attestation

- **Difficultés**
  - Code optimal
  - Quelle est la meilleure attaque ?
    - Délai maximum acceptable ?
- **Return-oriented programming**
  - Permet de faire des choses sans changer le « code »
  - Peut être utilisé pour cacher des choses…

Software-based Attestation: Clean System



Return Oriented Rootkit attack
[ACM CCS 2009]



Alors qu'est ce qu'il faut vérifier?

- **Vérifier l'intégrité au chargement ne suffit pas**
- **Dynamiquement**
  - Vérifier le code, ok mais les données?

- **« Dynamic root of trust »**
  - Faire les étapes importantes en isolation du reste du système
  - En garantissant l'isolation et l'intégrité par du matériel
  - … malgré une compromission complète du système

SMART

SECURE AND MINIMAL ARCHITECTURE FOR ESTABLISHING A DYNAMIC ROOT OF TRUST

Joint work with: Karim El Defrawy, Daniele Perito, Gene Tsudik
NDSS 2012

SMART in a Nutshell

- **Changements simples de l'architecture du CPU**
- **Buts:**
  - Authentification
  - Attestation d'une région de code
  - Trusted Execution:
    - On sait que telle code a été exécuté
- **Reporting, pas de l'enforcement !**

SMART main Idea

"Interface utilisateur": Smart comme un appel de fonction:

SMART(`a, b, x, jmp?, nonce, &result, param`**);**

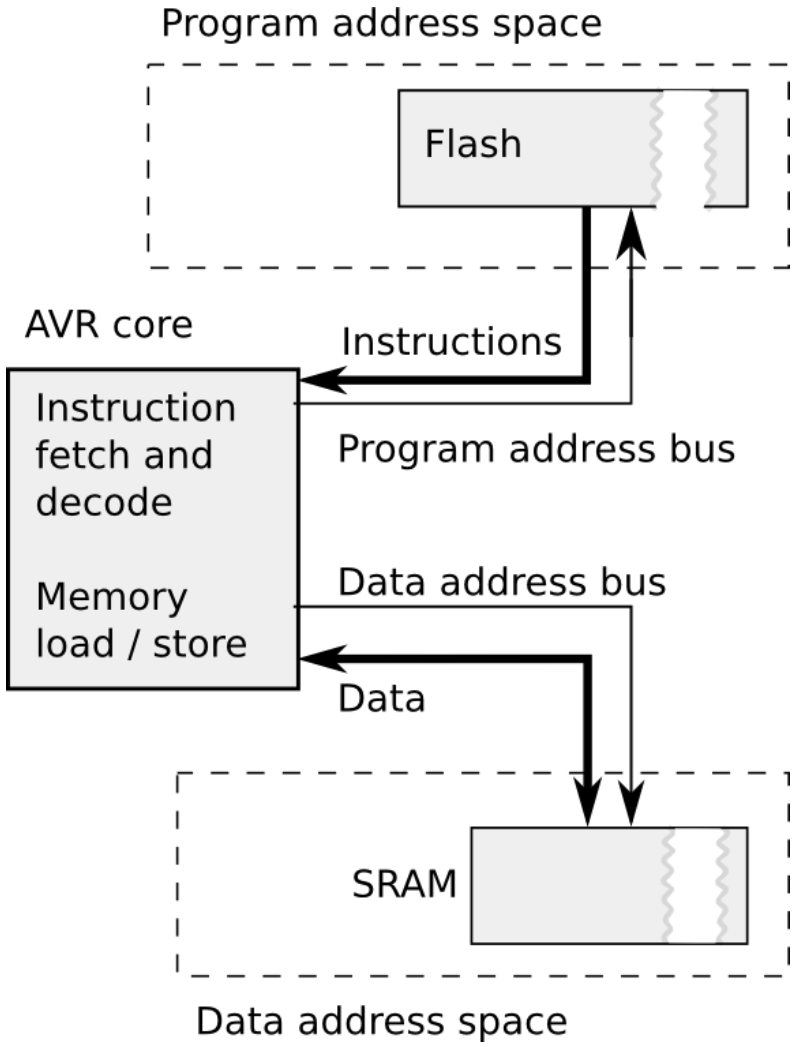
- **HMACde la mémoire [**`a,b`**] avec le** `nonce`
- **Si** `jmp?== 1` **jump to** `x` **après l'exécution de SMART avec le paramètre** `param`
- **Ecrit le résultat du HMAC a adresse** `&result`

EURECOM

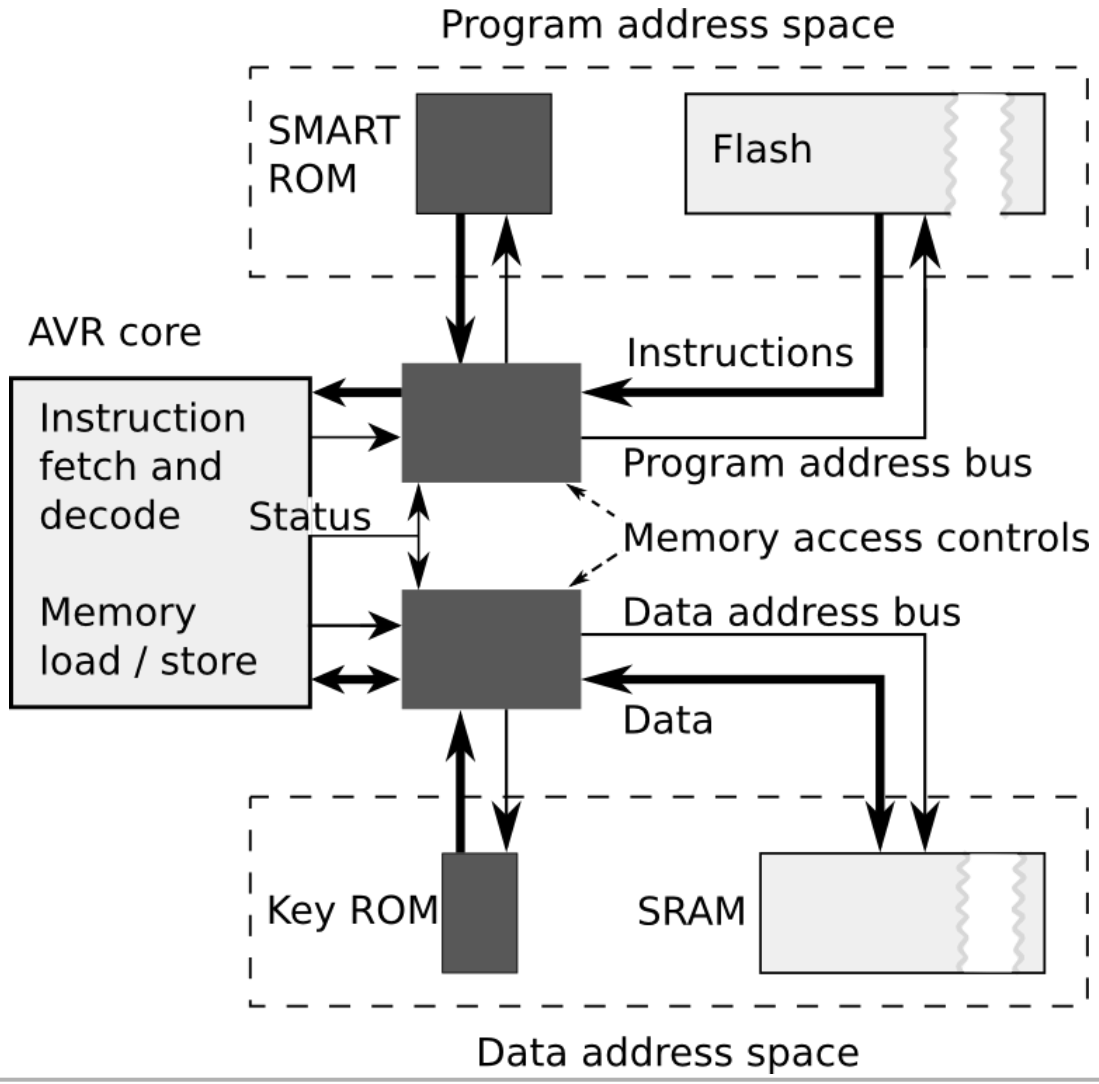
SMART building blocks

- **Secure Key Storage**
  - Seul le code SMART a accès a la clef
- **Mémoire Read-Only**
  - ROM dans le microcontrôleur
  - Calcule un HMAC (ou autre…)
- **Contrôles d'Accès**
  - Détecte accès a la clef
  - Prévient l'abus de la ROM
    - like ret-to-libc, ROP

SMART : Lightweight Hardware Dynamic Root of Trust



SMART : Lightweight Hardware Dynamic Root of Trust



POURQUOI FAIRE UN SYSTÈME SÉCURISÉ?

Pourquoi faire un système sécurisé?

- **Ca coute plus cher, c'est plus dur…**
- **Protéger l'utilisateur ?**
  - Est-ce que ca va permettre de vendre plus ? Plus vite ?
  - Qui est le client ? Que veut il protéger ?
- **Je croyais que ca n'existait plus**
  - Une discussion récente: notre système est sur, il y a AES dans le standard...
- **Une question de motivations !**
  - Y a-t-il des attaques existantes sur le système ?

Comment conclure ?

- **Par quelques conseils**
  - rester curieux
  - se poser des questions face a des boites noires …
  - … ou des choses que l'on ne comprends pas

- **Merci a**
  - Luka Malisa
  - Jonas Zaddach
  - Boris Danev
  - …

Questions ?

Publicité :

- **CARDIS 2013 a Berlin**
- **Co-chair avec Pankaj Rohatgi (Cryptography Research)**
  - Deadline 10 septembre 2013
  - Conférence 27-29 Novembre, 2013

References

- K. El Defrawy, A. Francillon, D. Perito, G. Tsudik. **SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust.** NDSS 2012
- Claude Castelluccia, Aurélien Francillon, Daniele Perito, and Claudio Soriente. **On the difficulty of software-based attestation of embedded devices.** CCS '09
- Aurélien Francillon, Quan Nguyen, Kasper B. Rasmussen, Gene Tsudik "**Systematic Treatment of Remote Attestation**", http://eprint.iacr.org/2012/713
- A. Francillon, B. Danev, S. Capkun. "**Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars"** NDSS 2011
- A. Francillon and C. Castelluccia "**Code injection attacks on Harvard-architecture devices"**, CCS '08