Vers la sécurité mobile :

caractérisation des attaques et contremesures

Jean-François Lalande

Soutenance d'HDR
30 novembre 2017

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES CENTRE VAL DE LOIRE
UNIVERSITE D'ORLEANS
LIFO

10 années de recherche en sécurité

Avant 2005 :

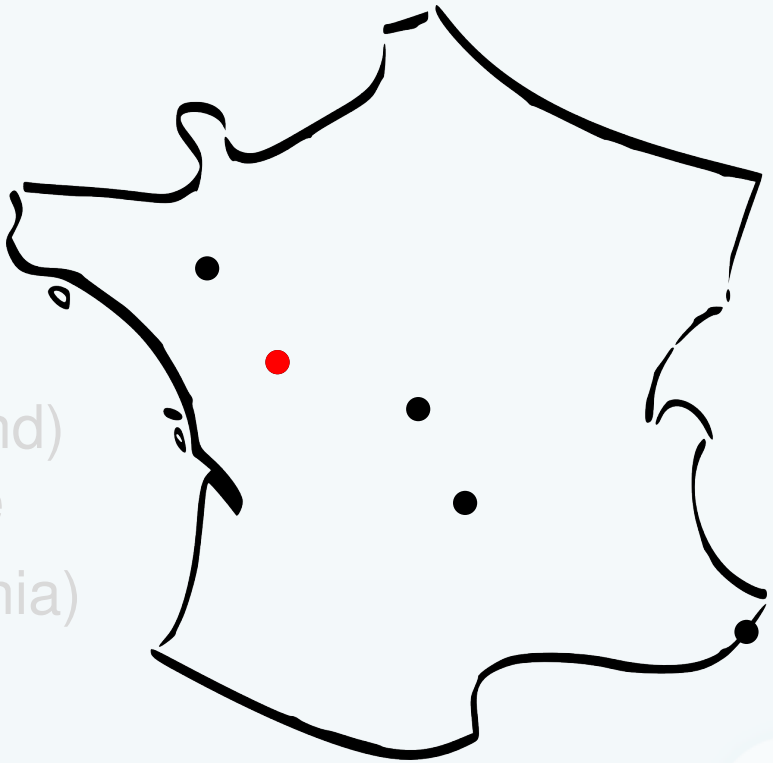
- Optimisation pour les réseaux (fibres optiques, satellites)
- Développement pour l'algorithmique de graphes

Depuis 2005 :

- Sécurité des systèmes à large échelle
  - Systèmes hautes performances
  - Systèmes collaboratifs
- Sécurité des systèmes embarqués :
  - Téléphones mobiles
  - Cartes à puces

Parcours

84 Ordinateur en carton
92 Club informatique
96 BAC S (Segré - 49)
96-98 MPSI + MP (Rennes)
98-01 Ingénieur ISIMA (Clermont Ferrand)
+ DEA Optimisation / Productique
01-04 Doctorat Univ. de Nice (Inria Sophia)
04-05 ATER (Polytech'Nice)
05-13 MCF (INSA CVL)
13-15 Délég. Inria (CentraleSupélec Rennes)
15-16 MCF (INSA CVL)



Parcours

84 Ordinateur en carton

92 Club informatique

96 BAC S (Segré - 49)

96-98 MPSI + MP (Rennes)

98-01 Ingénieur ISIMA (Clermont Ferrand)
+ DEA Optimisation / Productique

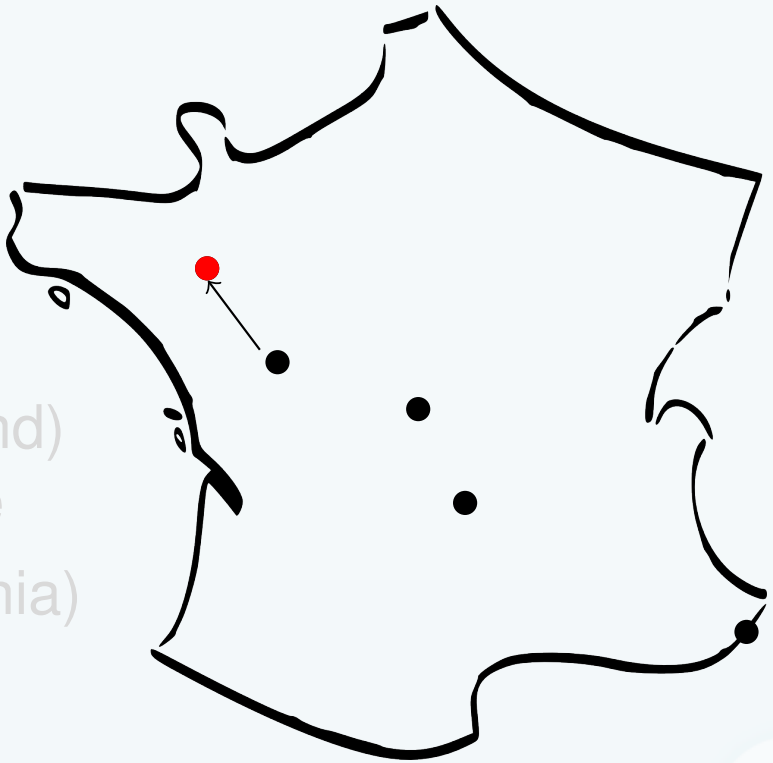
01-04 Doctorat Univ. de Nice (Inria Sophia)

04-05 ATER (Polytech'Nice)

05-13 MCF (INSA CVL)

13-15 Délég. Inria (CentraleSupélec Rennes)

15-16 MCF (INSA CVL)



Parcours

84 Ordinateur en carton

92 Club informatique

96 BAC S (Segré - 49)

96-98 MPSI + MP (Rennes)

98-01 Ingénieur ISIMA (Clermont Ferrand)
+ DEA Optimisation / Productique

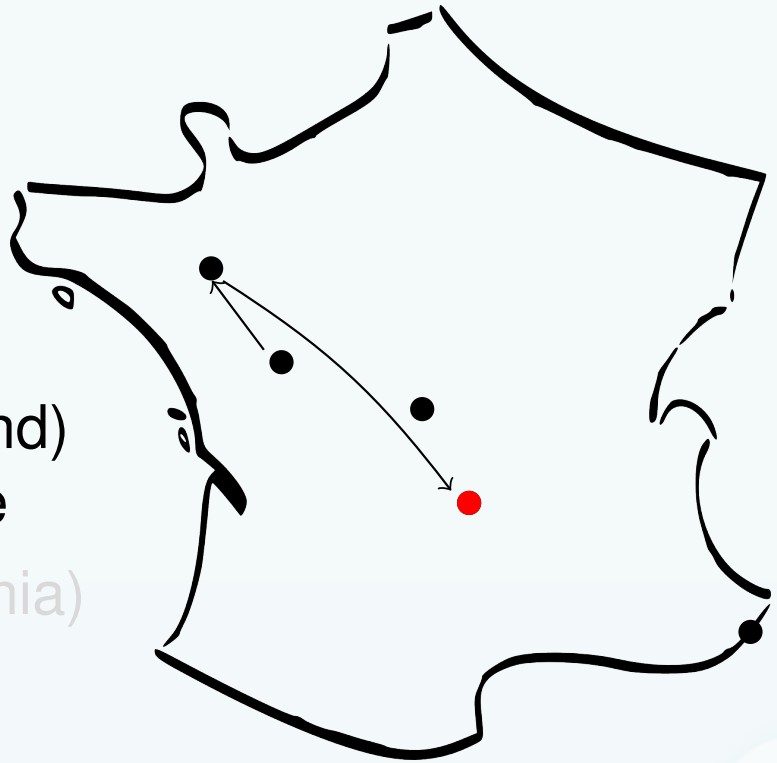
01-04 Doctorat Univ. de Nice (Inria Sophia)

04-05 ATER (Polytech'Nice)

05-13 MCF (INSA CVL)

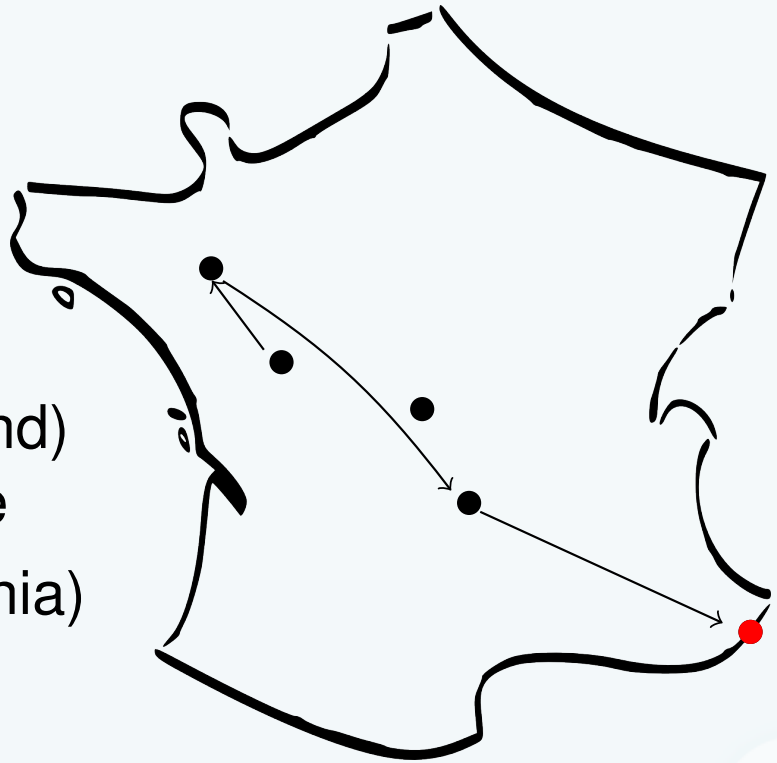
13-15 Délég. Inria (CentraleSupélec Rennes)

15-16 MCF (INSA CVL)



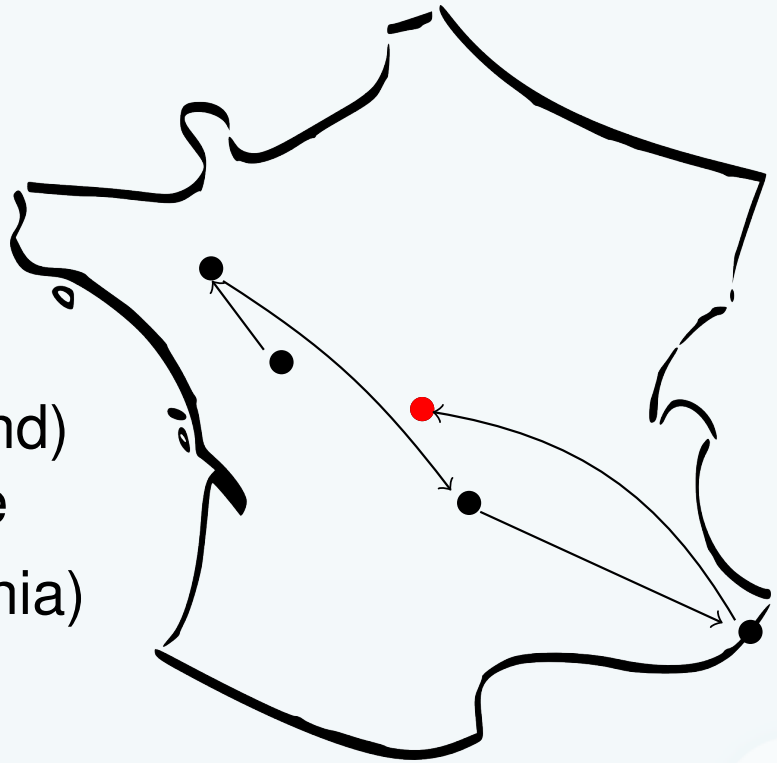
Parcours

- 84 Ordinateur en carton
- 92 Club informatique
- 96 BAC S (Segré - 49)
- 96-98 MPSI + MP (Rennes)
- 98-01 Ingénieur ISIMA (Clermont Ferrand) + DEA Optimisation / Productique
- 01-04 Doctorat Univ. de Nice (Inria Sophia)
- 04-05 ATER (Polytech'Nice)
- 05-13 MCF (INSA CVL)
- 13-15 Délég. Inria (CentraleSupélec Rennes)
- 15-16 MCF (INSA CVL)



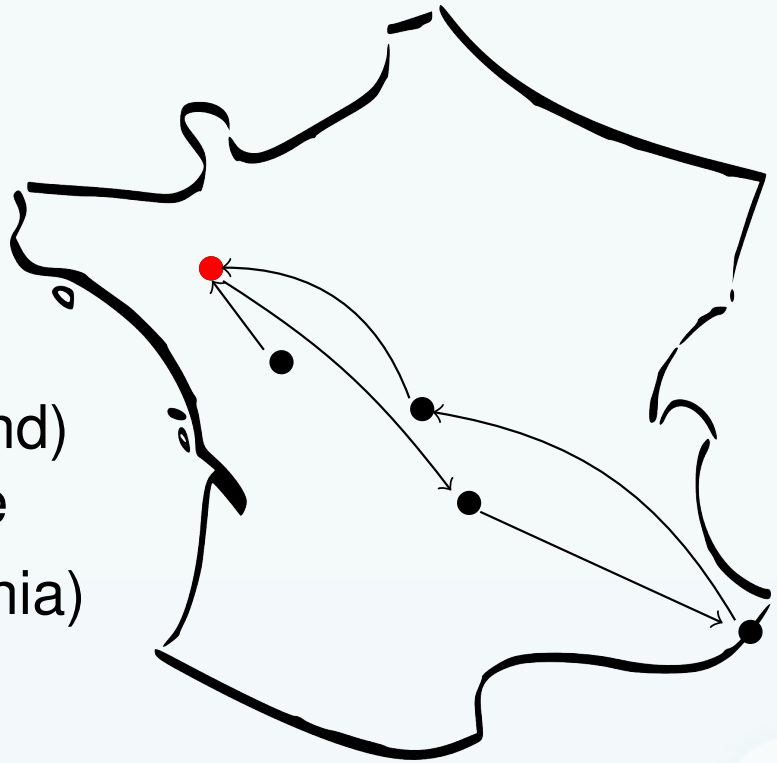
Parcours

- 84 Ordinateur en carton
- 92 Club informatique
- 96 BAC S (Segré - 49)
- 96-98 MPSI + MP (Rennes)
- 98-01 Ingénieur ISIMA (Clermont Ferrand) + DEA Optimisation / Productique
- 01-04 Doctorat Univ. de Nice (Inria Sophia)
- 04-05 ATER (Polytech'Nice)
- 05-13 MCF (INSA CVL)
- 13-15 Délég. Inria (CentraleSupélec Rennes)
- 15-16 MCF (INSA CVL)



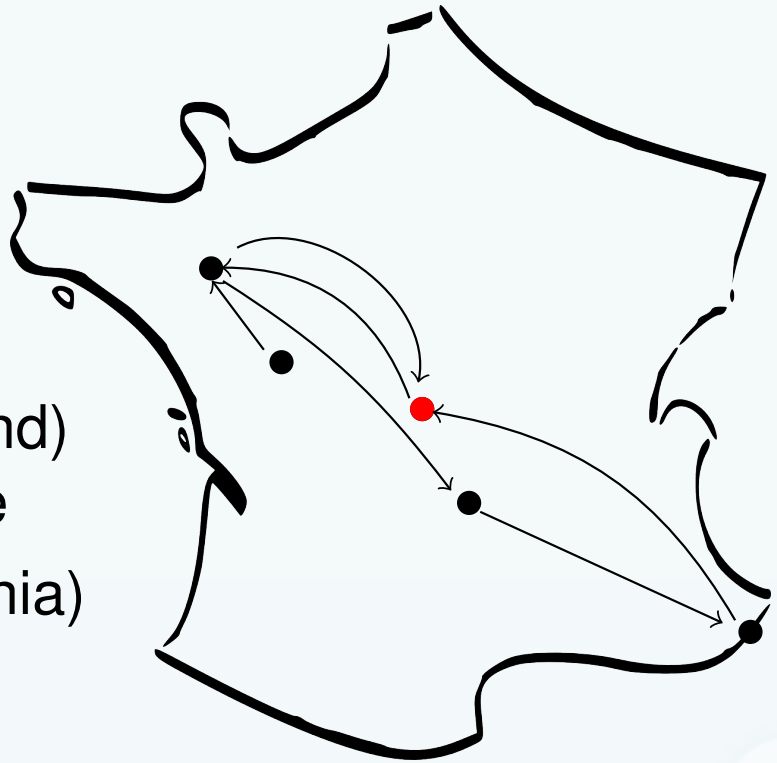
Parcours

- 84 Ordinateur en carton
- 92 Club informatique
- 96 BAC S (Segré - 49)
- 96-98 MPSI + MP (Rennes)
- 98-01 Ingénieur ISIMA (Clermont Ferrand) + DEA Optimisation / Productique
- 01-04 Doctorat Univ. de Nice (Inria Sophia)
- 04-05 ATER (Polytech'Nice)
- 05-13 MCF (INSA CVL)
- 13-15 Délég. Inria (CentraleSupélec Rennes)
- 15-16 MCF (INSA CVL)



Parcours

- 84 Ordinateur en carton
- 92 Club informatique
- 96 BAC S (Segré - 49)
- 96-98 MPSI + MP (Rennes)
- 98-01 Ingénieur ISIMA (Clermont Ferrand) + DEA Optimisation / Productique
- 01-04 Doctorat Univ. de Nice (Inria Sophia)
- 04-05 ATER (Polytech'Nice)
- 05-13 MCF (INSA CVL)
- 13-15 Délég. Inria (CentraleSupélec Rennes)
- 15-16 MCF (INSA CVL)

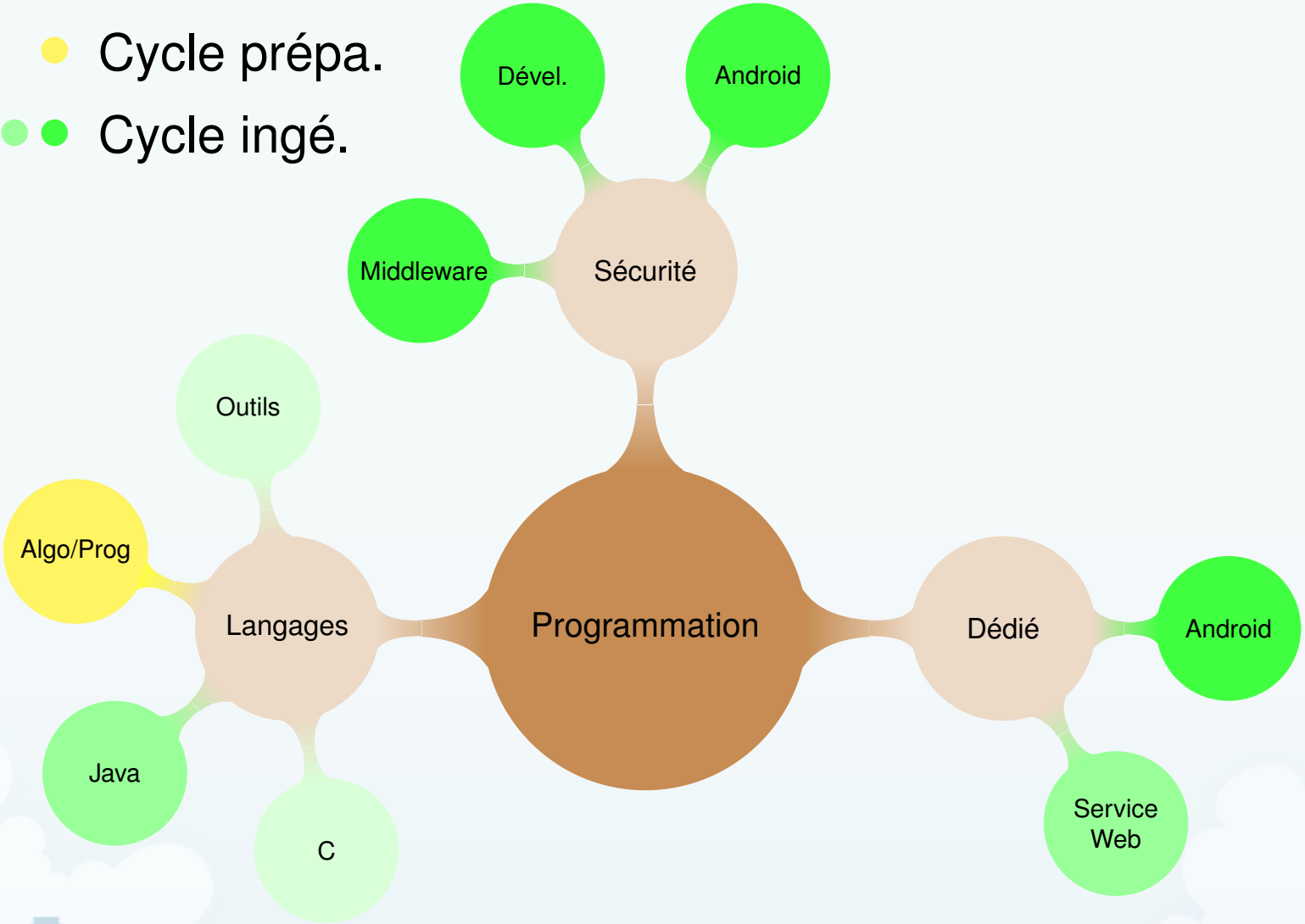


Plan

1. Parcours et formation
2. Enseignement
3. Recherche en sécurité
4. Flot de contrôle
5. Vie privée
6. Conclusion

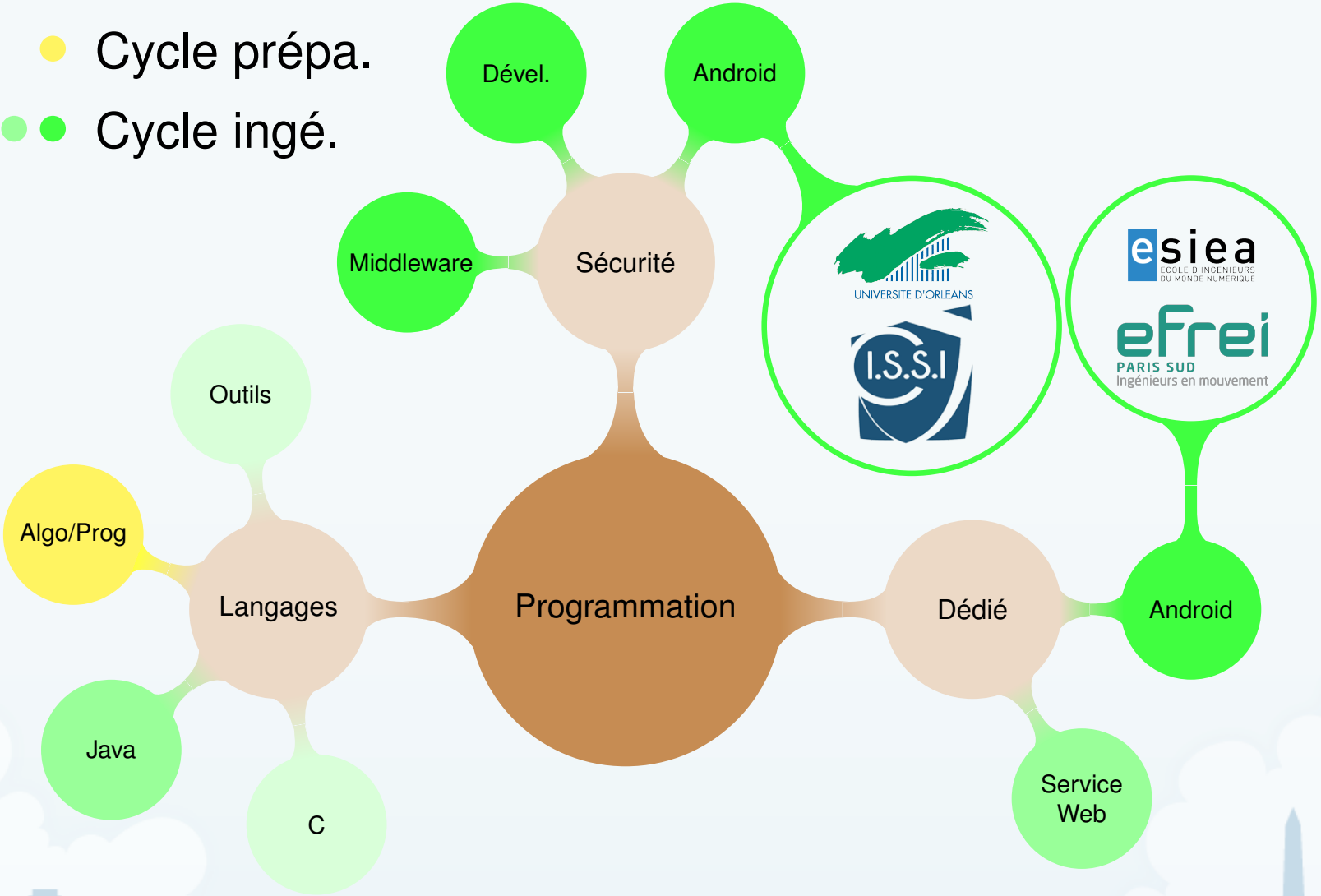
Enseignement

- Cycle prépa.
- Cycle ingé.

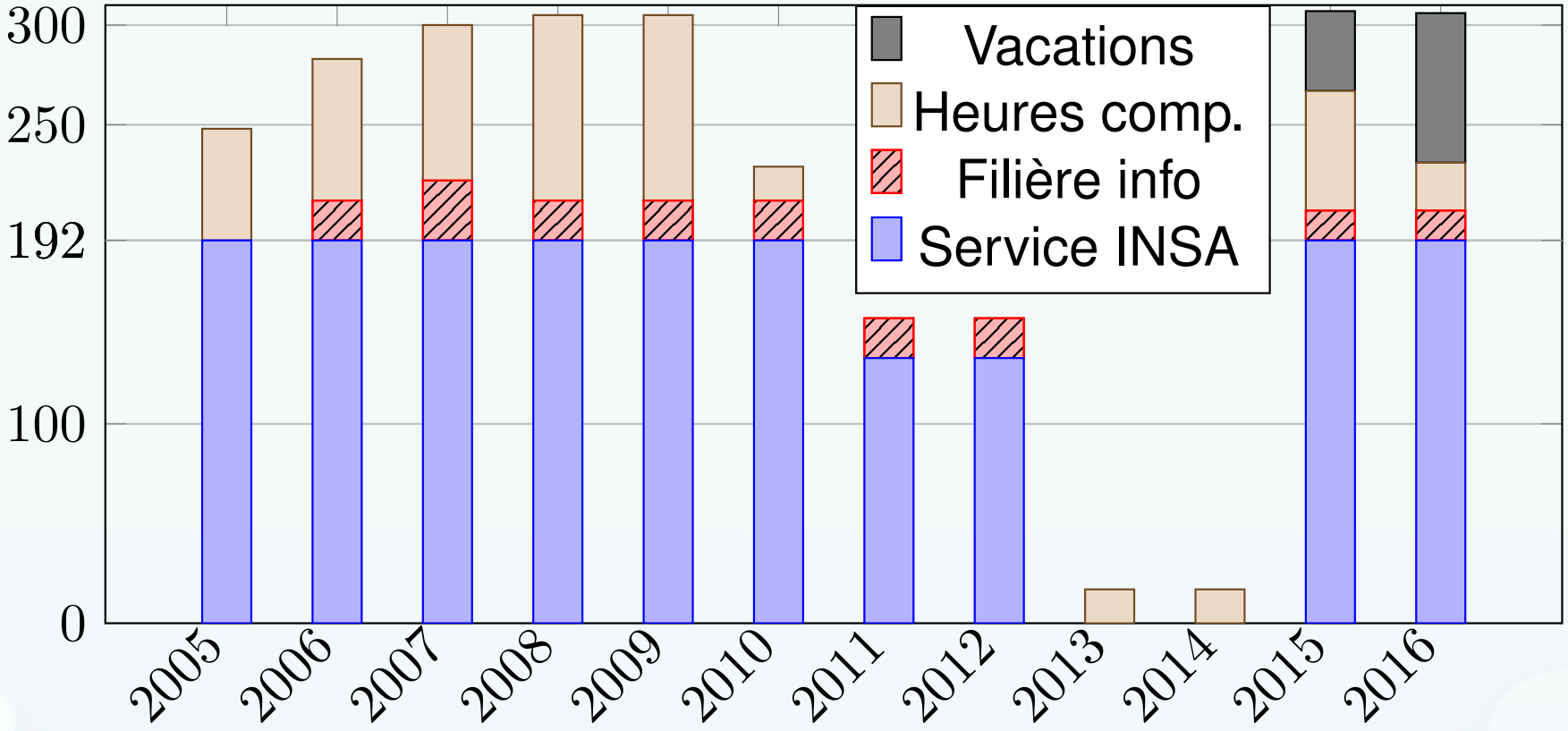


Enseignement

- Cycle prépa.
- Cycle ingé.



Enseignement : volumétrie



+ Administrateur de la plate-forme d'ENT pédagogique de l'INSA Centre Val de Loire (2005-2013) : Claroline

Formation par la recherche

Jérémy Briffaut (20% - 2008 - C. Toinard)

Formalisation et garantie de propriétés de sécurité système : application à la détection d'intrusions

Xavier Kauffmann-Tourkestansky (50% - 2012 - P. Berthomé)

Analyses sécuritaires de code de carte à puce sous attaques physiques simulées

Ghada Arfaoui (80% - 2015 - P. Berthomé)

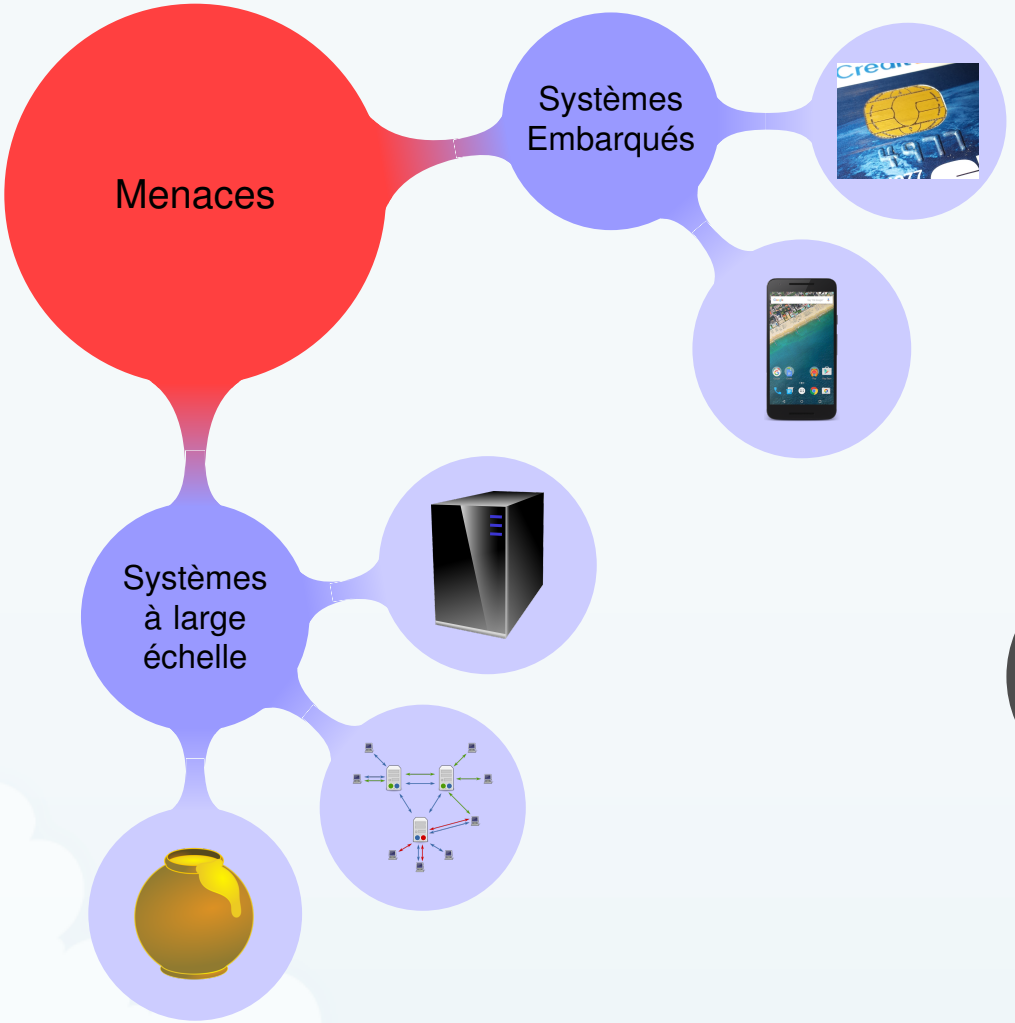
Protocoles cryptographiques préservant la vie privée pour les services mobiles sans contact

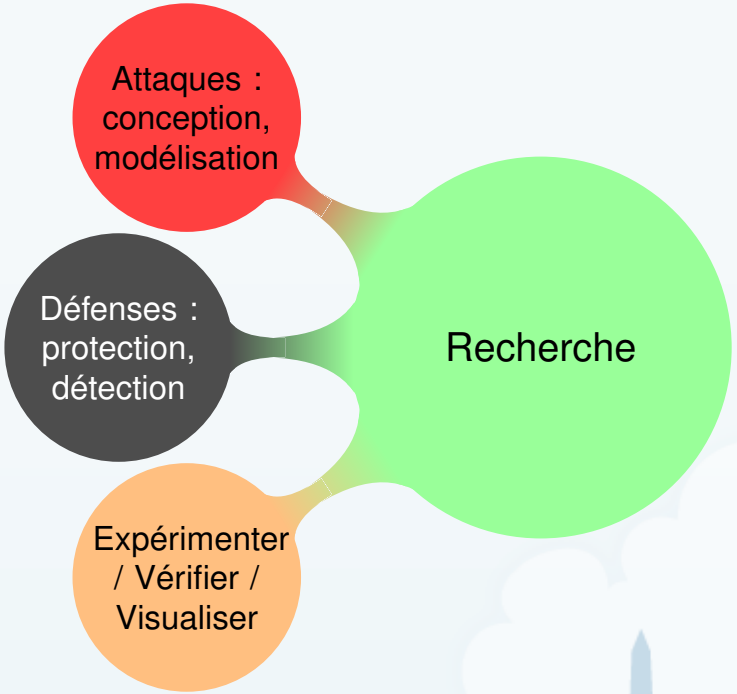
Mourad Leslous (V. Viet Triem Tong)

Sécurité Android

Sécurité : des menaces à la recherche

Sécurité : des menaces à la recherche





Systèmes à large échelle

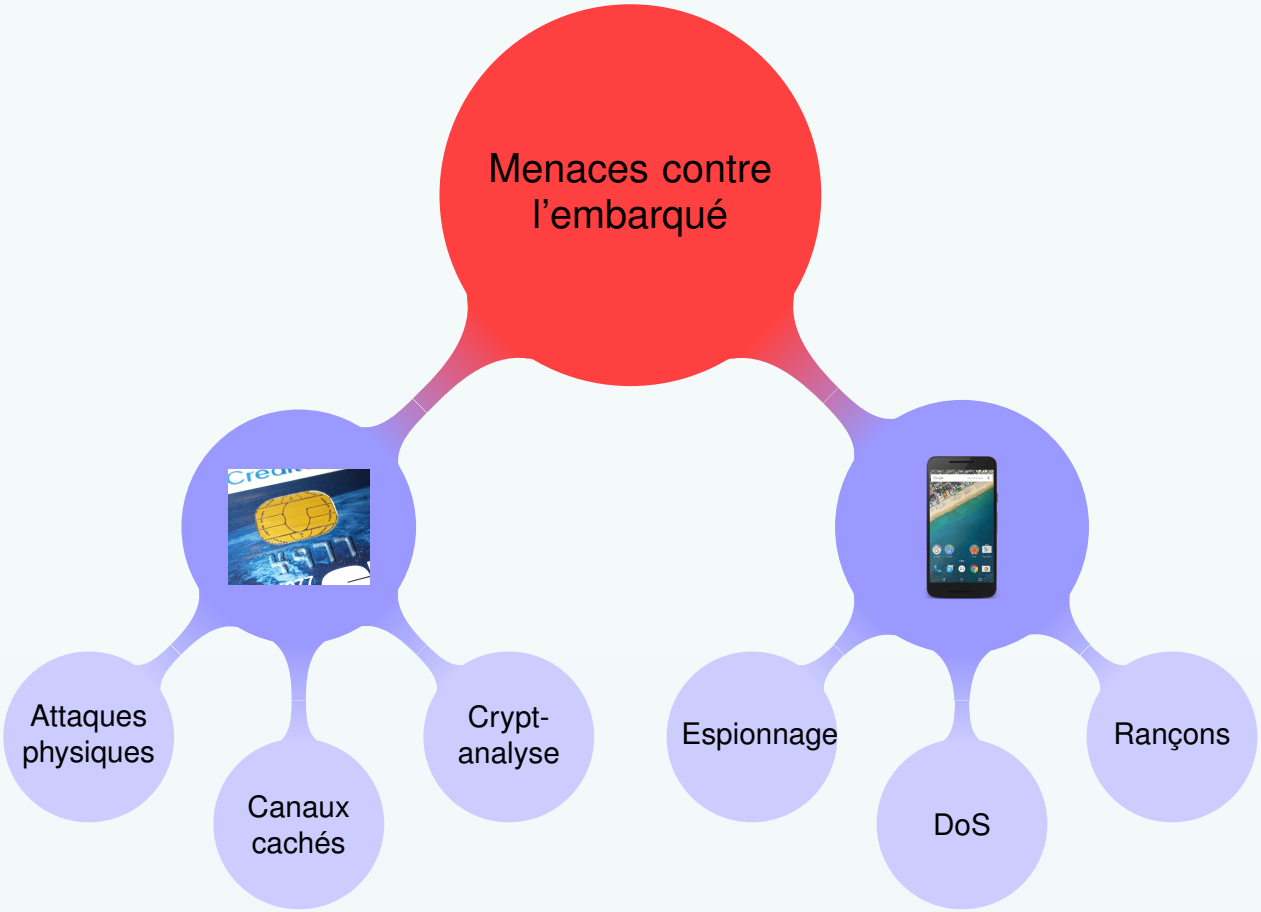
Menaces

- Mauvaise gestion de l'accès à l'information
- Escalade de privilèges (vulnérabilité)
- Mauvaise configuration de politiques

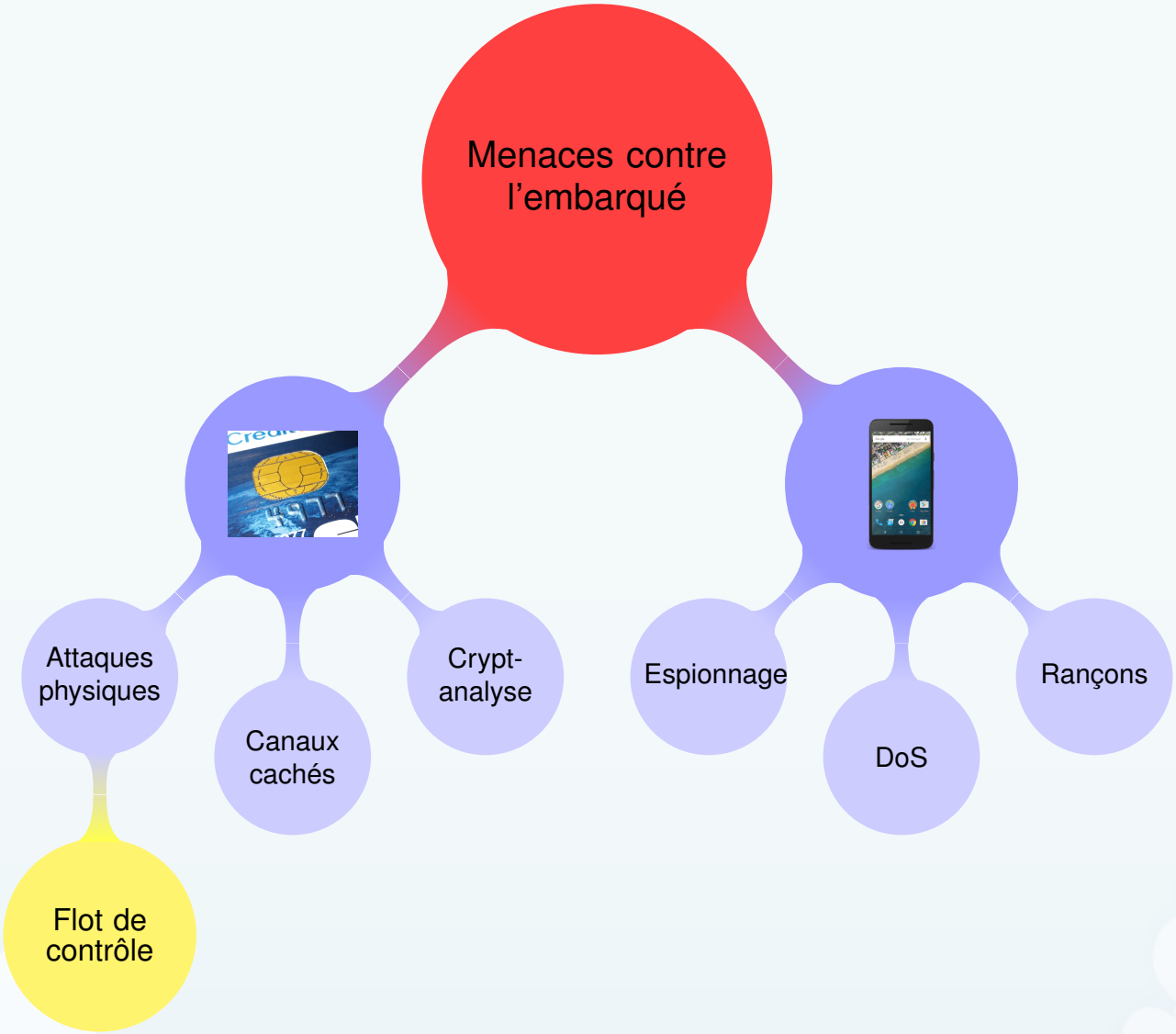
Contributions : politique et contrôle d'accès

- Capturer et analyser des attaques
- Exprimer le contrôle d'accès [FGCS'13&14]
- Administrer les politiques [IJAS'09]

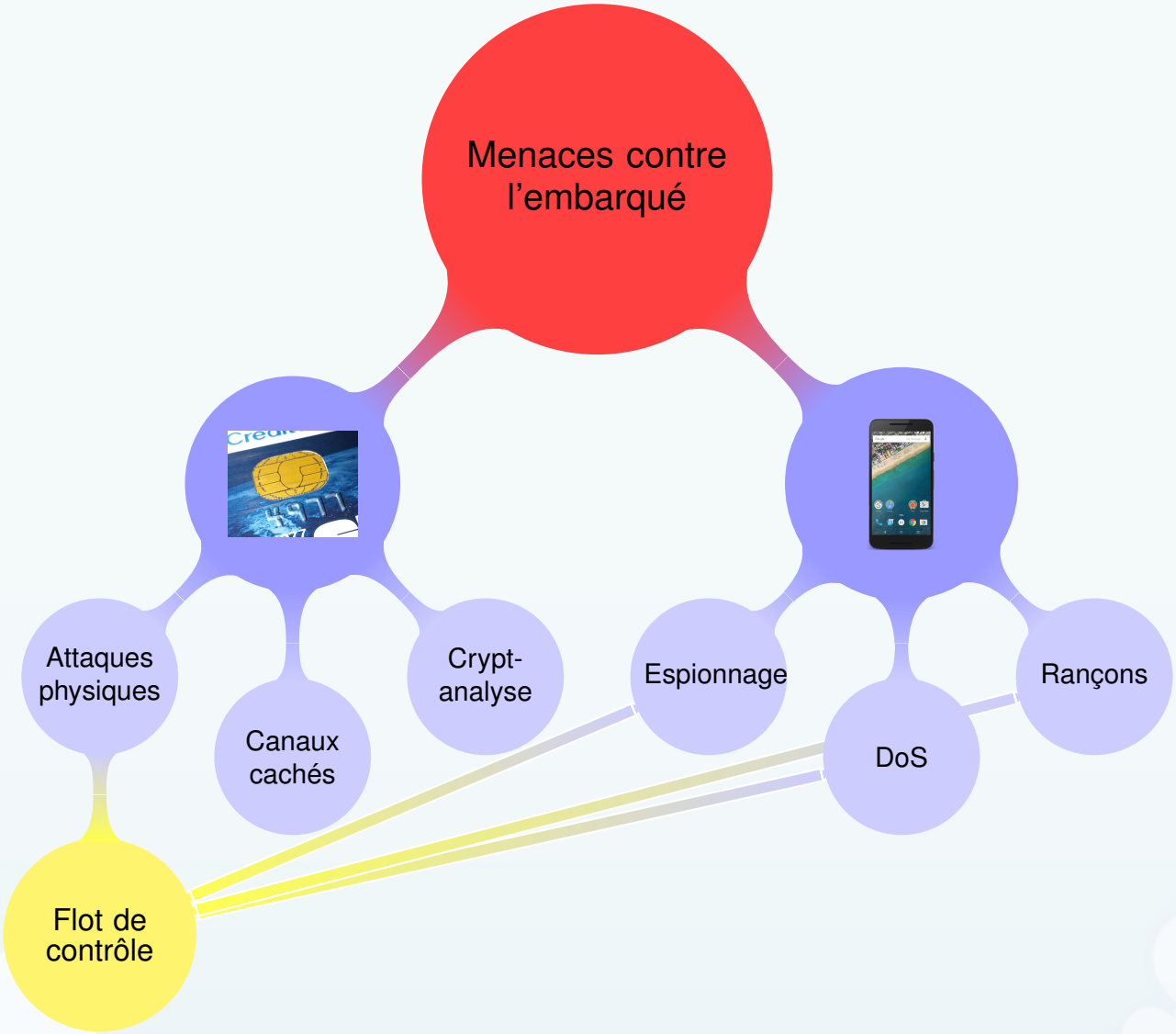
Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons

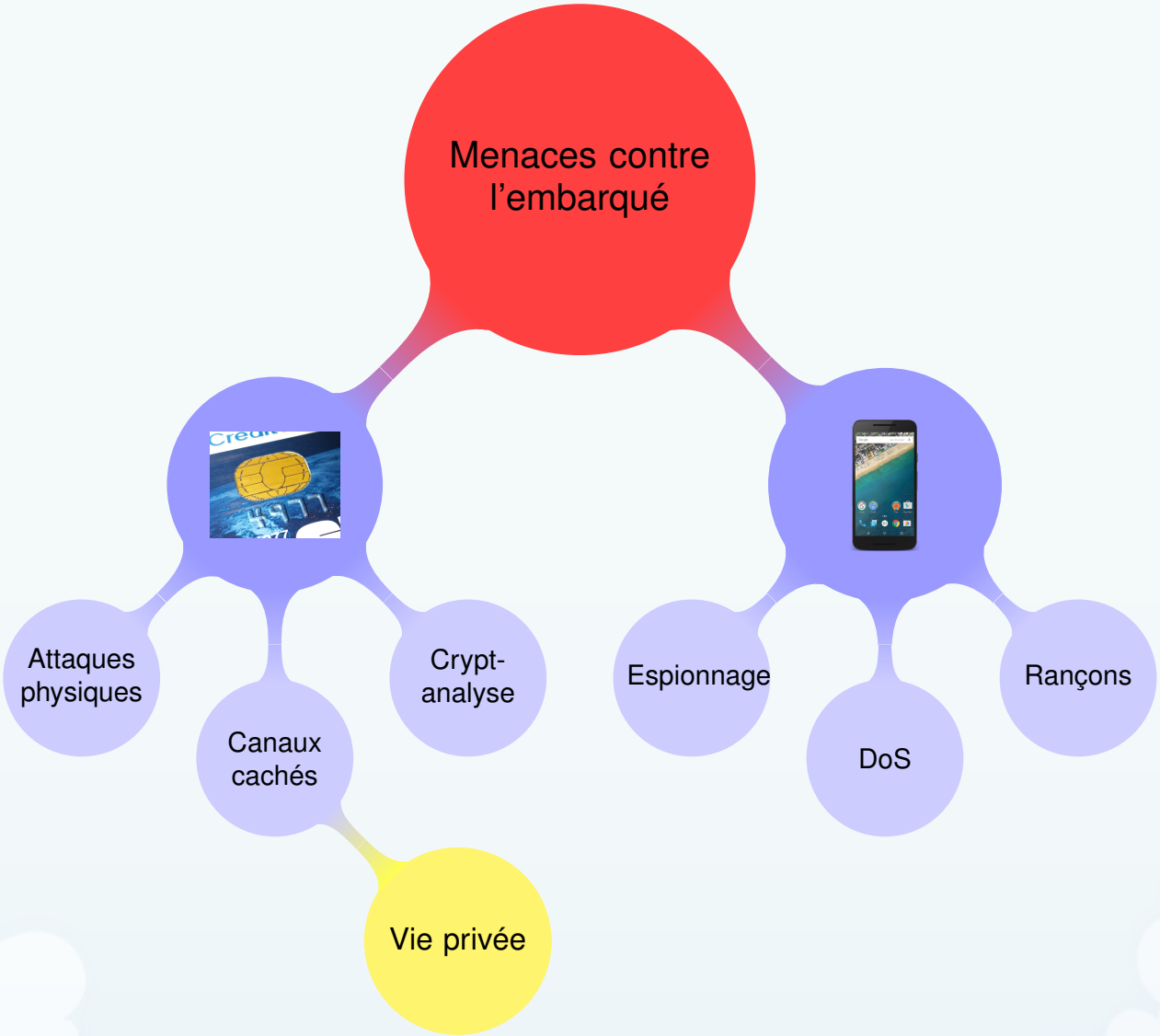
Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Flot de contrôle

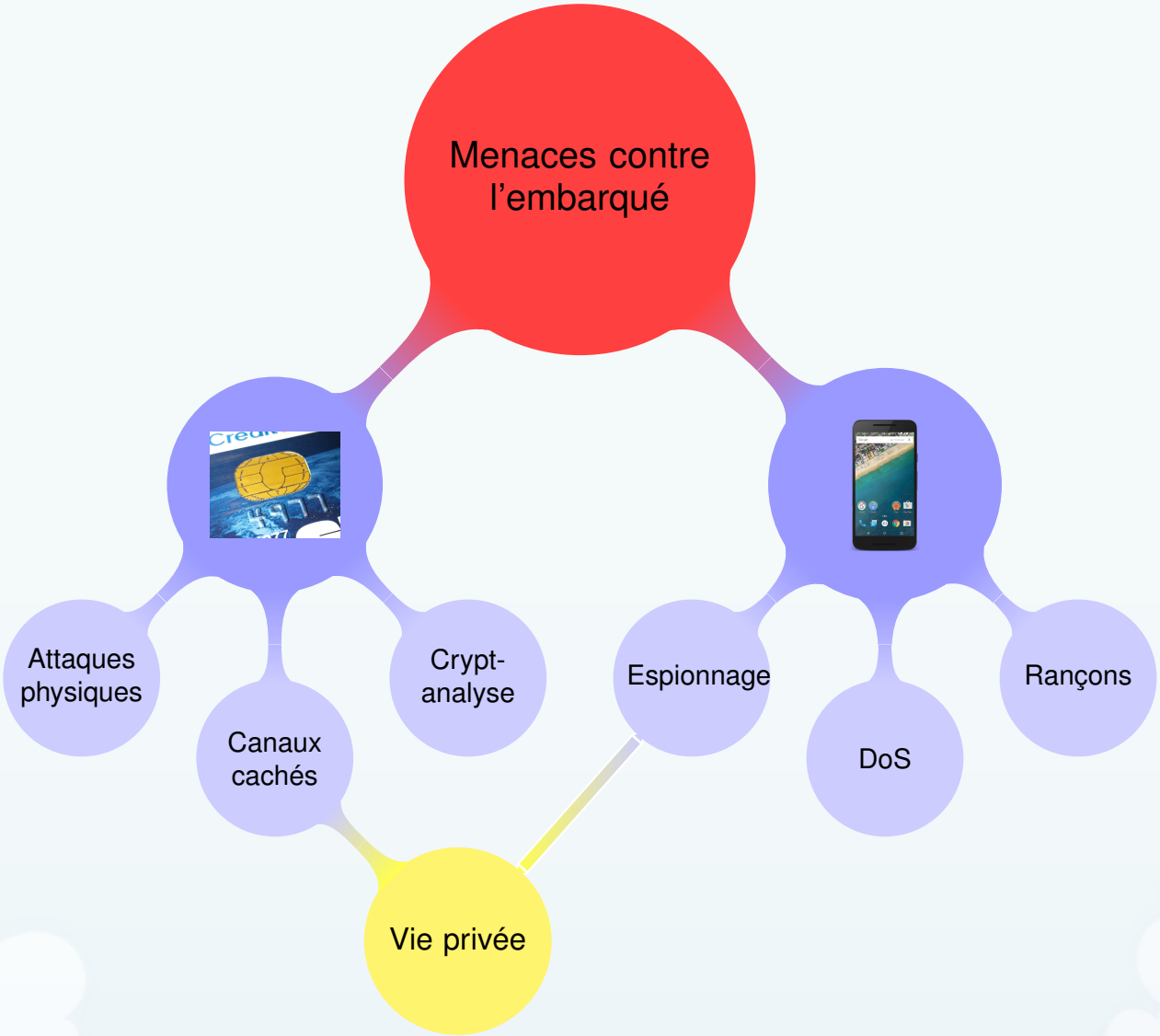
Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Flot de contrôle

Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Vie privée
Espionnage
DoS
Rançons

Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Vie privée

Plan

1. Parcours et formation
2. Enseignement
3. Recherche en sécurité
4. Flot de contrôle
5. Vie privée
6. Conclusion

Le flot de contrôle

Contexte carte à puces

- Garantir l'intégrité du code exécuté
- Effets des attaques physiques sur le flot de contrôle ?
- Contremesures pour sécuriser ce flot ?

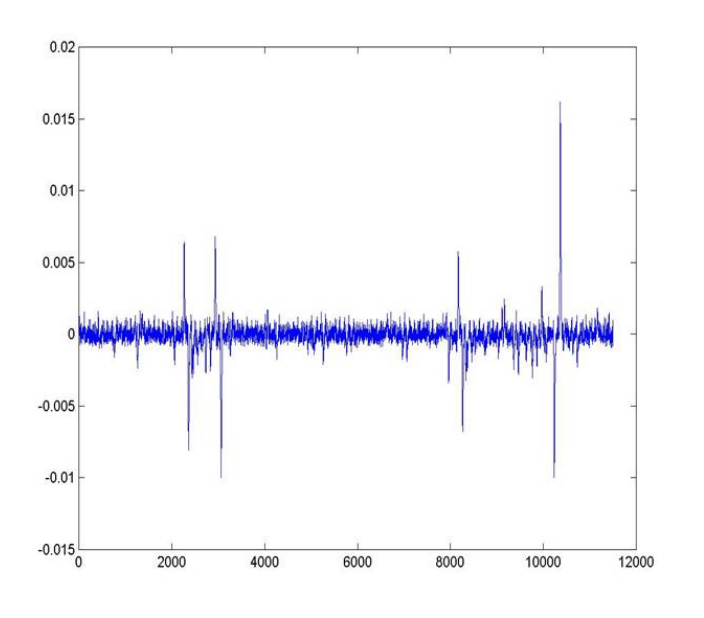
Contexte *malware* Android

- Exécuter et observer des *malware* à large échelle
- Peut-on aider l'exécution de *malware* ?
- Peut-on agir sur leur flot de contrôle ?

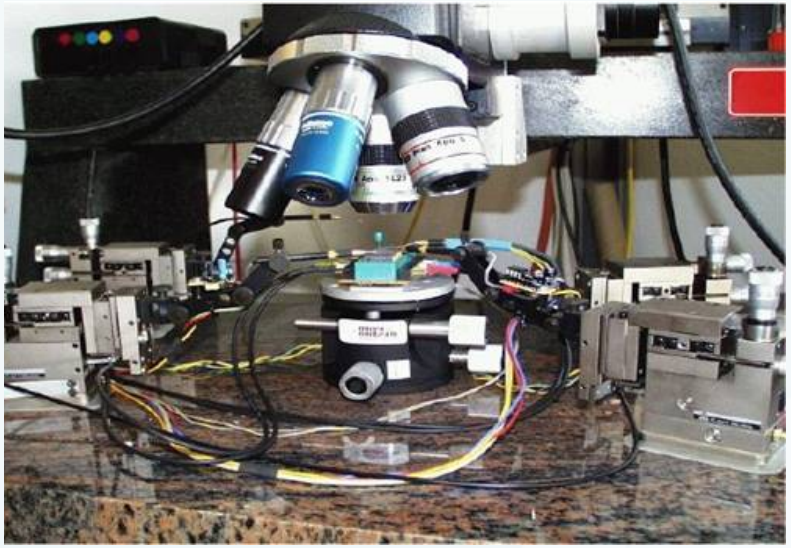
Attaques physiques contre les cartes à puce

Attaques physiques :

- Moyens : tir laser, perturbation de l'horloge, ...
- But : modifier l'exécution d'un programme



Observer



Attaquer

Sécurité et attaques physiques

- Codes très particuliers (authentification, chiffrement, . . .)
- Analyse et modification du code au niveau *source*

Démarche

1. Modéliser les attaques
2. Evaluer la robustesse du code
3. Ajouter des contremesures au niveau source

Thèse CIFRE : Xavier Kauffmann-Tourkestansky

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES CENTRE VAL DE LOIRE — UNIVERSITE D'ORLEANS — LIFO

Les modèles d'attaque

A **bas niveau** :

- un bit flip
- écrasement d'un bit/octet (valeur contrôlée)
- écrasement d'un bit/octet (valeur aléatoire)

Au **niveau du source** :

- Perturbent des variables
- Modifient le flot de contrôle :
  - Changement de direction d'un branchement
  - Exécution de NOPs
  - Exécution d'un JMP

Exemple d'attaque du flot de contrôle

```
1  uint user_tries = 0; // initialization of the number of tries for this session
2  uint max_tries = 3; // max number of tries
3  while (...) /* card life cycle: */
4  {
5     incr_tries(user_tries);
6     res = get_pin_from_terminal(); // receives 1234
7     pin = read_secret_pin(); // read real pin: 0000
8     if (compare(res, pin))
9        { dec_tries(user_tries);
10         do_stuff() ;  }
11    if (user_tries >= max_tries)
12       {  killcard() ;  }
13 }
```

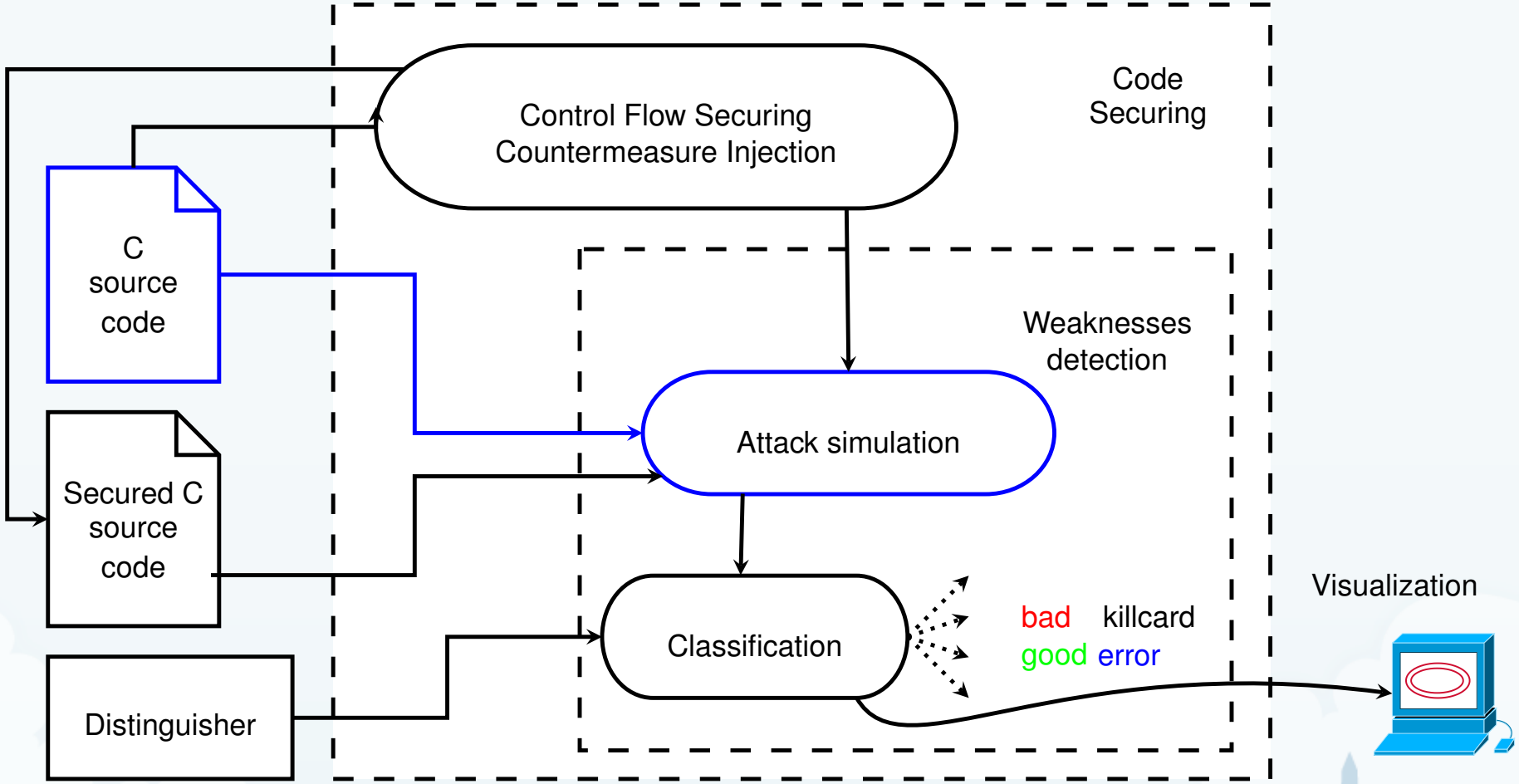
Listing 1 – Authentification avec un code pin

Exemple d'attaque du flot de contrôle

```
1  uint user_tries = 0; // initialization of the number of tries for this session
2  uint max_tries = 3; // max number of tries
3  while (...) /* card life cycle: */
4  {
5     incr_tries(user_tries);
6     res = get_pin_from_terminal(); // receives 1234
7     pin = read_secret_pin(); // read real pin: 0000
8     if (compare(res, pin))  ⇒ NOP ... NOP
9        { dec_tries(user_tries);
10          do_stuff() ;  }
11    if (user_tries >= max_tries)
12       {  killcard() ;  }
13 }
```

Listing 2 – Authentification avec un code pin

Evaluer la robustesse du code



Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240
241     while (i--)
242     {
243         buf[i] ^= key[i];
244         cpk[i] = key[i];
245         cpk[16+i] = key[16 + i];
246     }
247     ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240      goto dest ;
241     while (i--)
242     {
243   dest :buf[i] ^= key[i];
244         cpk[i] = key[i];
245         cpk[16+i] = key[16 + i];
246     }
247     ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240      goto dest ;
241     while (i--)
242     {
243         buf[i] ^= key[i];
244   dest :cpk[i] = key[i];
245         cpk[16+i] = key[16 + i];
246     }
247     ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240      goto dest ;
241     while (i--)
242     {
243        buf[i] ^= key[i];
244        cpk[i] = key[i];
245   dest :cpk[16+i] = key[16 + i];
246     }
247     ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240      goto dest ;
241     while (i--)
242     {
243        buf[i] ^= key[i];
244        cpk[i] = key[i];
245        cpk[16+i] = key[16 + i];
246 dest :}
247     ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240      goto dest ;
241     while (i--)
242     {
243        buf[i] ^= key[i];
244        cpk[i] = key[i];
245        cpk[16+i] = key[16 + i];
246     }
247 dest :;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240      dest :
241     while (i--)
242     {
243        buf[i] ^= key[i];
244        cpk[i] = key[i];
245        cpk[16+i] = key[16 + i];
246     }
247     ;  goto dest ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240
241     while (i--)
242     {
243  dest :buf[i] ^= key[i];
244         cpk[i] = key[i];
245         cpk[16+i] = key[16 + i];
246     }
247     ; goto dest ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240
241     while (i--)
242     {
243         buf[i] ^= key[i];
244   dest :cpk[i] = key[i];
245         cpk[16+i] = key[16 + i];
246     }
247     ; goto dest ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

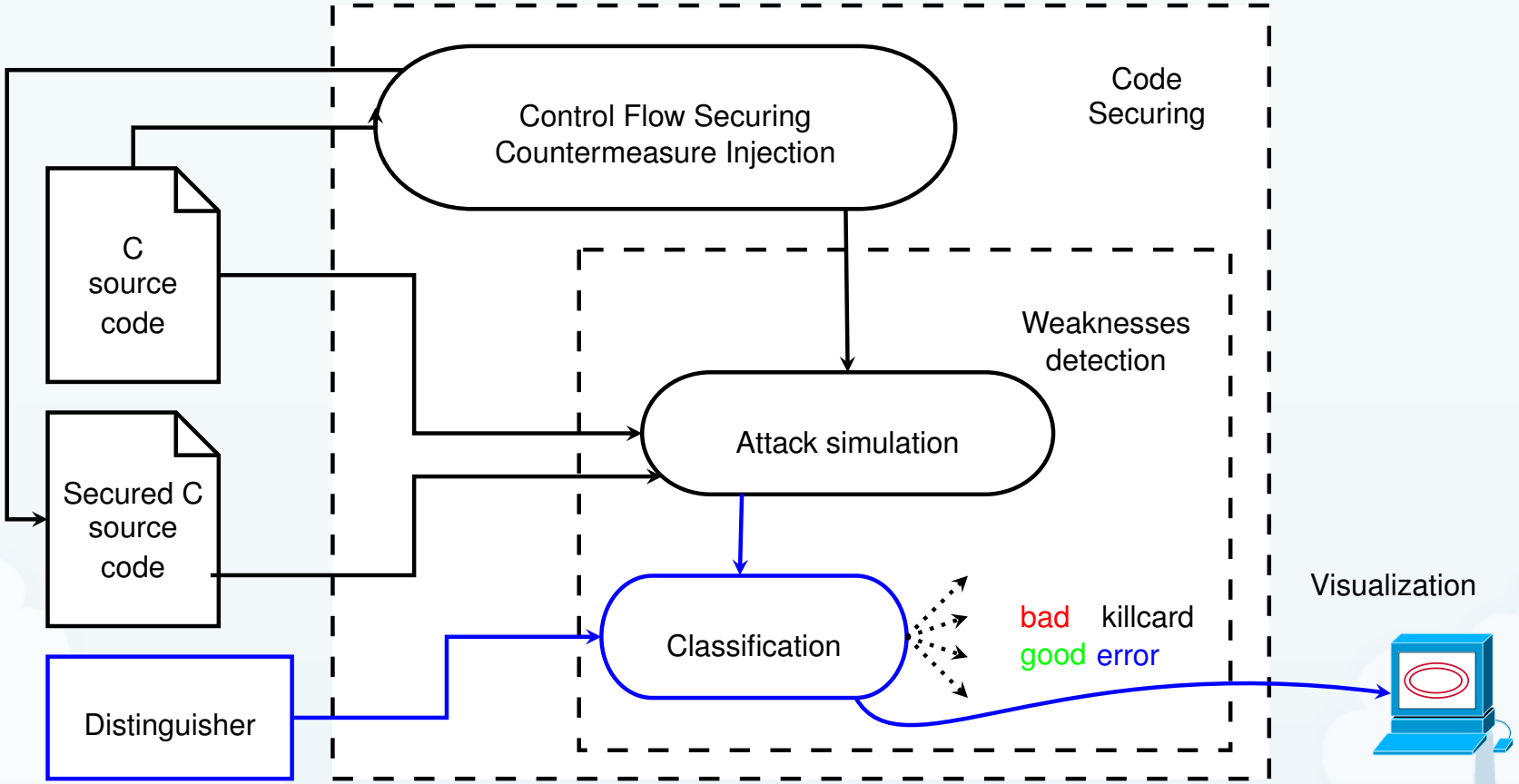
```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240
241     while (i--)
242     {
243         buf[i] ^= key[i];
244         cpk[i] = key[i];
245   dest :cpk[16+i] = key[16 + i];
246     }
247     ; goto dest ;
248 } /* aes_addRoundKey_cpy */
```

Simulation des attaques

```
237 void aes_addRoundKey_cpy(uint8_t *buf, uint8_t *key, uint8_t *cpk)
238 {
239     register uint8_t i = 16;
240
241     while (i--)
242     {
243         buf[i] ^= key[i];
244         cpk[i] = key[i];
245         cpk[16+i] = key[16 + i];
246 dest :}
247     ; goto dest ;
248 } /* aes_addRoundKey_cpy */
```

Distinction des attaques dangereuses

- définir un **scénario fonctionnel**
- savoir **distinguer** les exécutions



Classification des attaques

Scénario :
Chiffrement de données par AES [Levin 07], SHA et Blowfish [Guthaus et al. 01]

Classes distinctes :

- **bad** (Wrong Answer) :
  - **bad j>1** : ($jumpsize \geq 2$ lines) sortie erronée
  - **bad j=1** : ($jumpsize = 1$ line) sortie erronée
- **good** (Effect Less) : output intact
- **error** or **timeout** : erreur, crash, boucle infinie
- **killcard** : attaque détectée

Classification des attaques

Scénario :

Chiffrement de données par AES [Levin 07], SHA et Blowfish [Guthaus et al. 01]

Classes distinctes :

- **bad** (Wrong Answer) :
  - **bad j>1** : ($jumpsize \geq 2$ lines) sortie erronée
  - **bad j=1** : ($jumpsize = 1$ line) sortie erronée
- **good** (Effect Less) : output intact
- **error** or **timeout** : erreur, crash, boucle infinie
- **killcard** : attaque détectée

Visualisation des faiblesses

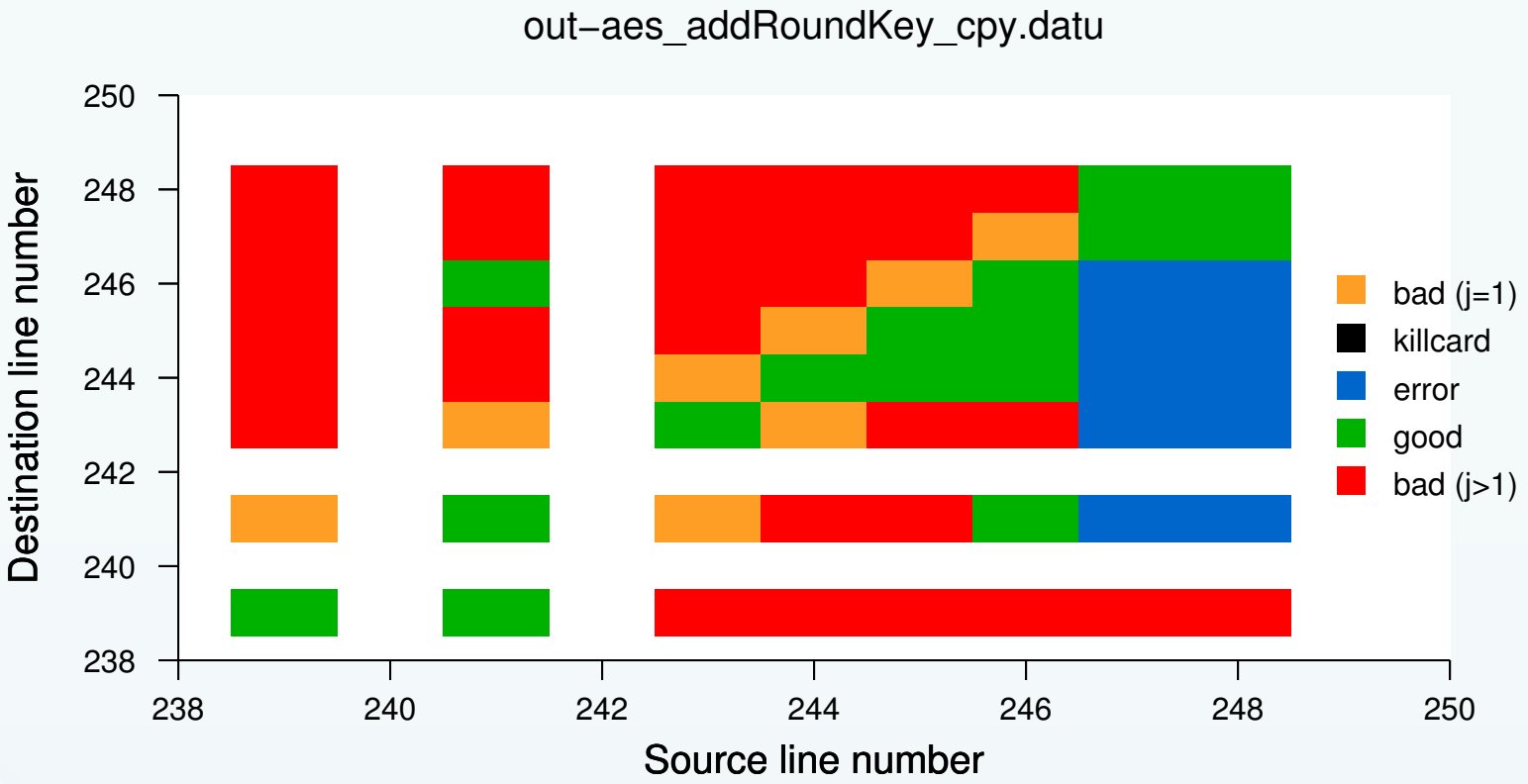
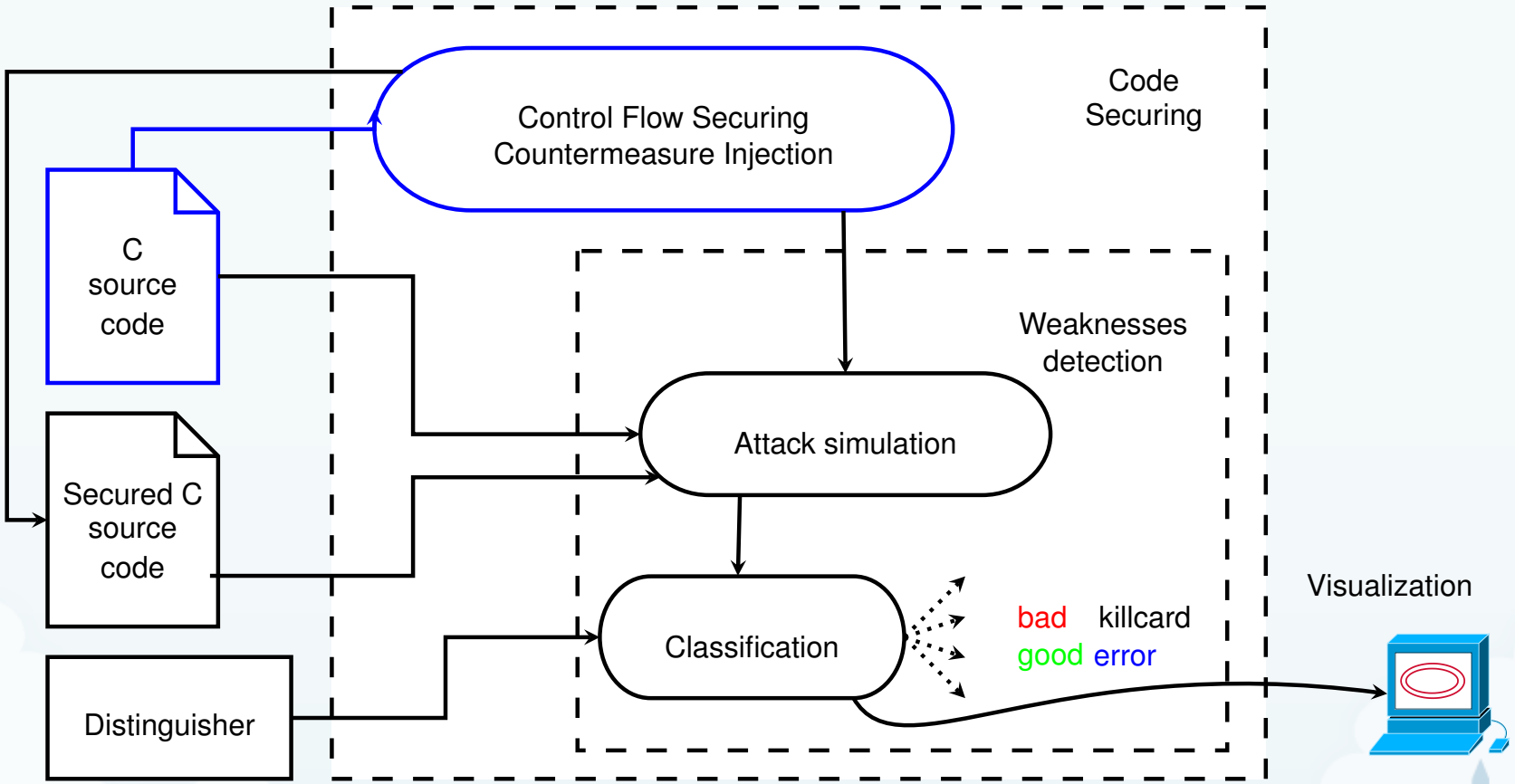


FIGURE – Visualisation des faiblesses de aes_addRoundKey_cpy

Sécurisation du flot de contrôle



Protection du flot de contrôle

Solutions existantes pour le **C***ontrol* **F***low* **I***ntegrity*

- Codes assembleurs modifiés [Abadi et al. 05]
- JVM modifiées [Iguchi-Cartigny et al. 11, Lackner et al. 13]
- Signature de *basic block* [Oh et al. 02, Nicolescu et al. 03]

Contraintes

- Garder le code assembleur intact
- Pas de modification de l'OS
- Détecter les attaques internes à un *basic block*

Protection du flot de contrôle

Solutions existantes pour le **C***ontrol* **F***low* **I***ntegrity*

- Codes assembleurs modifiés [Abadi et al. 05]
- JVM modifiées [Iguchi-Cartigny et al. 11, Lackner et al. 13]
- Signature de *basic block* [Oh et al. 02, Nicolescu et al. 03]

Contraintes

- Garder le code assembleur intact
- Pas de modification de l'OS
- Détecter les attaques internes à un *basic block*

Protection du flot de contrôle

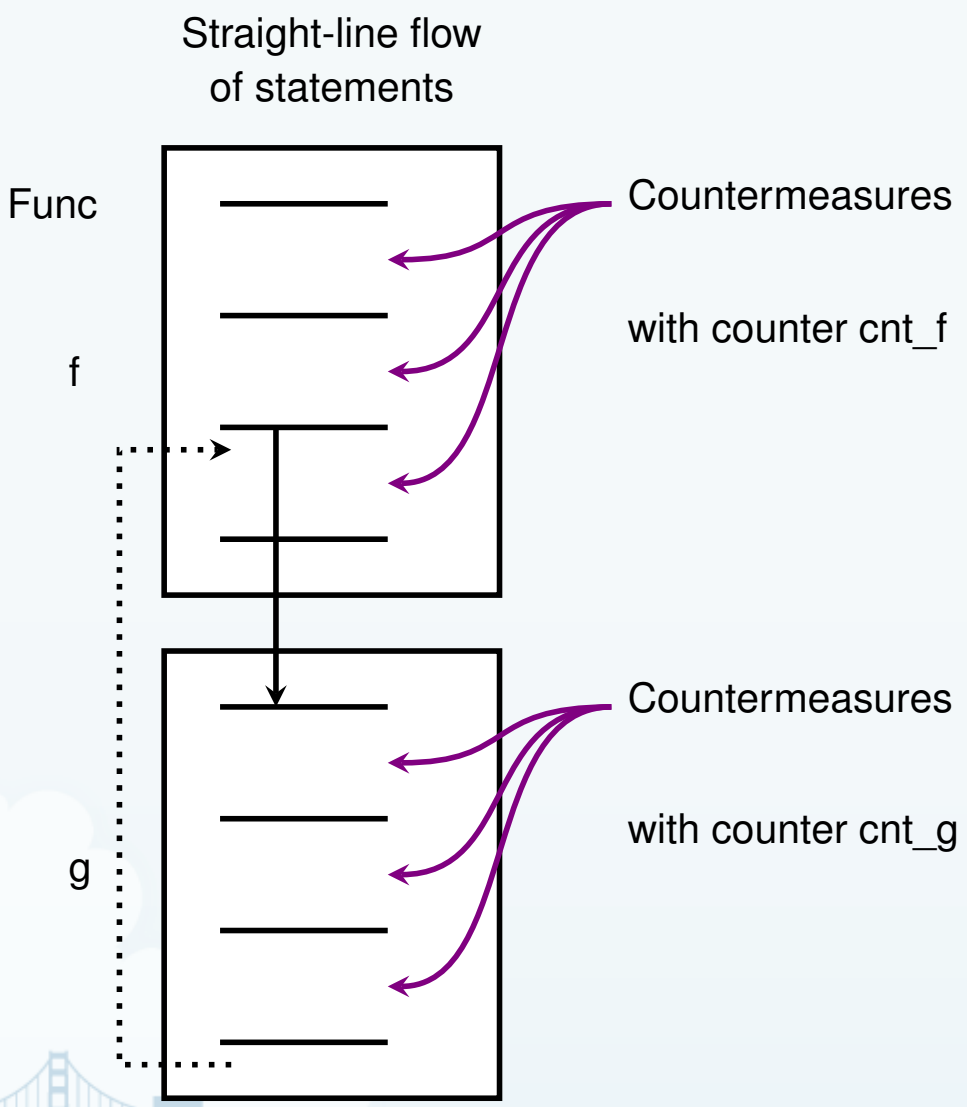
Solutions existantes pour le **C***ontrol* **F***low* **I***ntegrity*

- Codes assembleurs modifiés [Abadi et al. 05]
- JVM modifiées [Iguchi-Cartigny et al. 11, Lackner et al. 13]
- Signature de *basic block* [Oh et al. 02, Nicolescu et al. 03]

Contraintes

- Garder le code assembleur intact
- Pas de modification de l'OS
- Détecter les attaques internes à un *basic block*

Principe de sécurisation



Check

cnt = (cnt == val+N ?
cnt +1 : killcard()) ;

Clefs du *design*

- Entrelacement
- Conditions
- Vérification

Constructions

- pour le if-then-else
- pour les boucles

Résultats

Attaques par sauts de j lignes

- Sans contremesures
- Avec contremesures (+CM)

	bad j > 1	bad j = 1	good	killcard	error	total
C JUMP ATTACKS	**Attaques à tous les rounds** *transients*					
AES	29%	4.2%	65%		0.4%	26370
AES + CM	**0%**	0.2%	5.3%	94%	0.0%	337516
SHA	75%	3.5%	19%		1.0%	43274
SHA + CM	**0%**	0.3%	1.2%	98%	0.1%	427690
Blowfish	32%	1.7%	62%		2.7%	213721
Blowfish + CM	**0%**	0.2%	23%	75%	0.4%	1400355

Visualisation des faiblesses

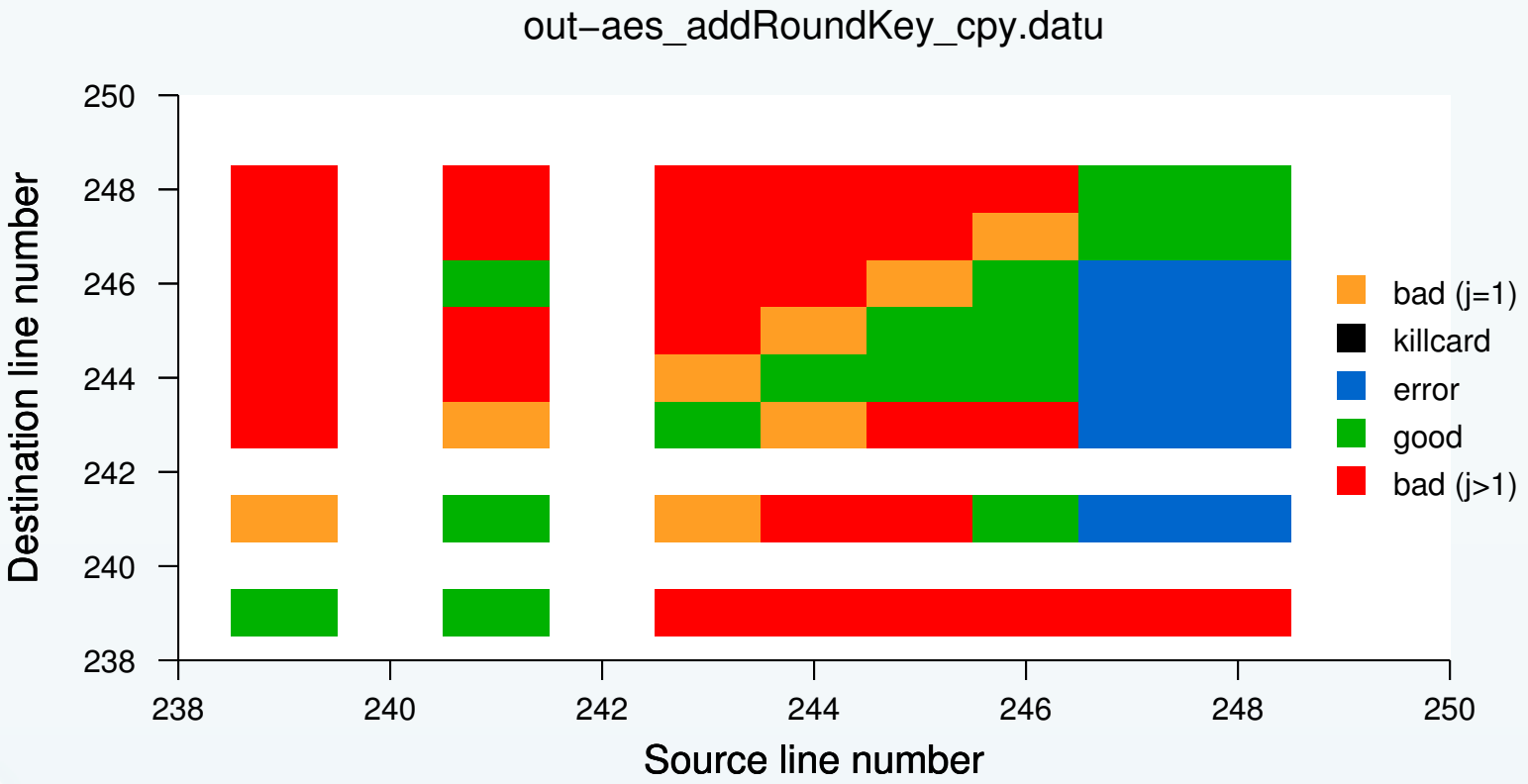


FIGURE – Visualisation des faiblesses de aes_addRoundKey_cpy

Visualisation des faiblesses

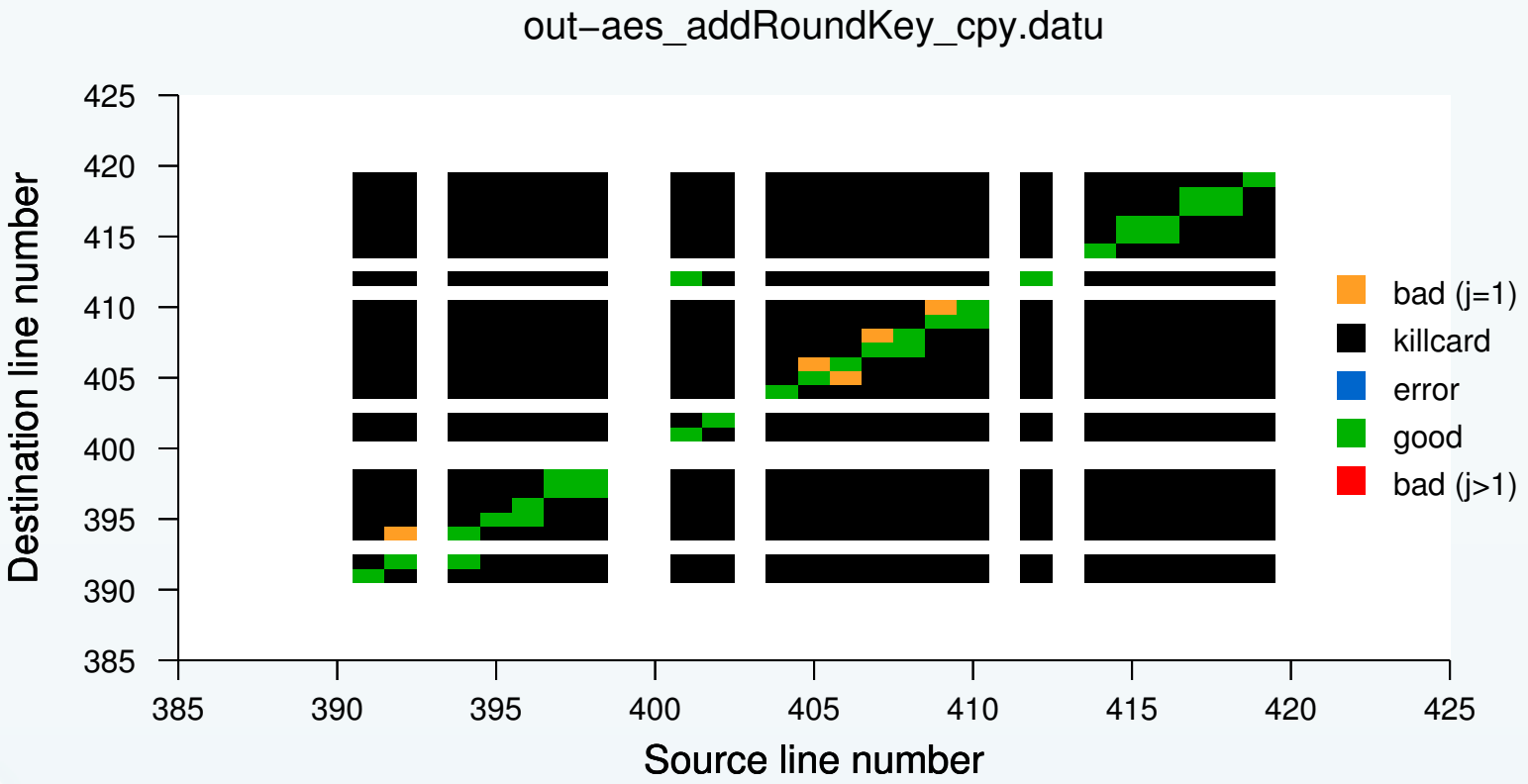
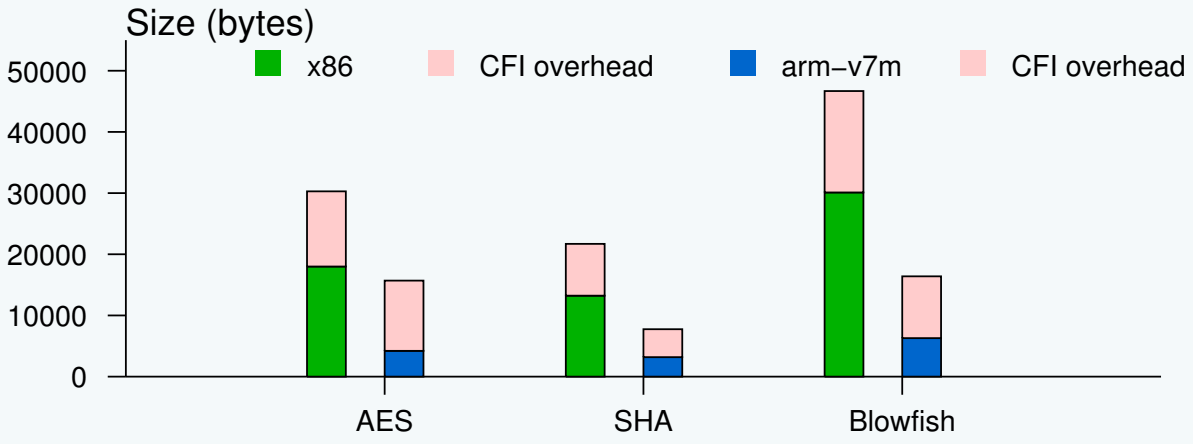
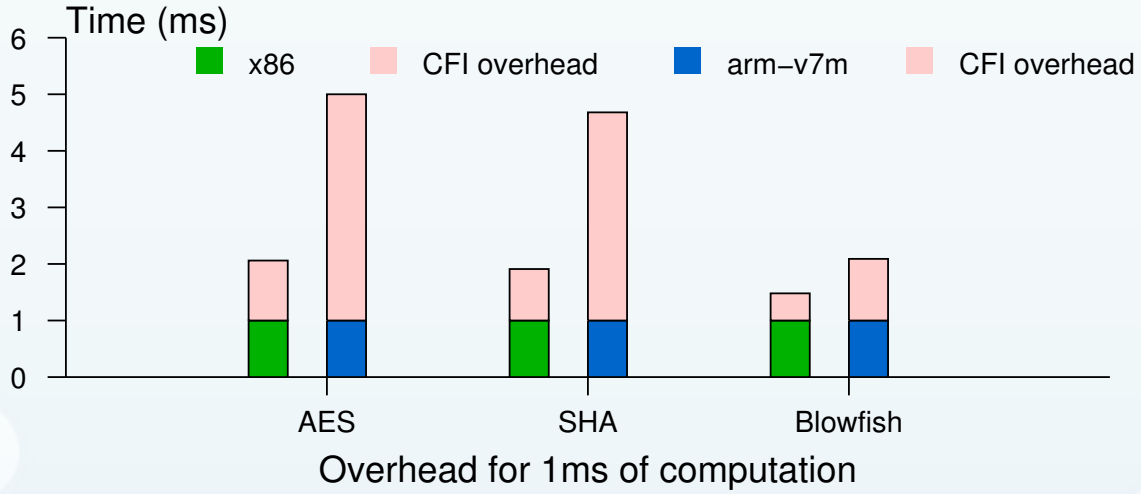


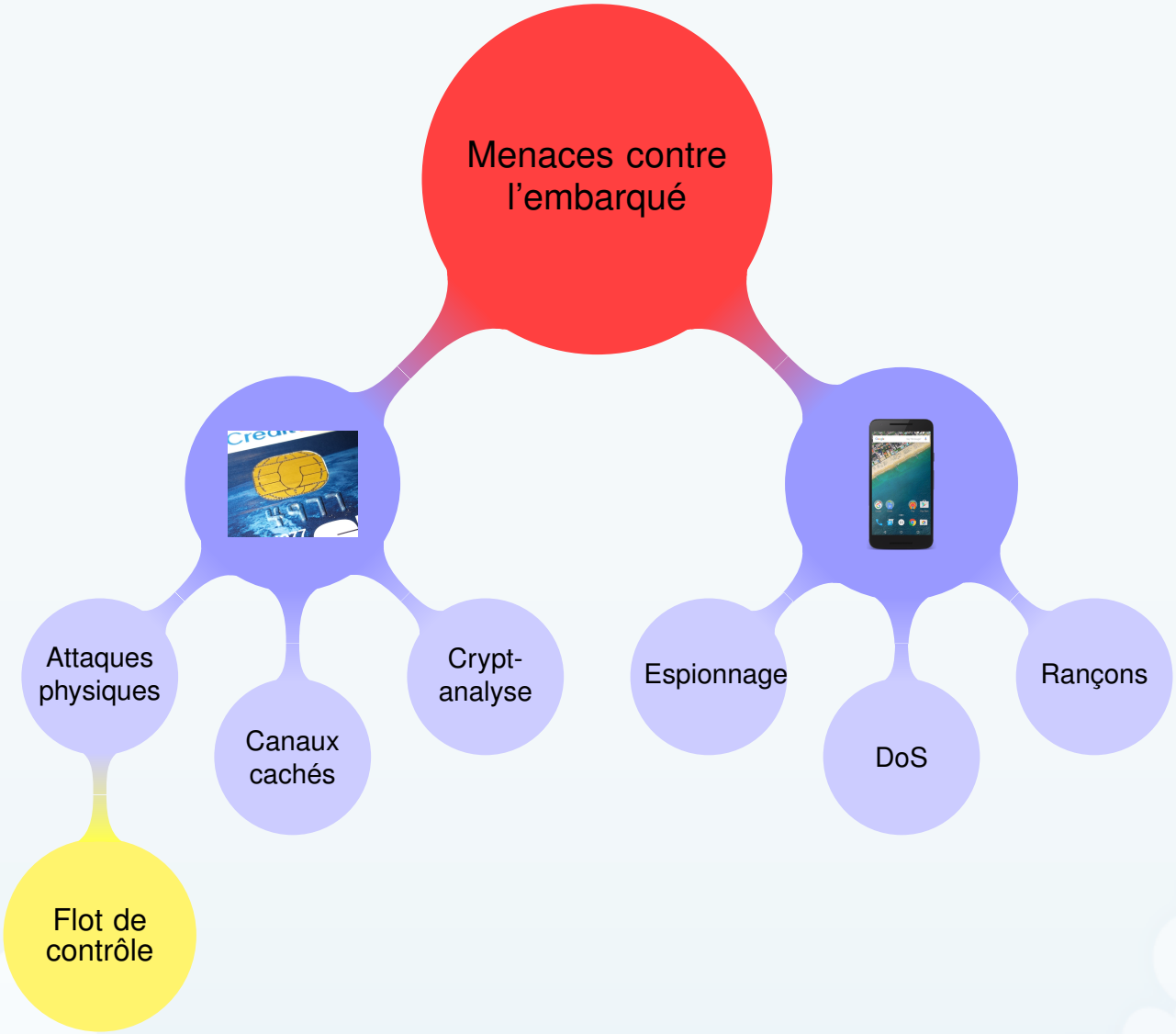
FIGURE – Faiblesses de aes_addRoundKey_cpy + CM (sécurisé)

Overheads - x86 et arm-v7m

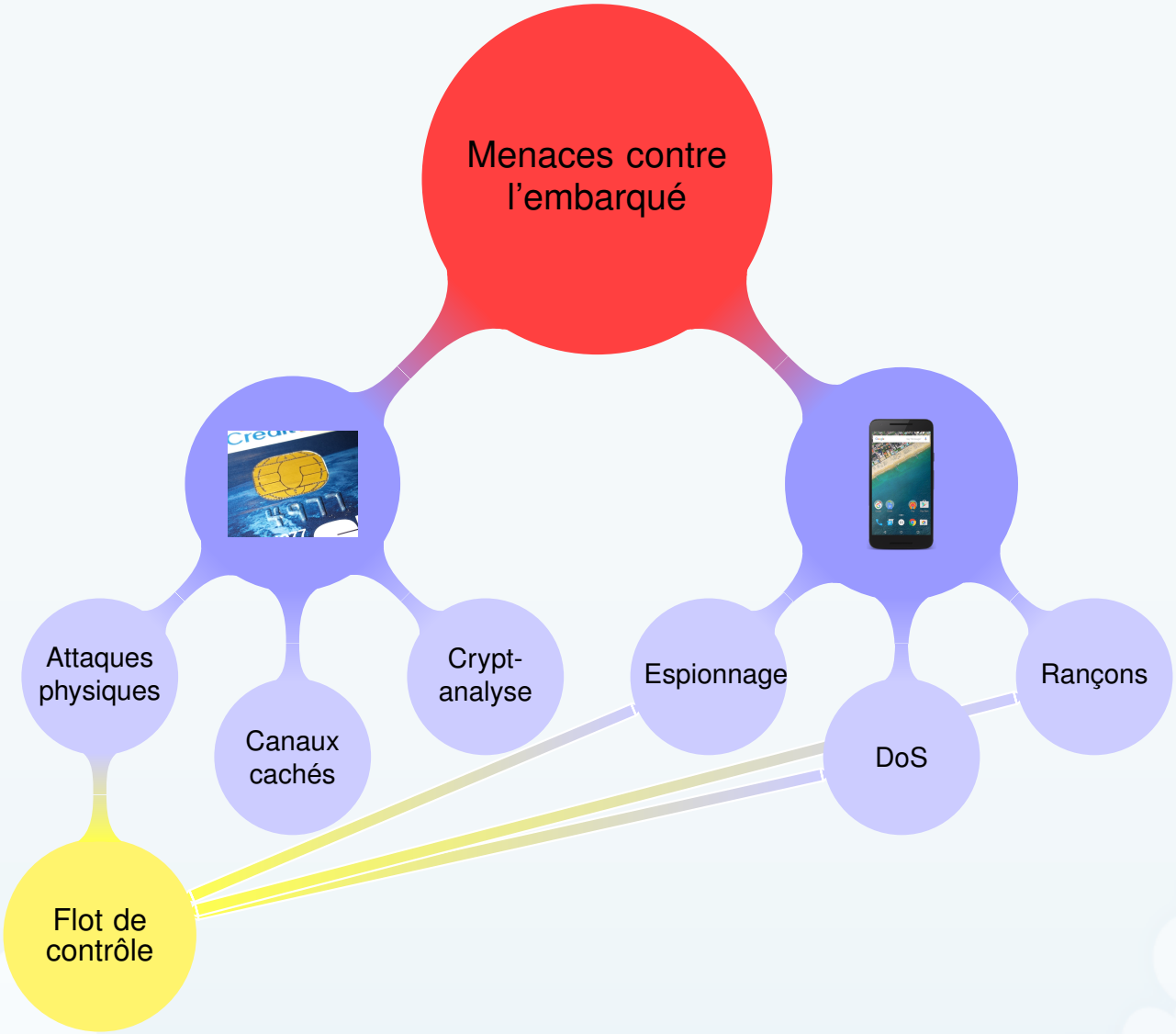




Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Flot de contrôle

Embarqué

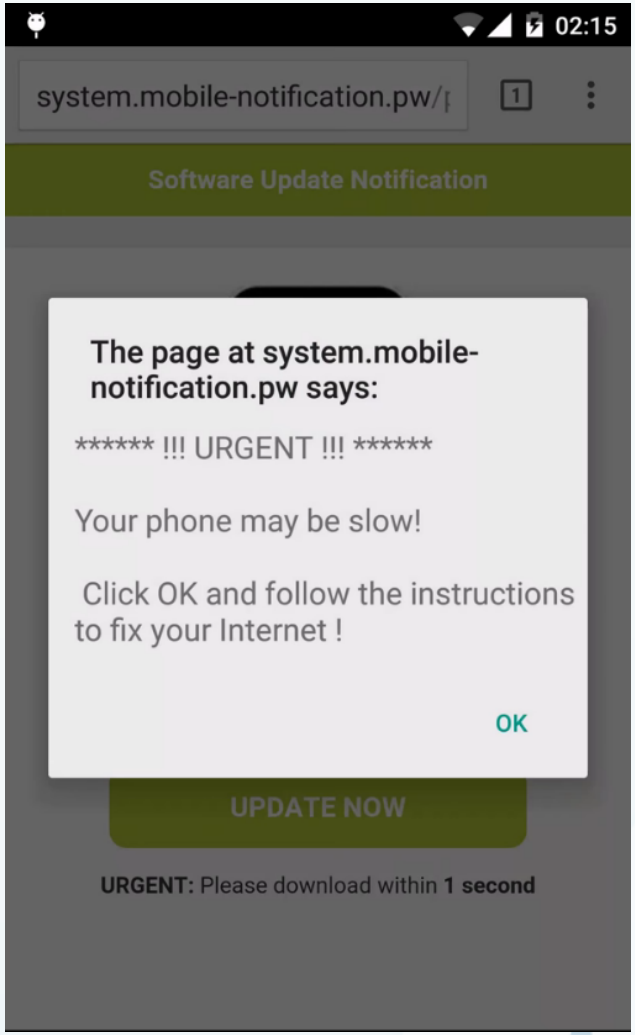

Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Flot de contrôle

Observation des *malware*

MobiDash : *aggressive adware*

- Publicités intempestives
- Démarre après 24h

Comment
observer son comportement ?



Aider l'exécution de *malware*

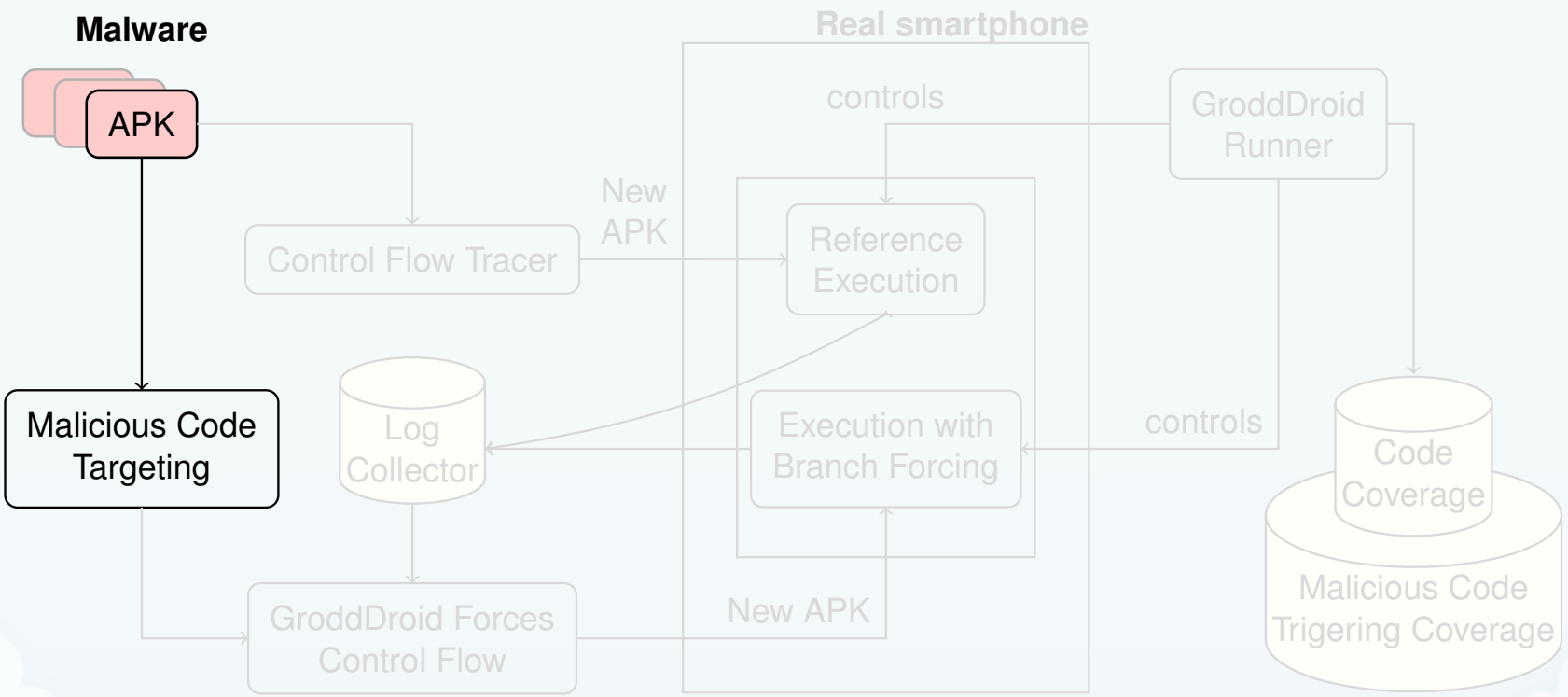
Des *malware* intelligents [LASER 16] :

- *time bomb*
- commandes distantes
- détection d'émulateurs
- obfuscation, chiffrement, ...

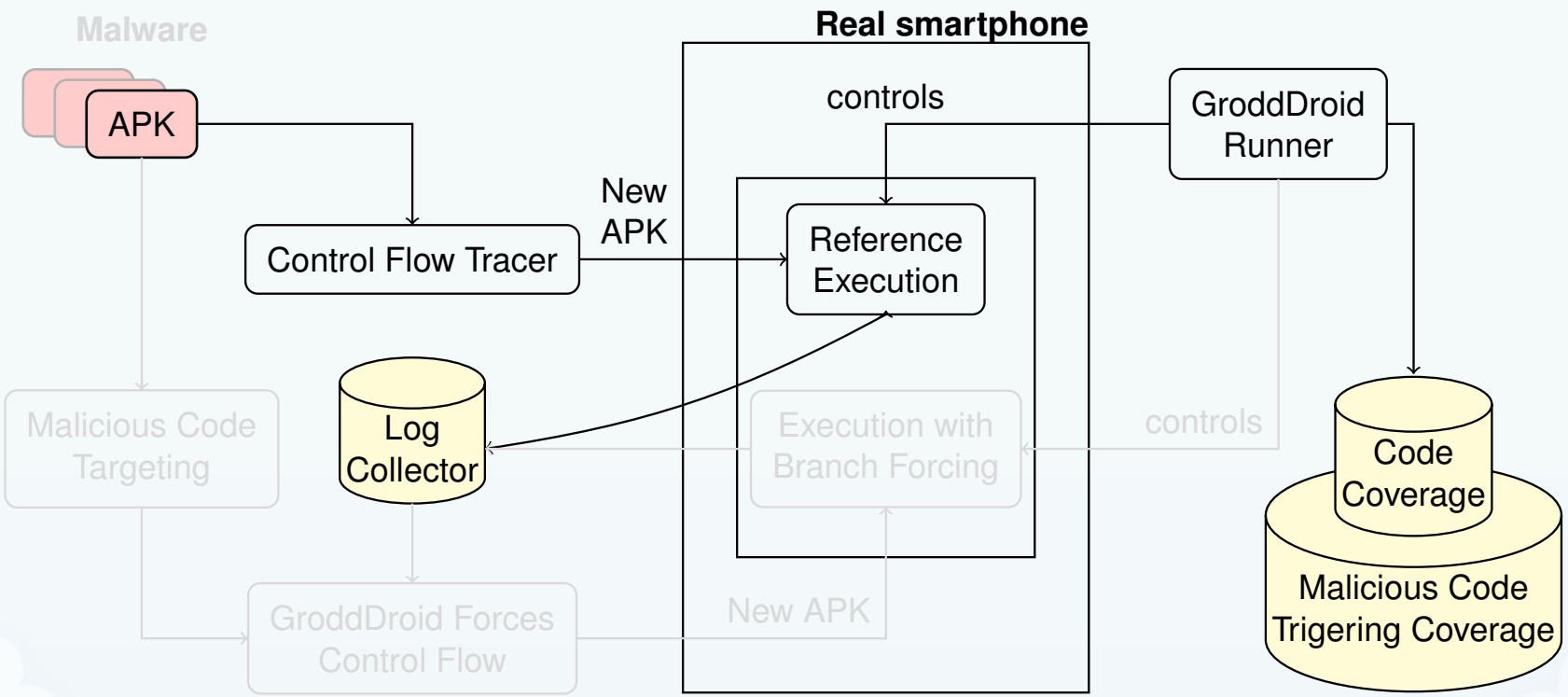
Observer et comprendre les *malware*

- Exécuter des *malware* à large échelle ?
- Eviter les contremesures des développeurs ?

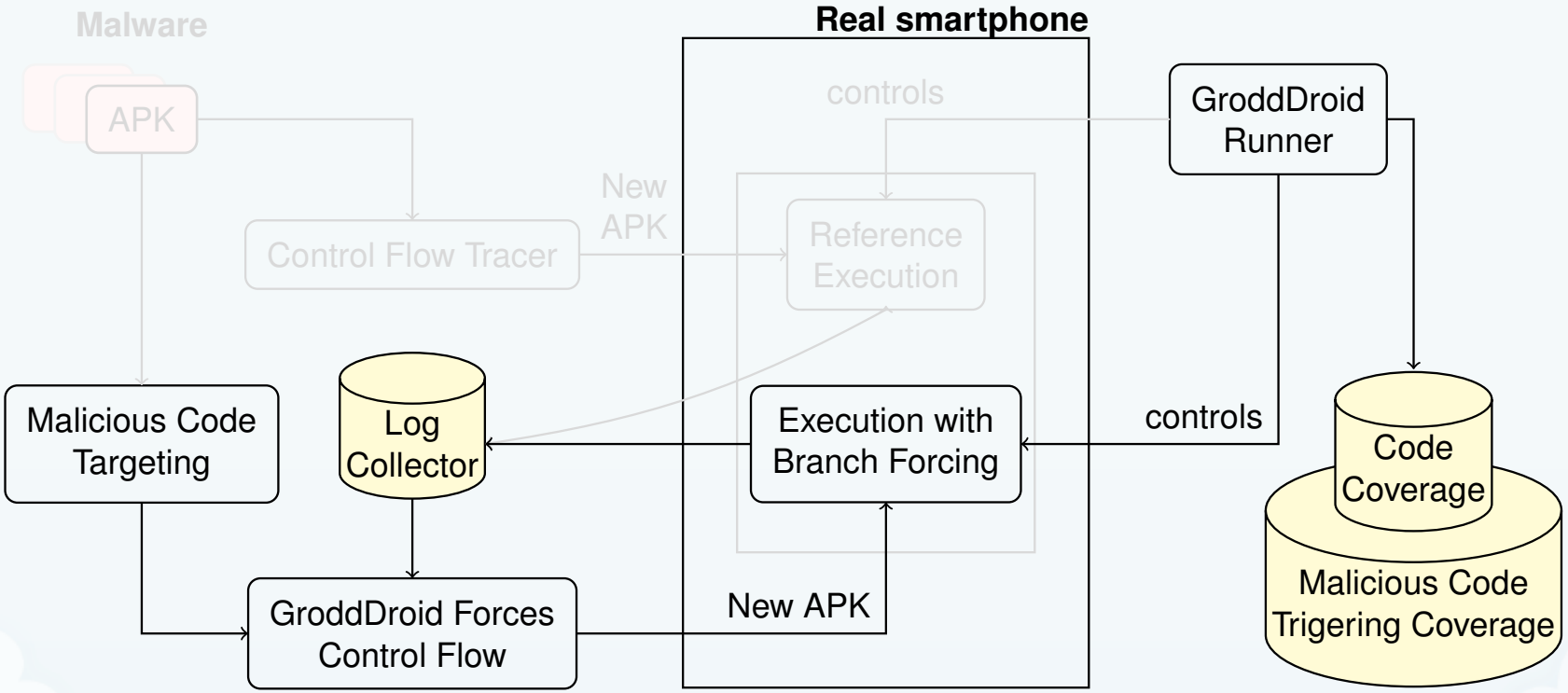
Etape 1 : Cibler le code malveillant



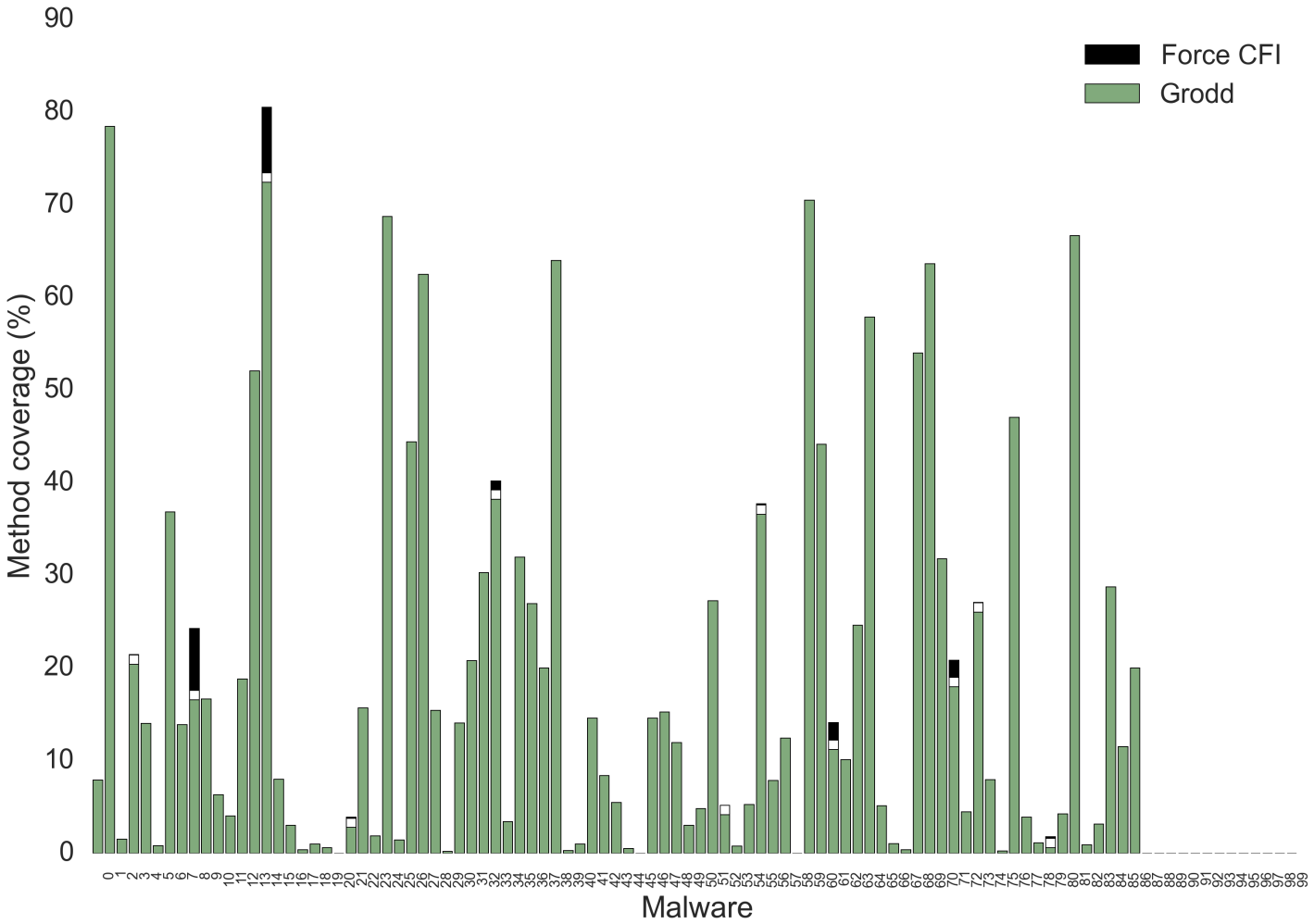
Etape 2 : Exécuter l'APK automatiquement



Etape 3 : Forcer le flot de contrôle



Résultats



Résultats

Couverture des fonctions malveillantes [MALWARE 15] :

- 20% : Monkey
- 24% : GroddDroid
- 28% : GroddDroid + Force CFI

Perspectives à court terme (53 000 *malware*) :

- Prise en compte des appels implicites (47% des chemins)
- Visualisation dynamique aidant au *reverse*

Résultats

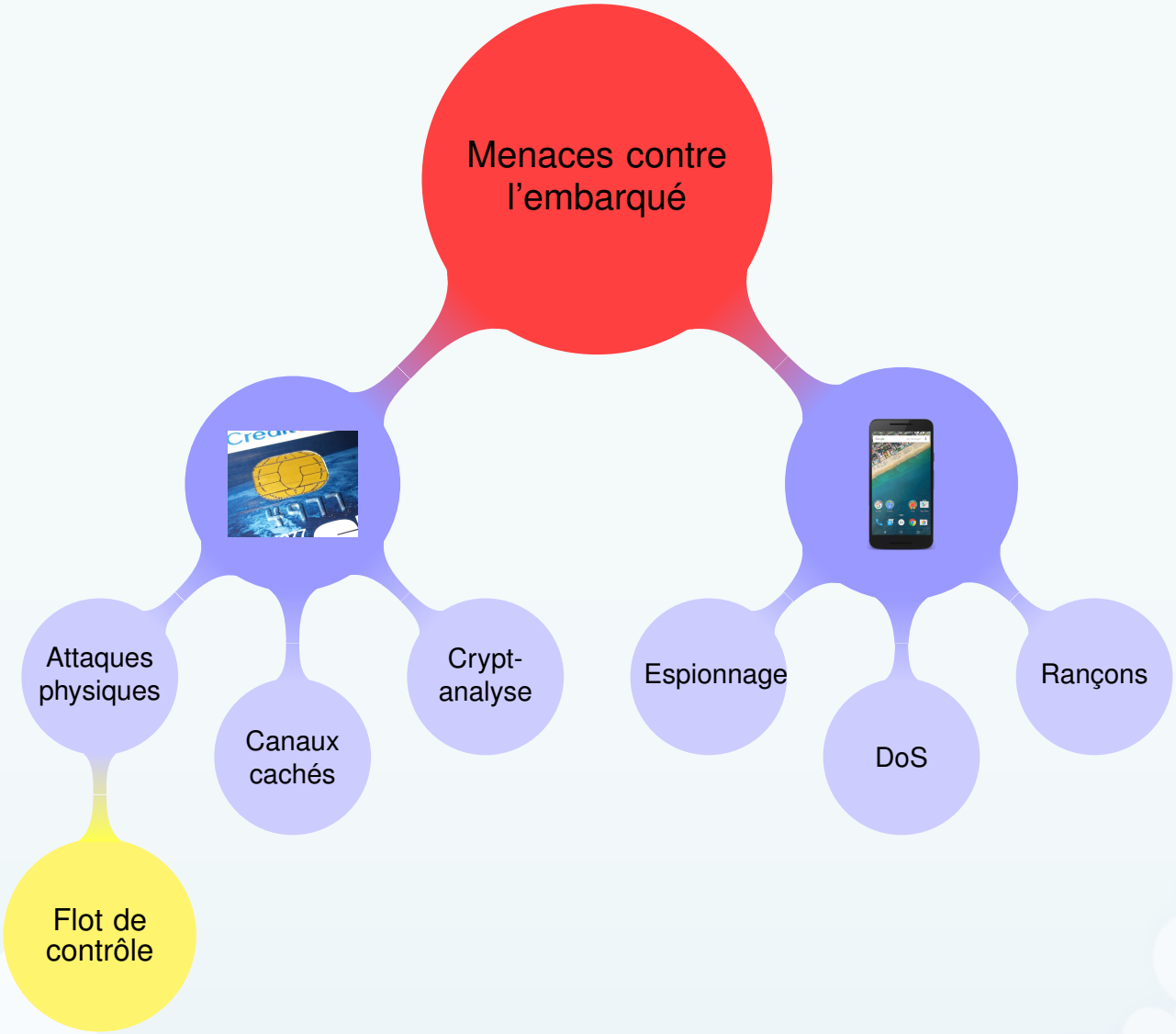
Couverture des fonctions malveillantes [MALWARE 15] :

- 20% : Monkey
- 24% : GroddDroid
- 28% : GroddDroid + Force CFI

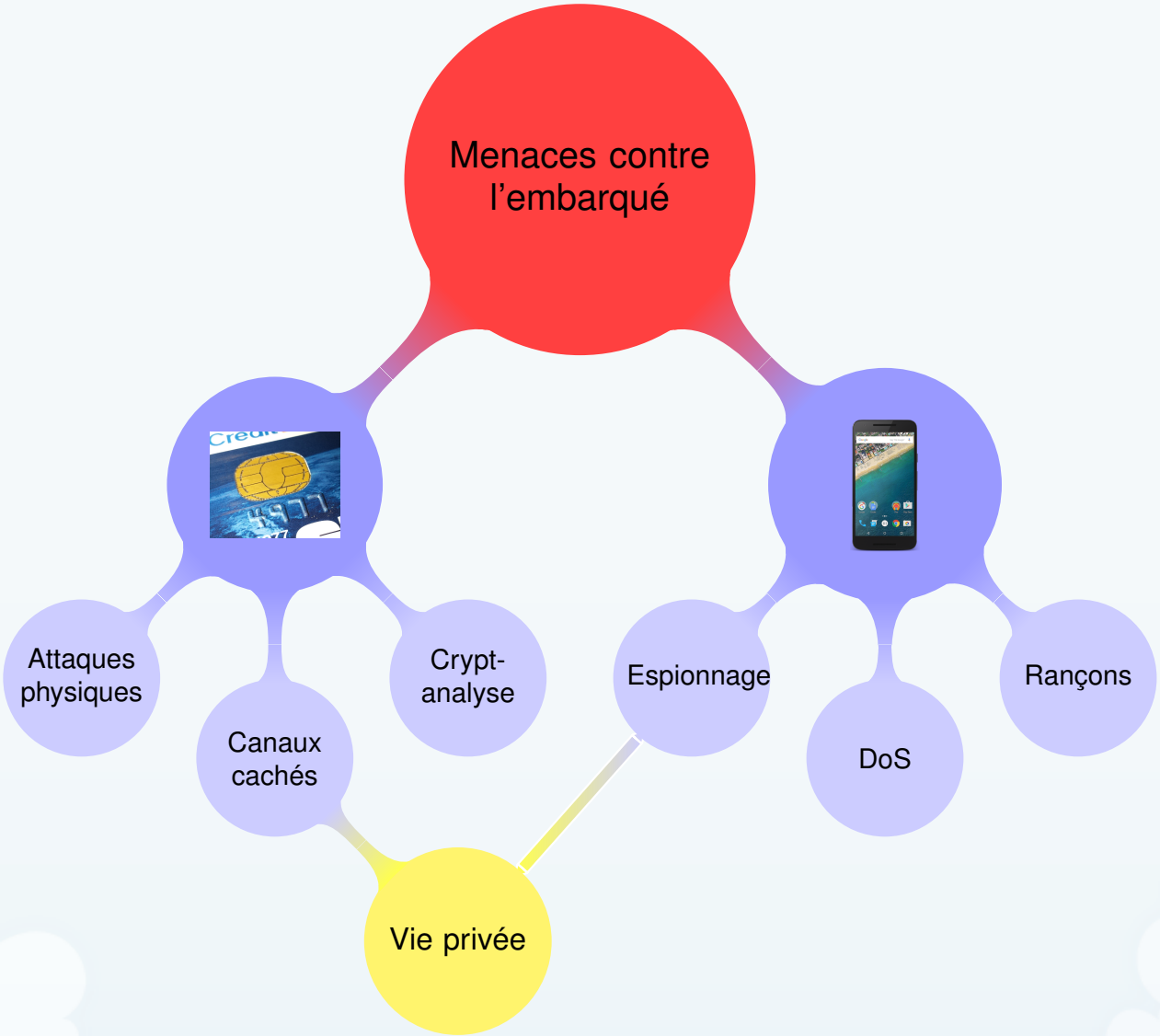
Perspectives à court terme (53 000 *malware*) :

- Prise en compte des appels implicites (47% des chemins)
- Visualisation dynamique aidant au *reverse*

Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Flot de contrôle

Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Vie privée

Protection de la vie privée

Canaux auxiliaires dans les téléphones mobiles

- Canaux auxiliaires plus discrets ?
- Méthodes de détection ?

Protocole pour le transport

- Tickets anonymes et intraçables ?
- Validation d'un ticket en sans contact

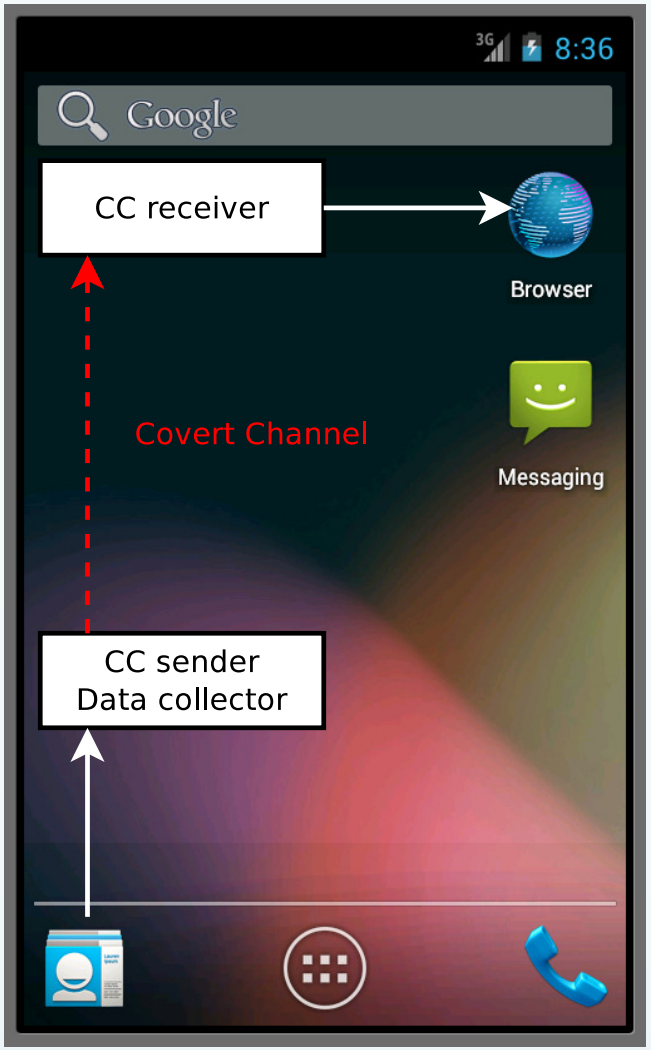
Scénario de référence

Required permissions

INTERNET

GET_TASKS

READ_CONTACTS



Canaux auxiliaires

Définition :

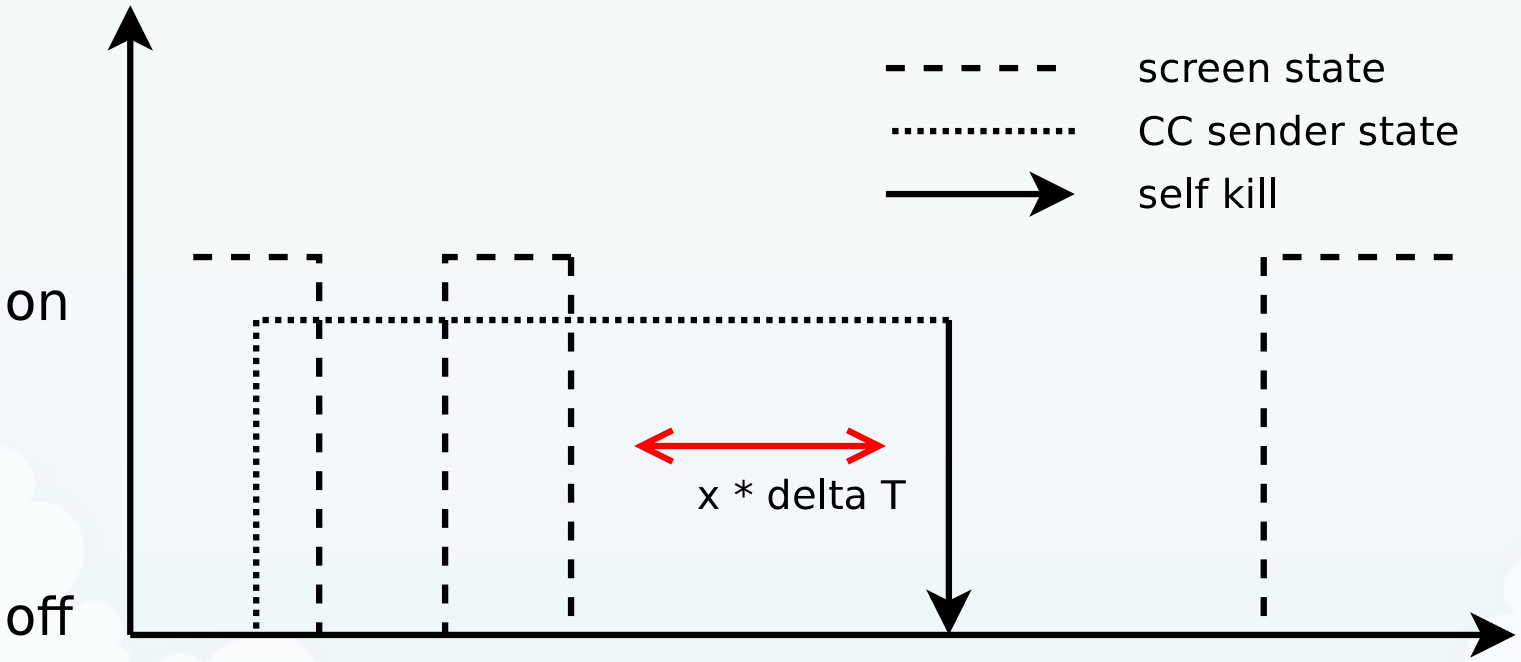
- Moyen de communication “caché”
- Echappe à la détection standard (signature, *tainting*)

Implémentations de la littérature :

- Type d’*Intent* [Marforio et al. 12]
- Paramètre de vibration ou de volume [Schlegel et al. 11]
- Verrou sur des fichiers [Schlegel et al. 11]
- Charge du système [Marforio et al. 12]
- Découverte de Socket Unix [Marforio et al. 12]

Nouveau canal auxiliaire

- L'écran s'éteinte ⇒ début de transmission
- CC sender **est tué** : ⇒ fin de transmission (GET_TASKS)

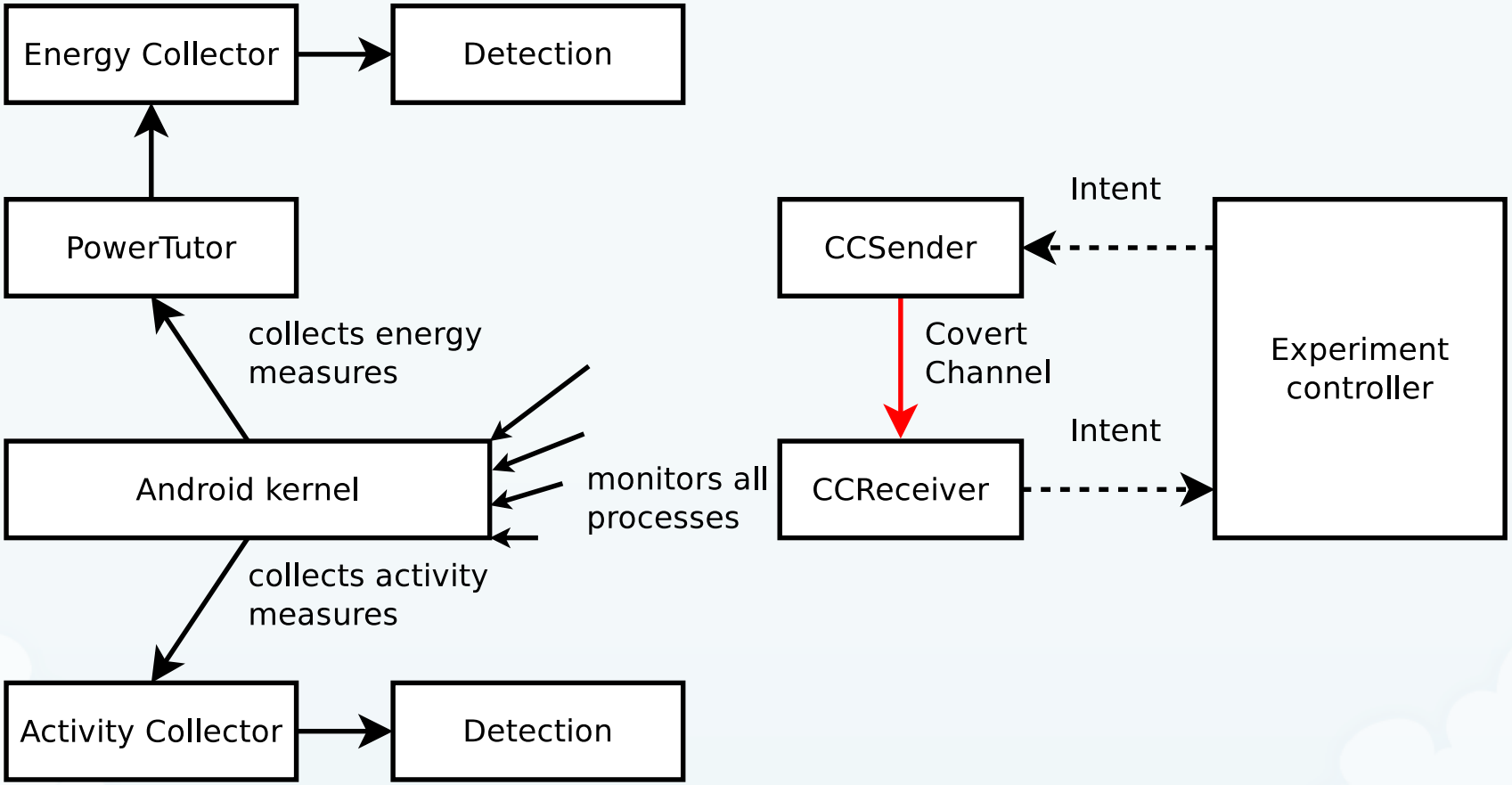


Nouveaux canaux auxiliaires

Covert channel type	Control channel	Data channel	Required permission
CC#1 : Task list/screen	screen state	task list	GET_TASK
CC#2 : Process prio./screen	screen state	process prio.	
CC#3 : Process priorities		process prio.	
CC#4 : Pure screen-based		screen based	WAKE_LOCK
CC#5 : File size		file size	
CC#6 : Memory load		memory	

Bande passante #1 - #4 : 0.2 bps - 18 bps

Collecte des données



Détection

Difficultés

- Processus inconnus
- Type du canal inconnu

Basée activité

- Corrélation des activités des processus [IJCSSE'16]

Basée énergie

- Prédiction de l'énergie consommée [TIFS'16]
  - réseaux de neurones
  - arbres de décision binaires

Résultats

Détection basée activité :

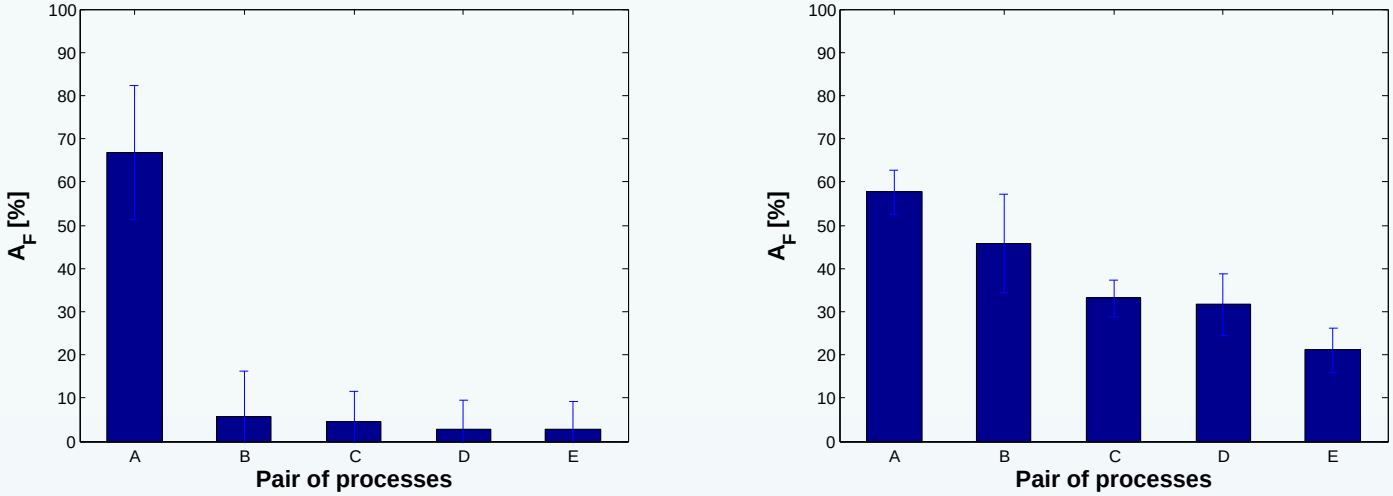
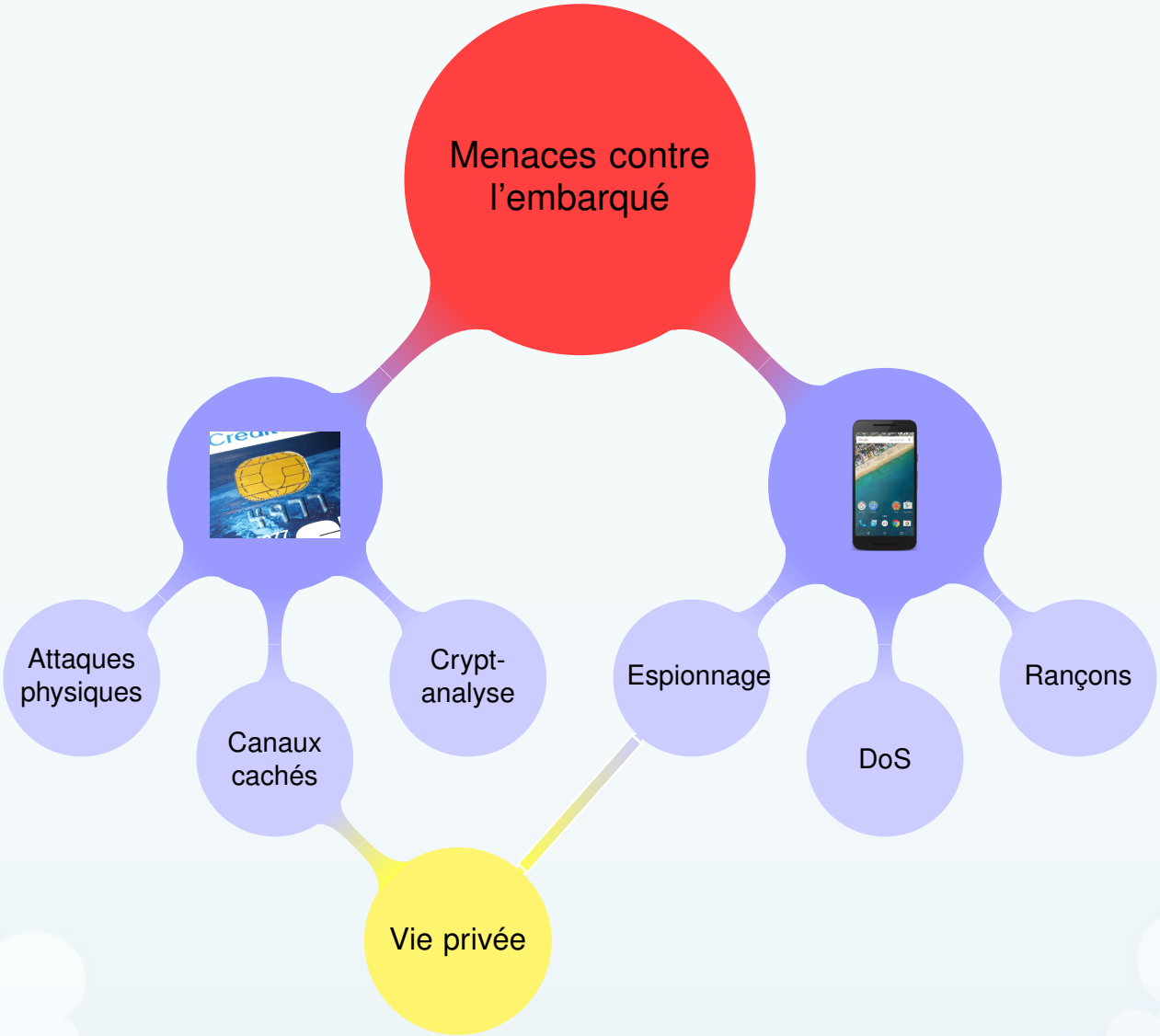


FIGURE – Sans / Avec utilisateur (Canal Aux. "Type d'Intent")

Détection basée énergie : juste à plus de 60% (sans utilisateur)

Embarqué


Menaces contre l'embarqué
Attaques physiques
Canaux cachés
Crypt-analyse
Espionnage
DoS
Rançons
Vie privée

Ticketing

Génère :

$B_k = \frac{1}{g^{s+k+1}}$

Thèse G. Arfaoui

Ensemble de tickets

RATP

Tickets B_k inutilisés

Validation d'un ticket

Preuve de connaissance ZK :

index $k \in [1, max]$

B_k est bien formé

Ticketing

Génère :

$B_k = \frac{1}{g^{s+k+1}}$

Thèse G. Arfaoui

Ensemble de tickets

SANS PAIRING

$\leq$ 300 MS

RATP

Tickets B_k inutilisés

Validation d'un ticket

Preuve de connaissance ZK :

index $k \in [1, max]$

B_k est bien formé

Plan

1. Parcours et formation
2. Enseignement
3. Recherche en sécurité
4. Flot de contrôle
5. Vie privée
6. Conclusion

Principales publications

Systèmes à hautes performances

- Politiques et contrôle d'accès [FGCS'13+14]

Carte à puce

- Intégrité du flot de contrôle [ESORICS'14]
- Protocoles de contrôle d'accès anonymes [PoPETS'15]

Téléphones mobiles

- Attaques par canaux auxiliaires [TIFS'16]
- Observation dynamique de malware **[MALWARE'15]**

Exposés, démos/tutoriels, workshops

Travail collaboratif

Coopération SDS [ESORICS'14, FGCS'13+14]

Coopération nationales (Inria - CentraleSupélec)

- UPMC : Securité des cartes à puces [ESORICS'14]
- CentraleSupélec : Malware [MALWARE'15, LASER'16]

Coopération internationales

- USA : Politiques pour la gestion de crise [FGCS'14]
- Europe : Pologne - Italie - Allemagne
  - Design de malware à canaux auxiliaires [ECTCM'14]
  - Détection de ces malware [TIFS'16]

Structuration de la recherche

Financement de projets de recherche

15-... Kharon Labex CominLabs : avec CentraleSupélec

11-15 ANR LYRICS : avec Orange, Oberthur, Worldline

09-12 ANR AGAPE : avec Inria Sophia, LIRMM, LITA

Comités de programme

Journ. International Journal on Advances in Security

Confs AINA, CTS, HPCS, HPCC, SECOTS

Wks BCSN, BDA4CS, BioSTAR, IWCC, METHOS, SHPCS, SPCLOUD, SPINS, SSO, WTMC

Logiciels

Développer et partager les codes expérimentaux (GPL) :

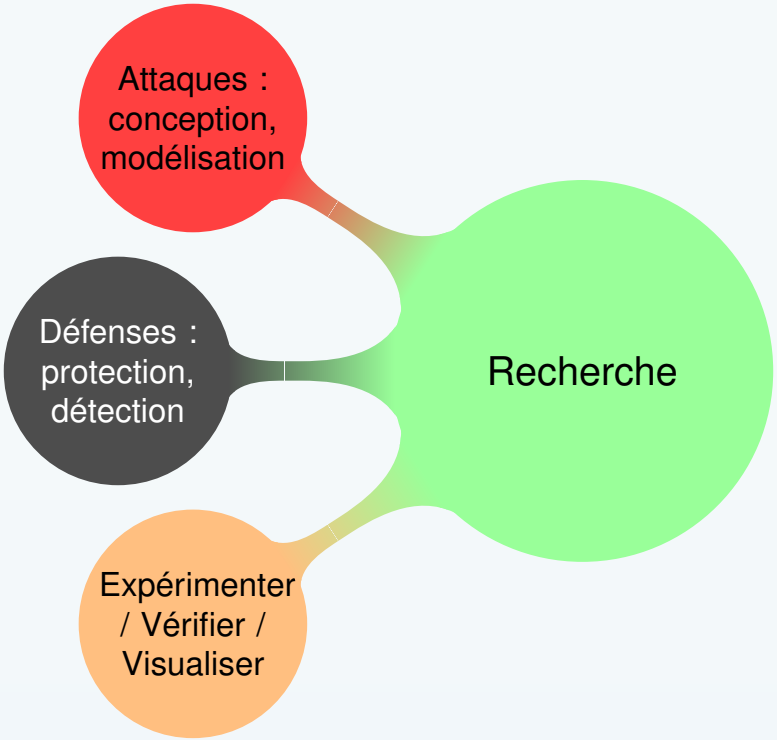
- **Agape** : algorithmes exponentiels
- **CFI-C** : simulation et contremesures pour le CFI
- **SteganoCC** : jouer avec les canaux auxiliaires Android
- **GroddDroid** : aider les *malware* à s'exécuter

A venir :

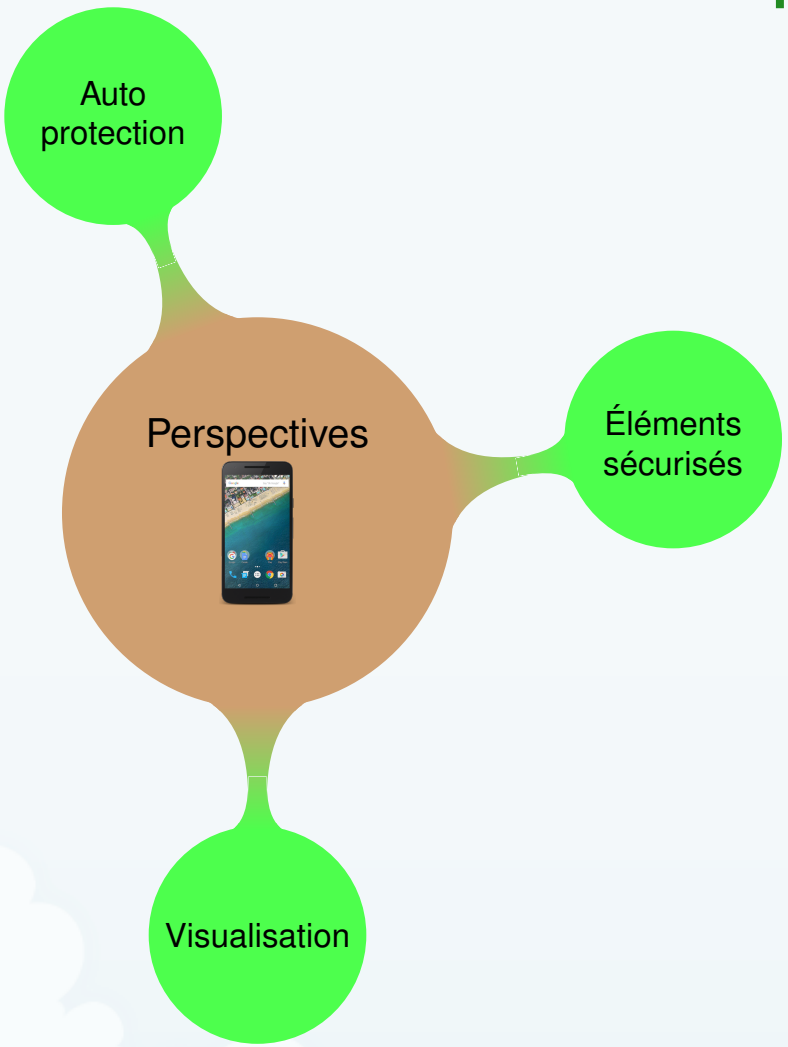
- **GroDDViewer** : visualisation dynamique de malware
- **Kharon** : exécution de vos malware en ligne $\Rightarrow$ LHS

+ contributions étudiants INSA CVL / CentraleSupélec / WUT

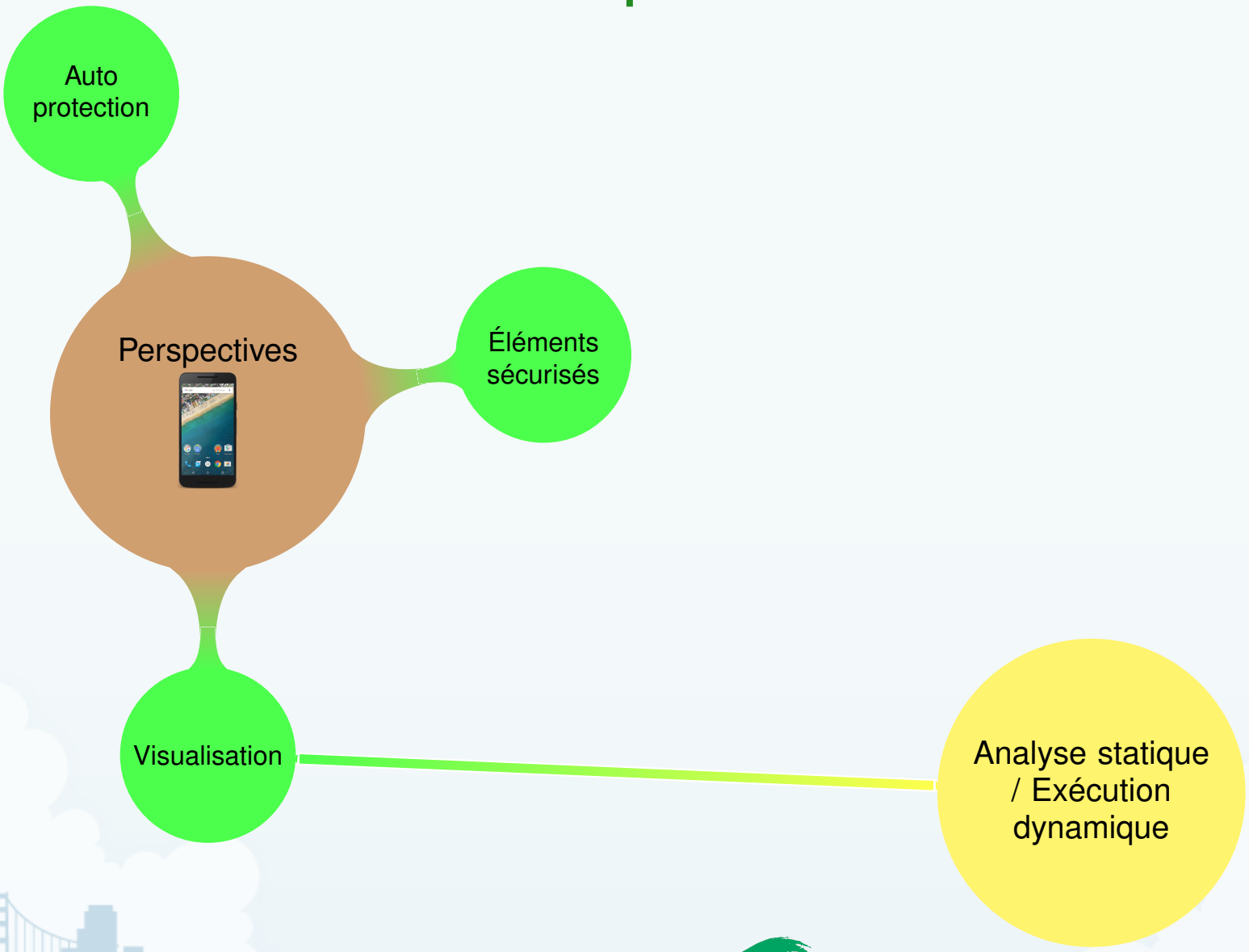
Conclusion


Attaques : conception, modélisation
Défenses : protection, détection
Expérimenter / Vérifier / Visualiser
Recherche

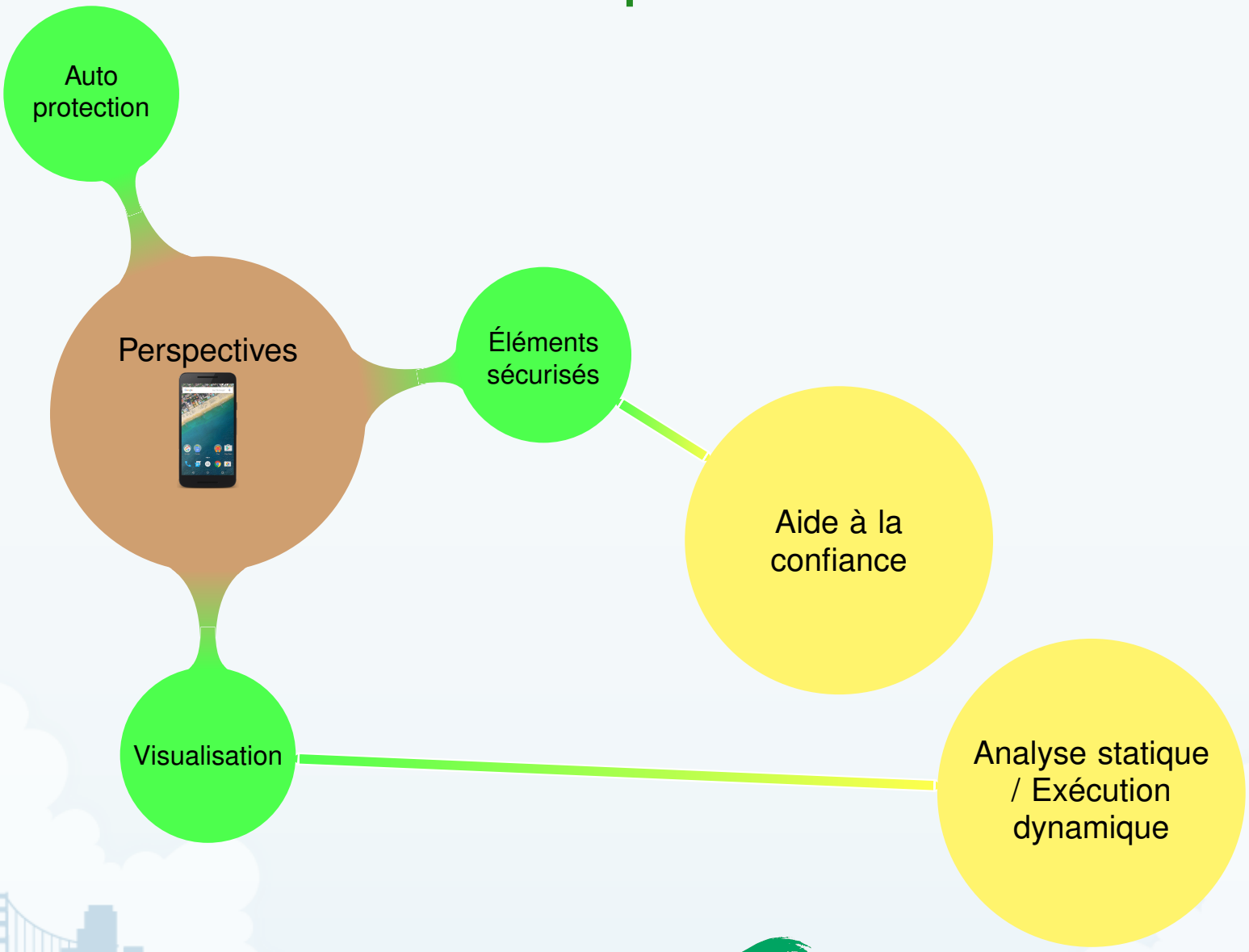
Perspectives



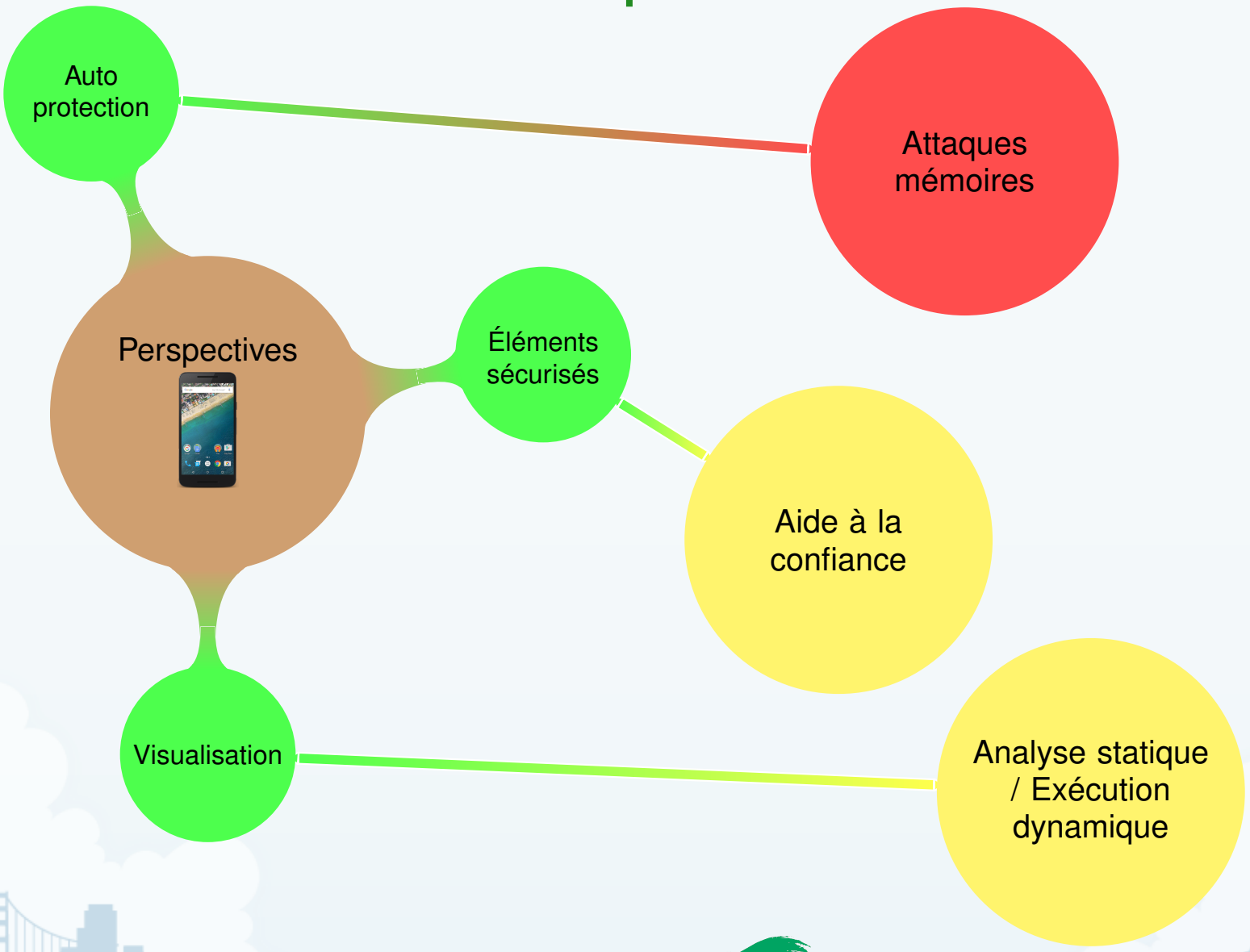
Perspectives



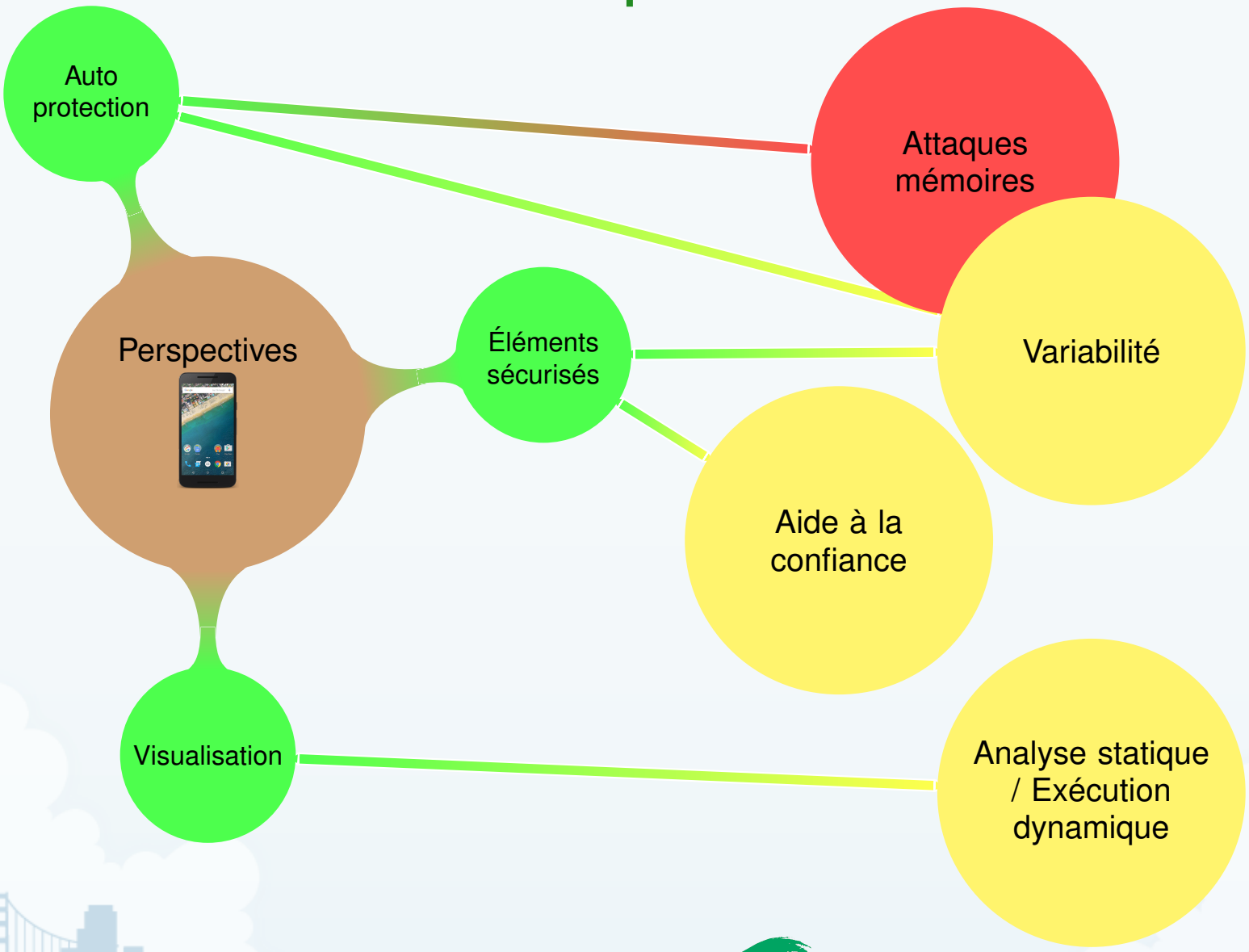
Perspectives



Perspectives



Perspectives



Questions ?

Solutions !
San José
Las Vegas
Miami
Puerto Rico
Prague, Wroclaw
Athènes, Izmir
Chypre
Kénitra
Khouribga
exposés
démos/tutoriels
workshops