From Academia to Real World : a Practical Guide to Hitag-2 RKE System Analysis

Ryad BENADJILA[1] Mathieu RENARD[2]

José LOPES-ESTEVES[2] Chaouki KASMI[2]

[1] ryadbenadjila@gmail.com [2] ANSSI, prenom.nom@ssi.gouv.fr

8 juin 2017

Symposium sur la Sécurité des Technologies de l'Information et des Communications

Contrôle d'accès dans le monde automobile

- Pour l'ouverture/fermeture, le démarrage.

Mécanique

Contrôle d'accès dans le monde automobile

- Pour l'ouverture/fermeture, le démarrage.

Mécanique

Smart-key

Contrôle d'accès dans le monde automobile

- Pour l'ouverture/fermeture, le démarrage.

Contrôle d'accès dans le monde automobile

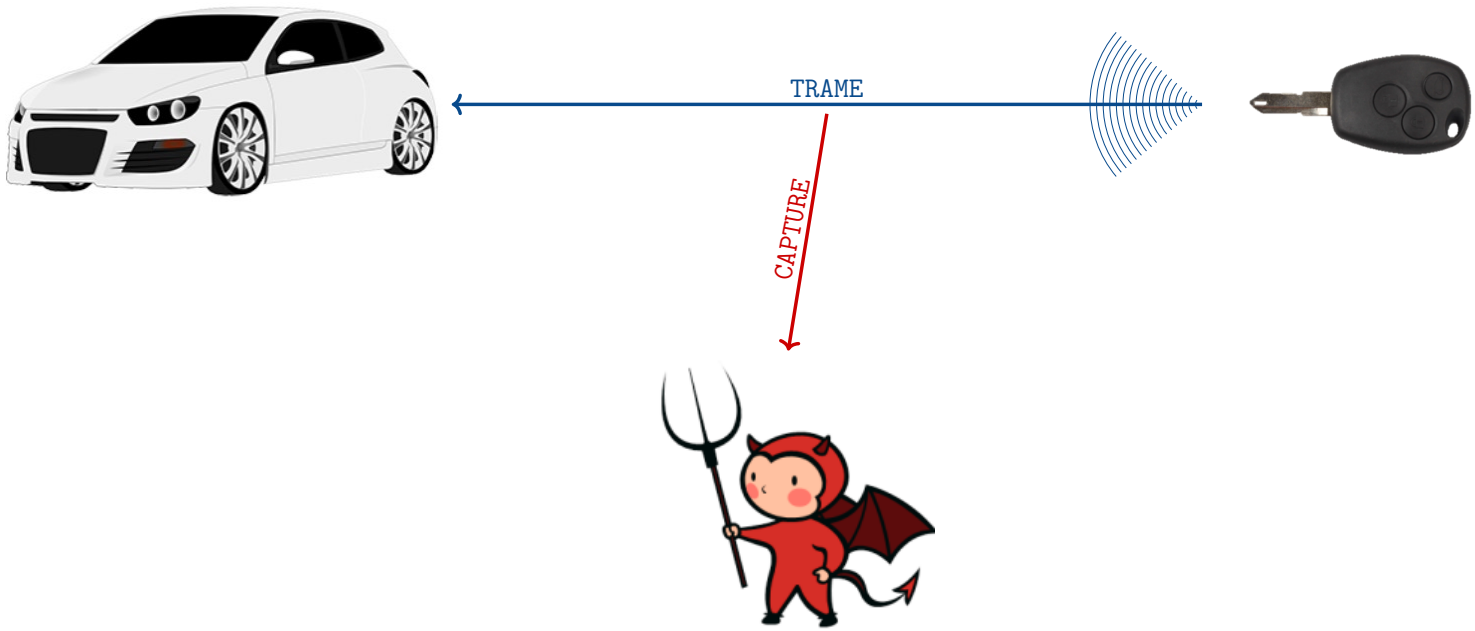
- Pour l'ouverture/fermeture, le démarrage.

Remote Keyless Entry

- RKE :
  1. Communication mono-directionnelle entre la clé et l'ECU.

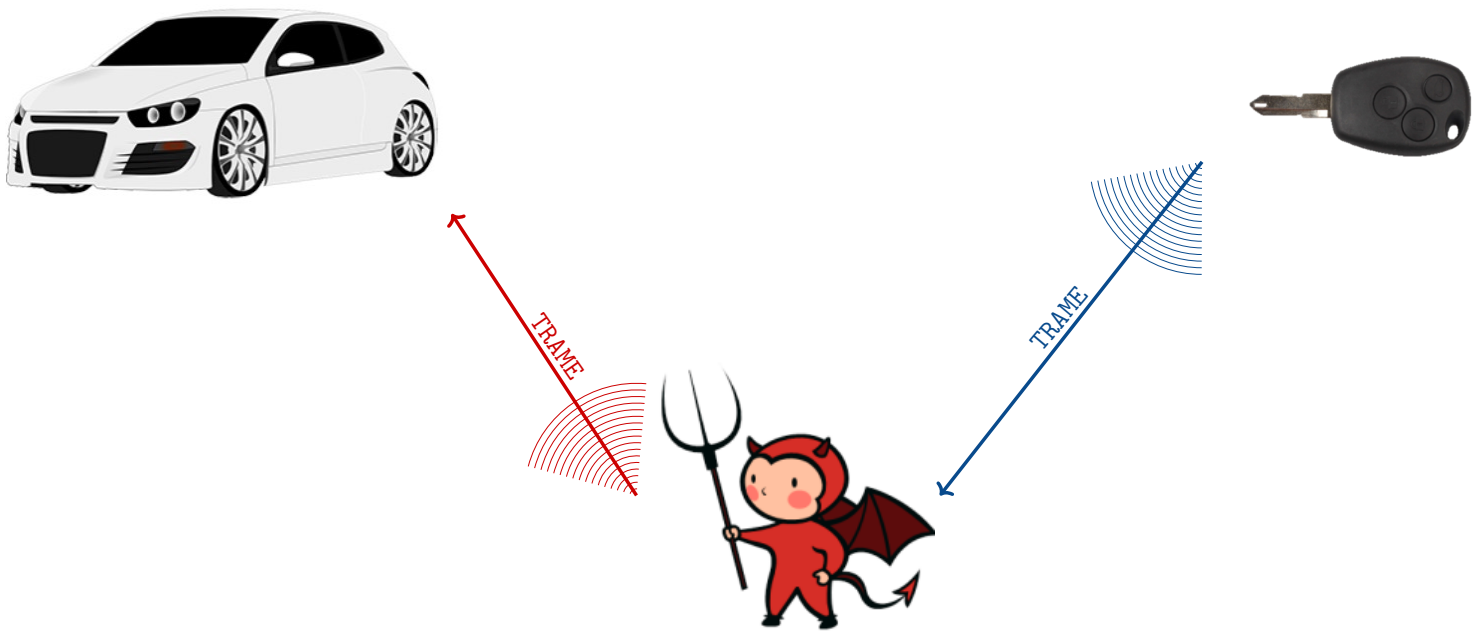
Remote Keyless Entry

- RKE :
  1. Communication mono-directionnelle entre la clé et l'ECU.
  2. Menaces : capture,



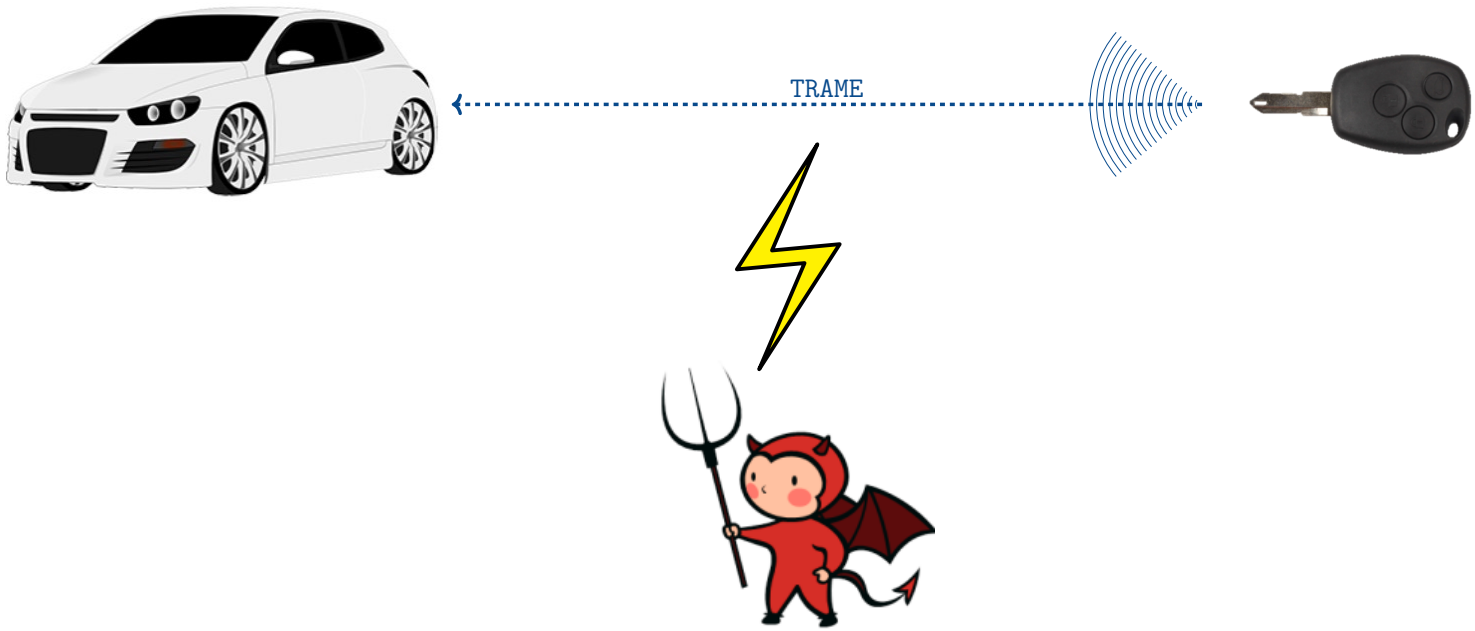
Remote Keyless Entry

- RKE :
  1. Communication mono-directionnelle entre la clé et l'ECU.
  2. Menaces : capture, rejeu,



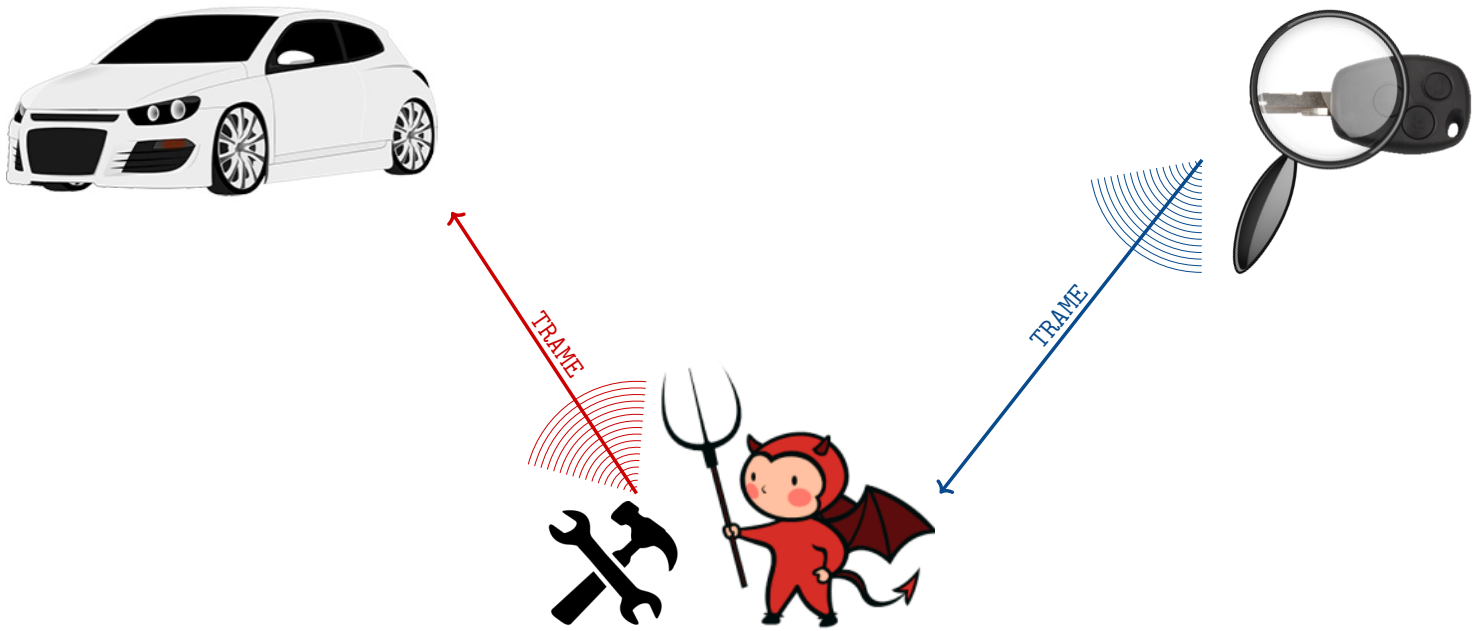
Remote Keyless Entry

- RKE :
  1. Communication mono-directionnelle entre la clé et l'ECU.
  2. Menaces : capture, rejeu, brouillage,



Remote Keyless Entry

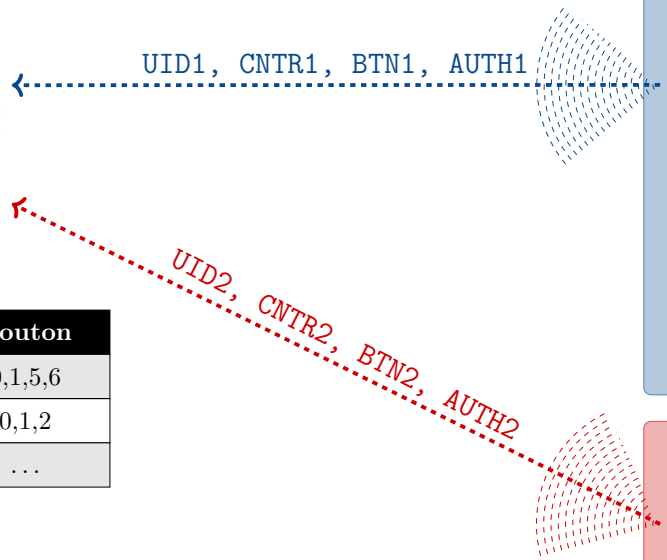
- RKE :
  1. Communication mono-directionnelle entre la clé et l'ECU.
  2. Menaces : capture, rejeu, brouillage, *spoofing*, ...



Remote Keyless Entry

ECU

UID	Clé	Compteur	Bouton
UID1	K1	10	0,1,5,6
UID2	K2	200	0,1,2
...	...	...	...



Remote Keyless Entry

UID1, CNTR1, BTN1, AUTH1

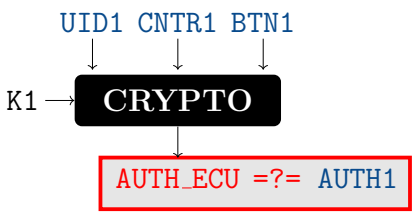
UID2, CNTR2, BTN2, AUTH2

ECU

UID	Clé	Compteur	Bouton
UID1	K1	10	0,1,5,6
UID2	K2	200	0,1,2
...	...	...	...

$\texttt{BTN1} \in \{0,1,5,6\}$?

$\texttt{CNTR1} \in [10, 10+\delta]$?



Remote Keyless Entry

UID1, CNTR1, BTN1, AUTH1

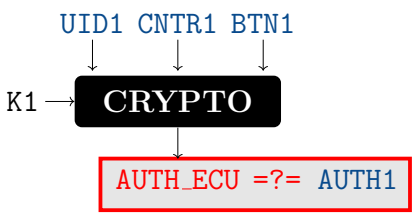
UID2, CNTR2, BTN2, AUTH2

ECU

UID	Clé	Compteur	Bouton
UID1	K1	11	0,1,5,6
UID2	K2	200	0,1,2
...	...	...	...

$\texttt{BTN1} \in \{0,1,5,6\}$?

$\texttt{CNTR1} \in [10, 10+\delta]$?



Usenix 2016 : attaques sur les RKE

- Article Usenix 2016 « Lock It and Still Lose It - On the (In)Security of Automotive Remote Keyless Entry Systems ».

- Deux attaques présentées :
  1. Volkswagen – bonne cryptographie mais clés partagées par tous les véhicules depuis les années 2000 !
  2. PCF7946 – transpondeur de Philips/NXP utilisant l'algorithme Hitag-2. Une nouvelle attaque a été présentée.

Usenix 2016 : attaques sur les RKE

- Objectif : reproduire l'attaque sur `PCF7946`.
  1. Capturer et décoder des trames radio.
  2. Implémenter l'attaque sur `Hitag-2` pour retrouver la clé secrète.
  3. Forger des trames valides.

- Contraintes : *black-box*.
  - Ne pas casser la voiture !
  - Pas d'attaque matérielle sur le transpondeur `PCF7946`.

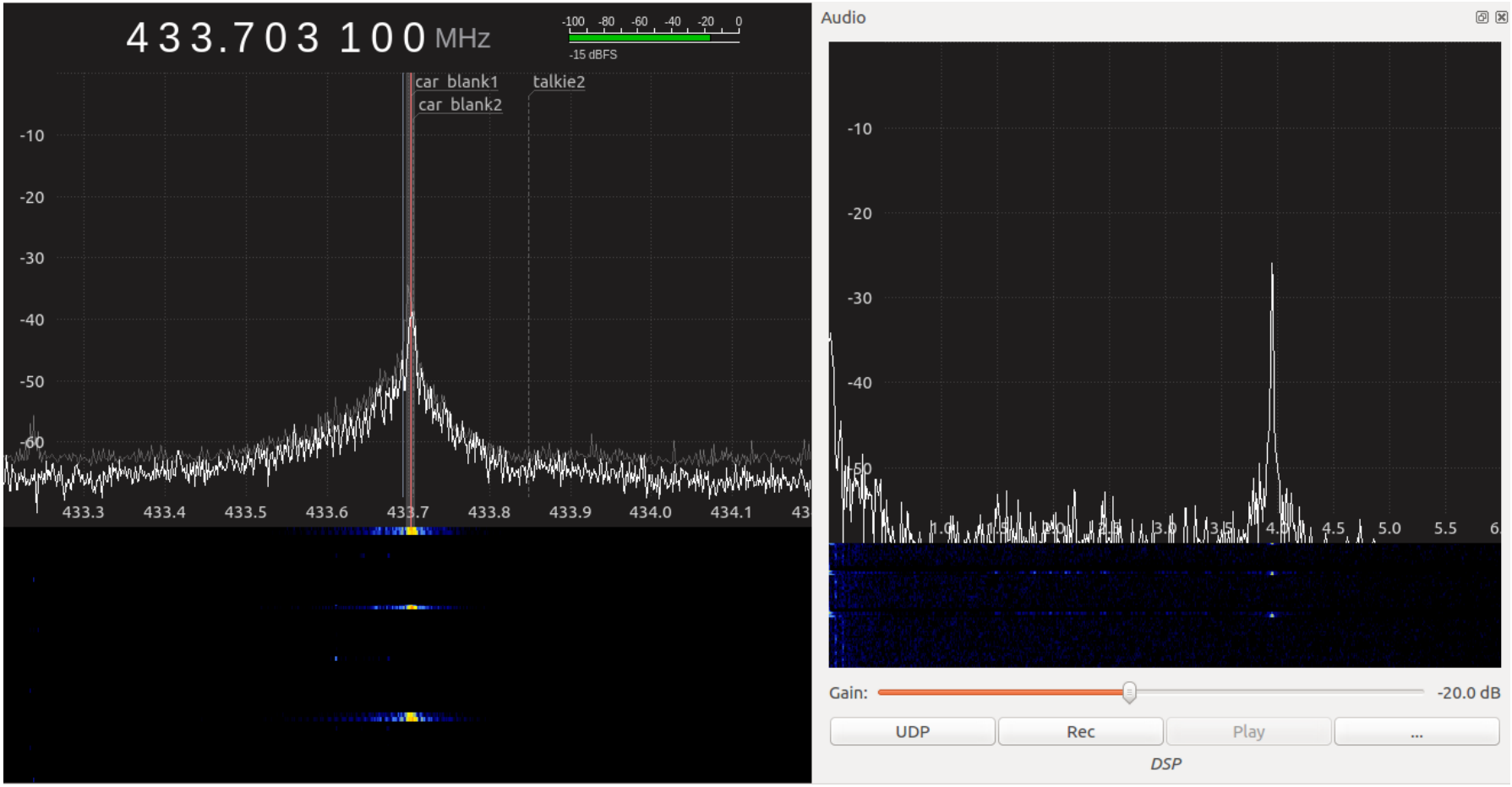
Analyse des signaux radiofréquences

- Nous devons déterminer :
  - La fréquence de la porteuse et la bande passante.
  - La modulation.
  - Le codage canal.
  - La structure des paquets.

- Analyse *white-box*, informations connues.

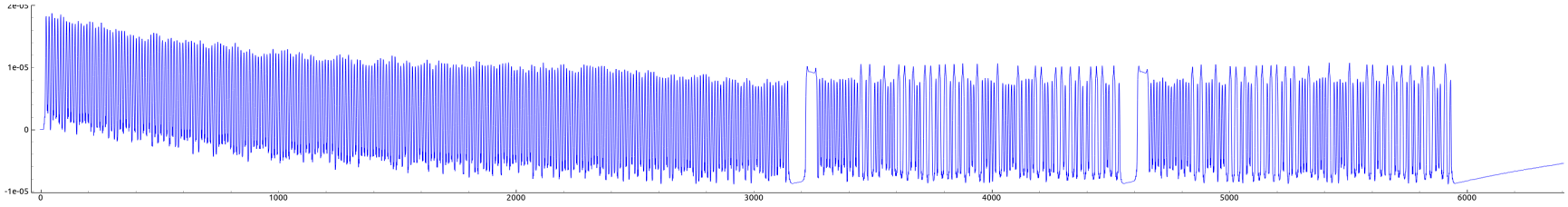
Paramètre	Valeur
Fréquences	ISM 433 MHz
Modulation	ASK/FSK
Codage canal	Manchester/NRZ
Format des trames	Usenix 2016

Démodulation : analyse spectrale

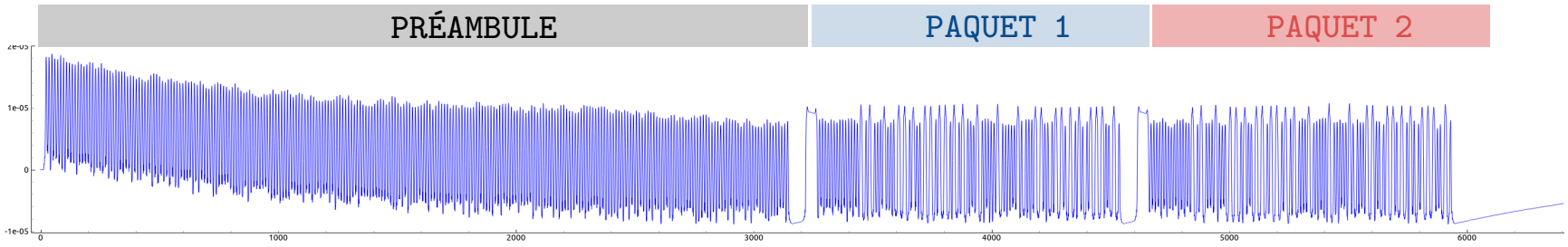


- Modulation utilisée : modulation d'amplitude (ASK).

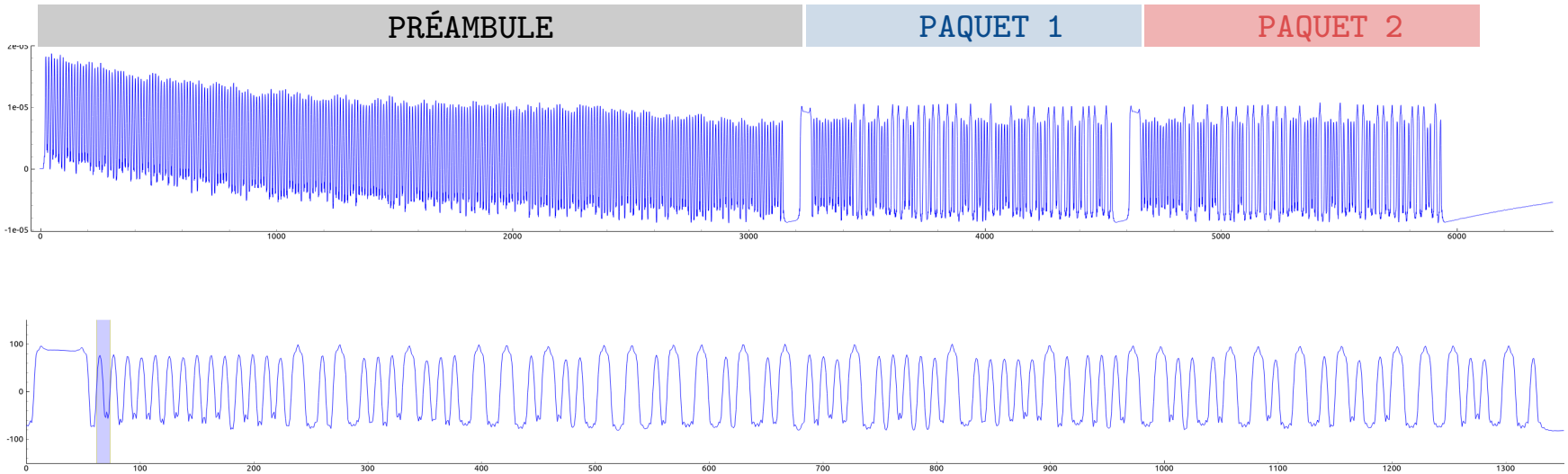
Décodage



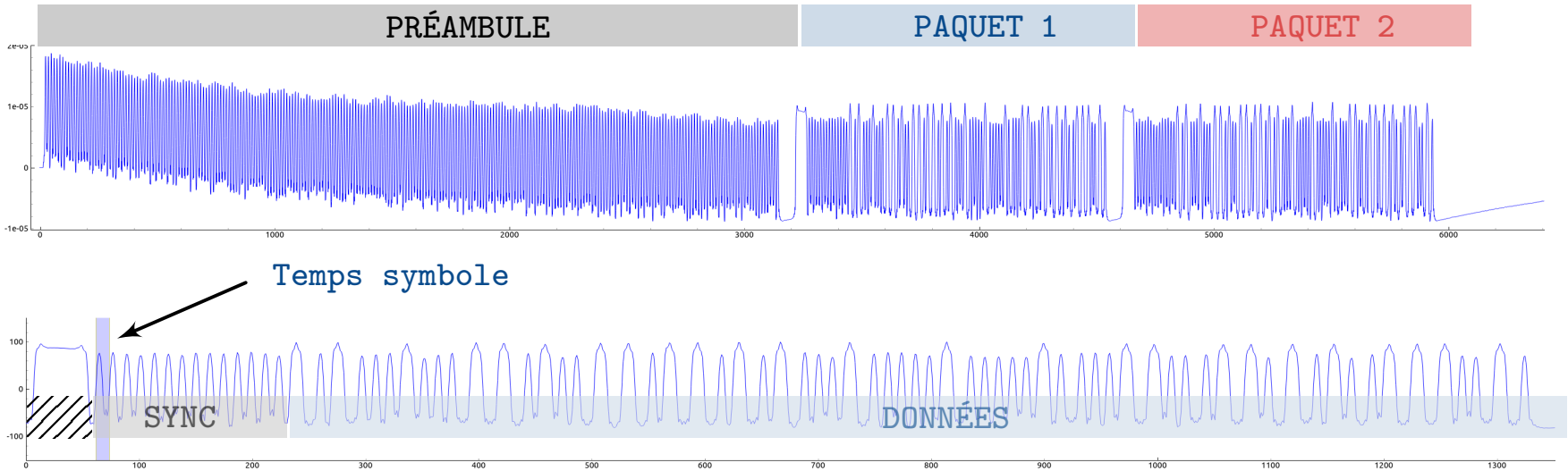
Décodage


PRÉAMBULE
PAQUET 1
PAQUET 2
1e-05
0
-1e-05
0
1000
2000
3000
4000
5000
6000

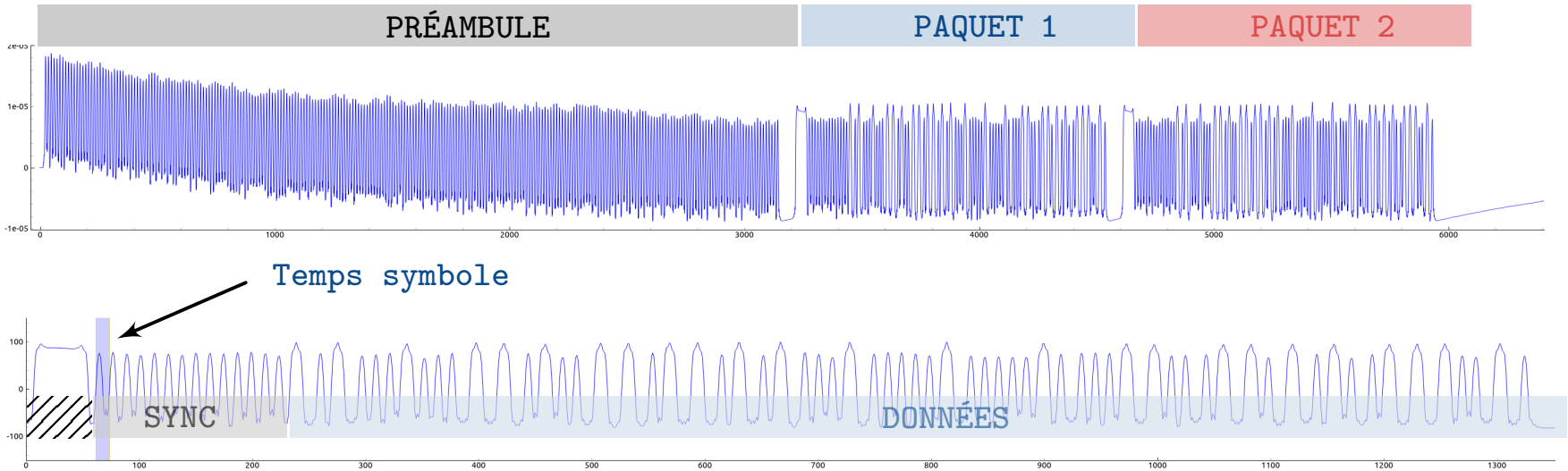
Décodage



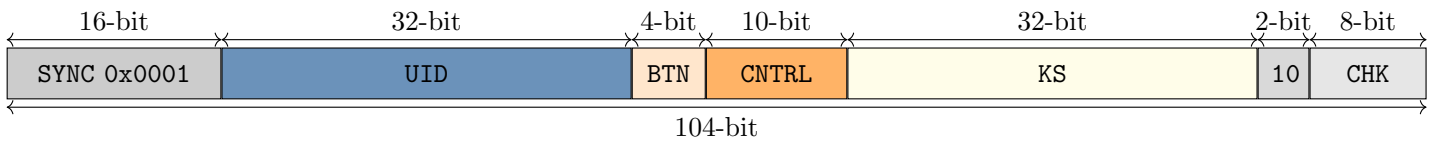
Décodage



Décodage



- Résultats :
  - Modulation ASK.
  - Codage de Manchester.
  - Observation des invariants pour remonter aux données.
  - Utilisation de la somme de contrôle.

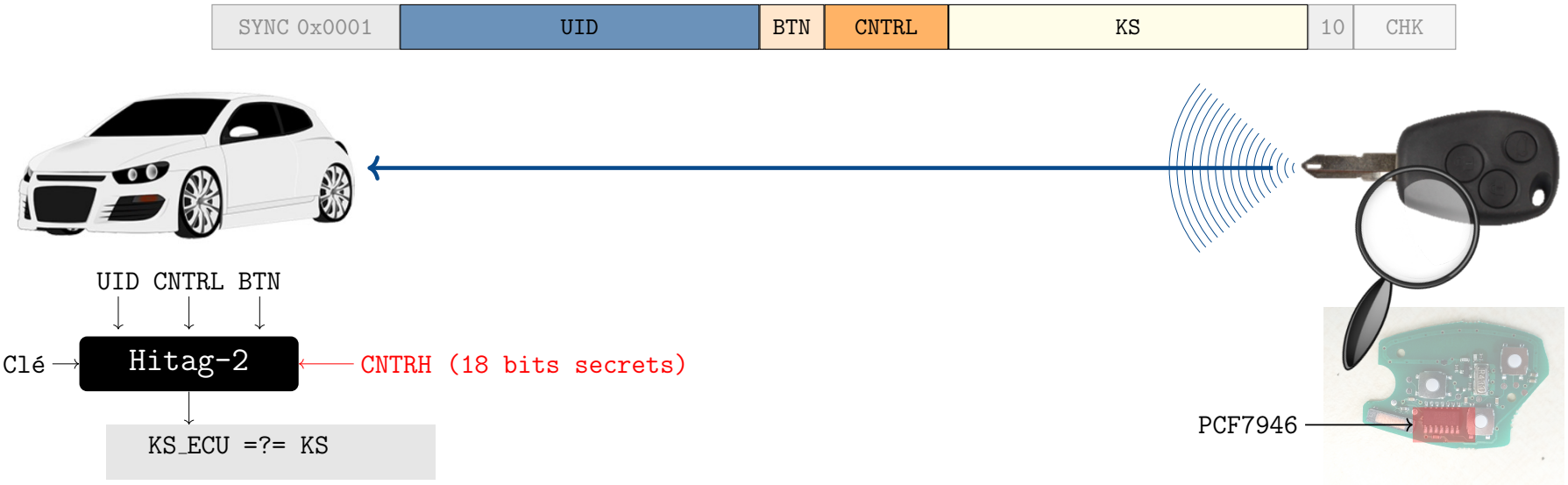


L'algorithme Hitag-2

- *Stream cipher* Philips (NXP) de la fin des années 90.
- *Reverse engineering* matériel en 2007.

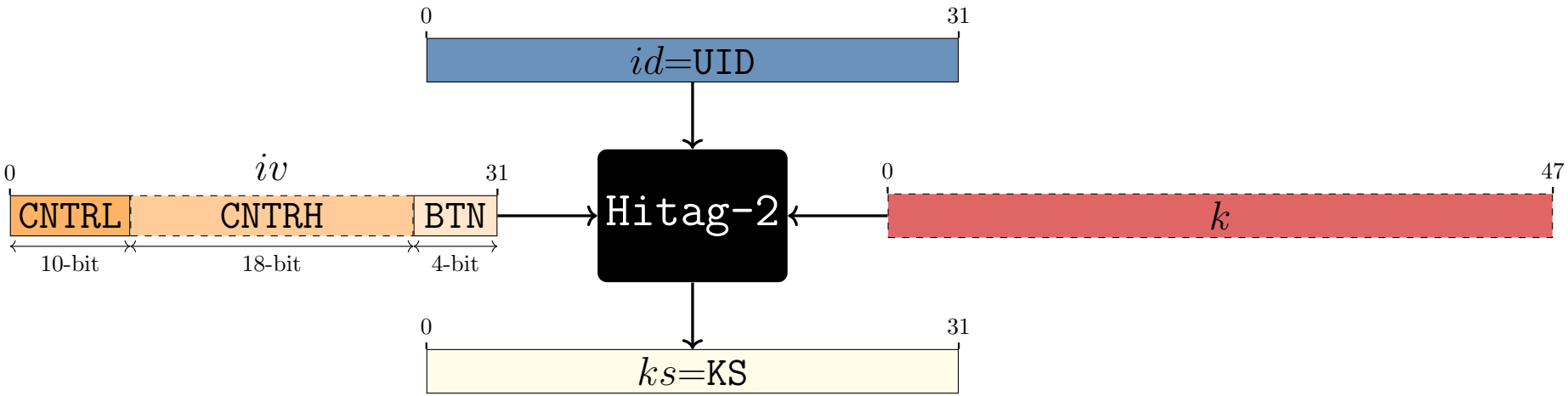
L'algorithme Hitag-2

- *Stream cipher* Philips (NXP) de la fin des années 90.
- *Reverse engineering* matériel en 2007.
- Utilisation dans un contexte RKE :

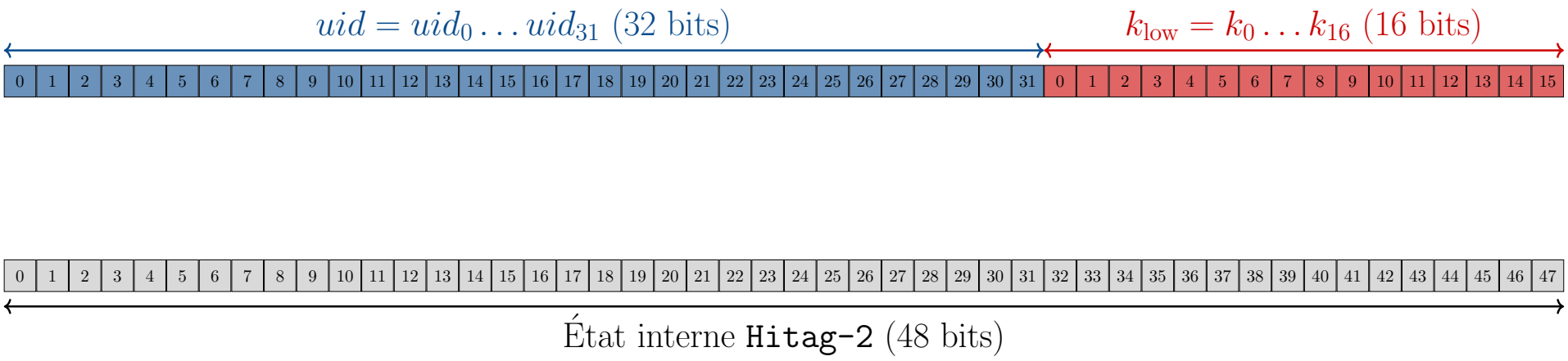


L'algorithme Hitag-2

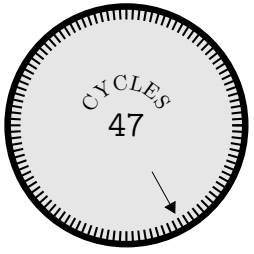
- *Stream cipher* Philips (NXP) de la fin des années 90.
- *Reverse engineering* matériel en 2007.
- Utilisation dans un contexte RKE :

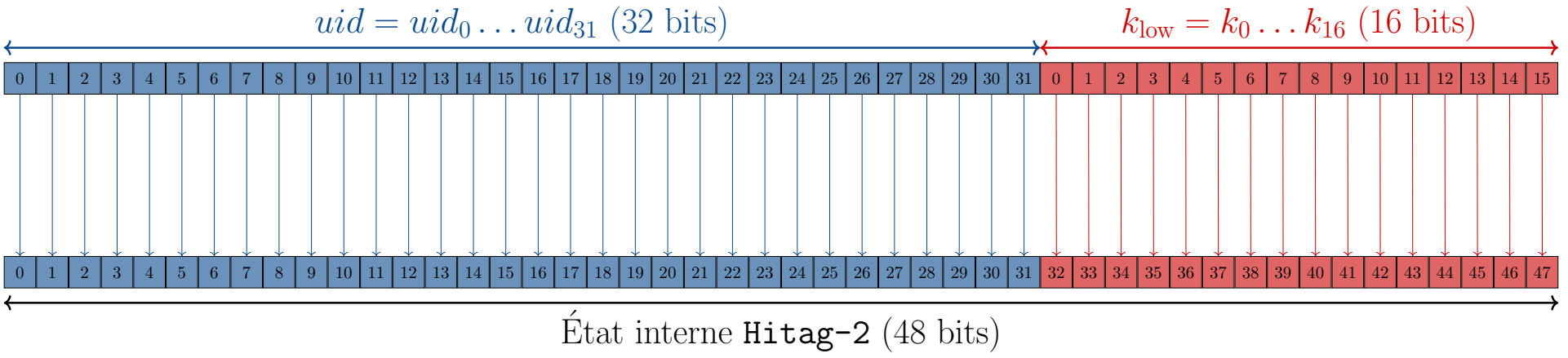


L'algorithme Hitag-2 : phase d'intialisation

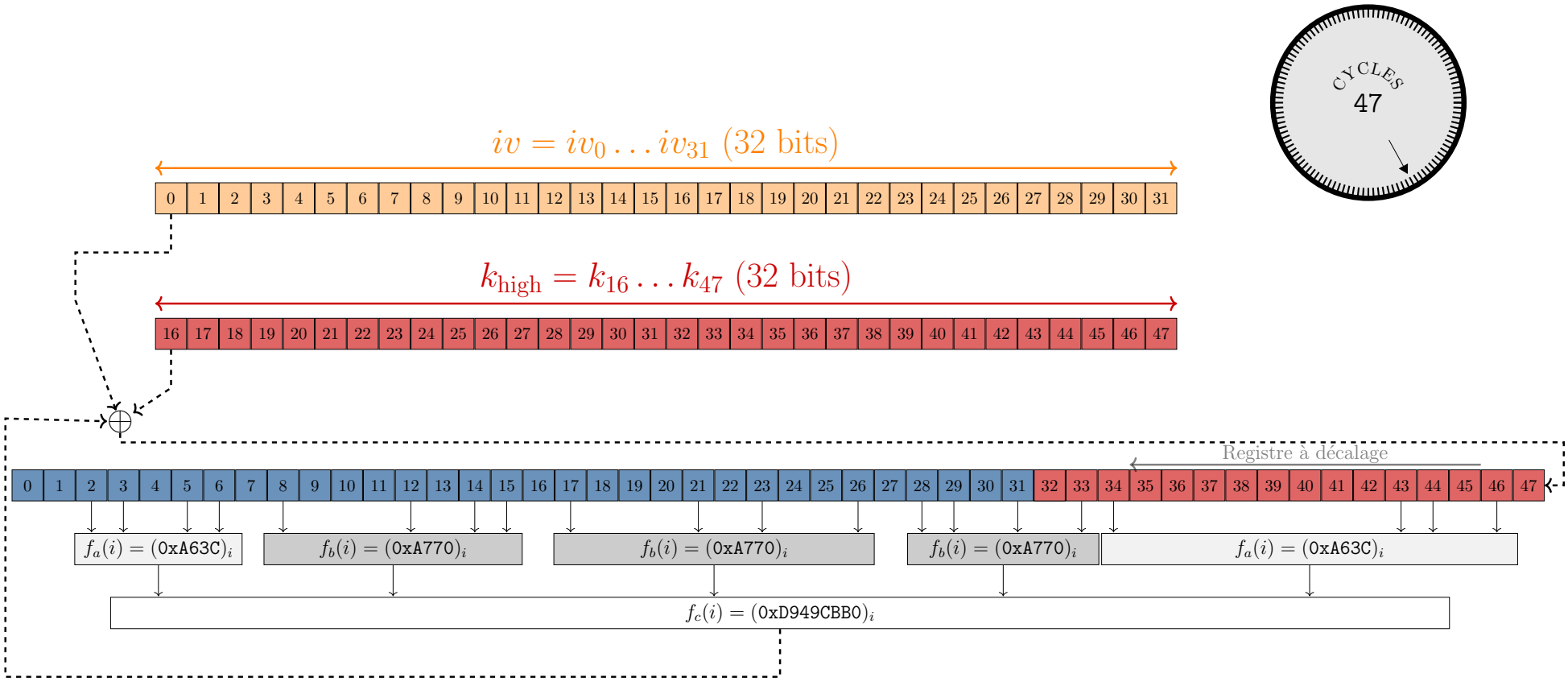


L'algorithme Hitag-2 : phase d'intialisation

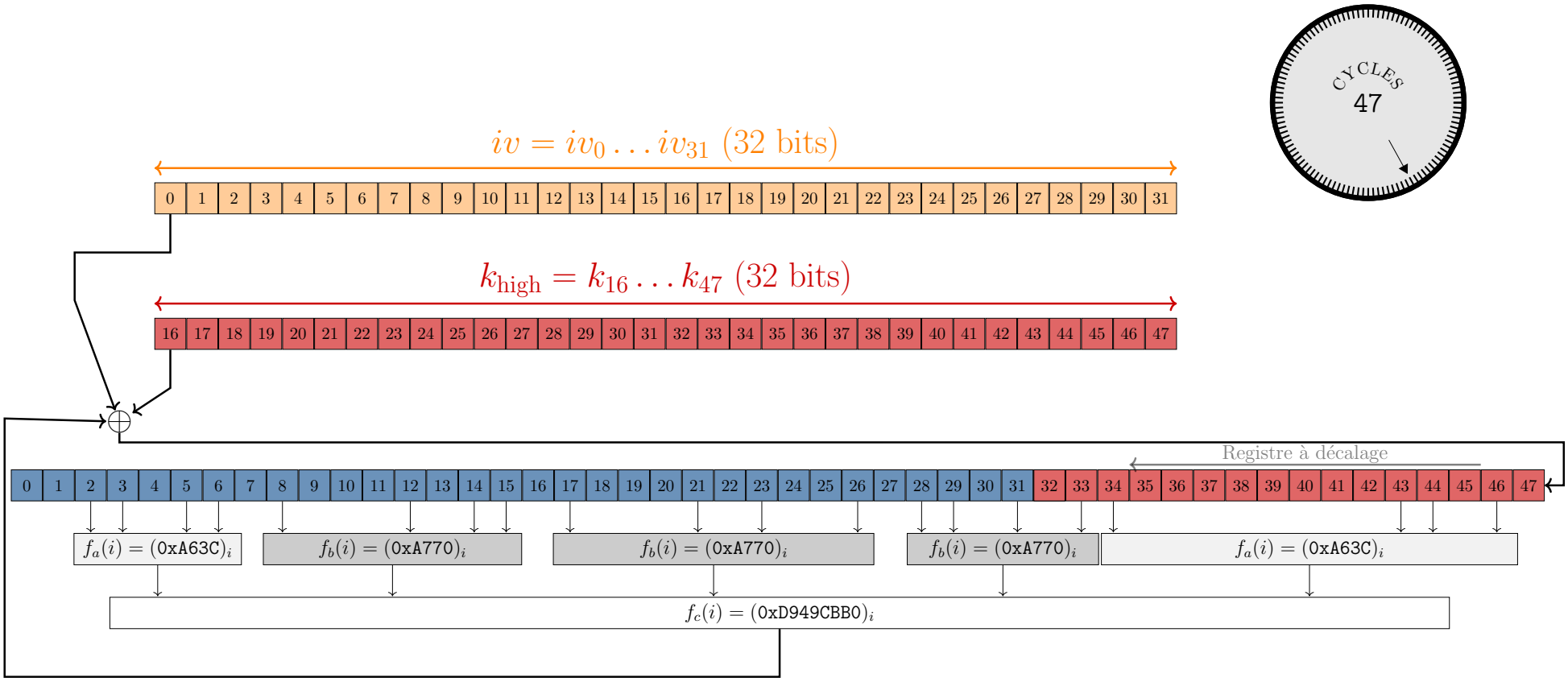




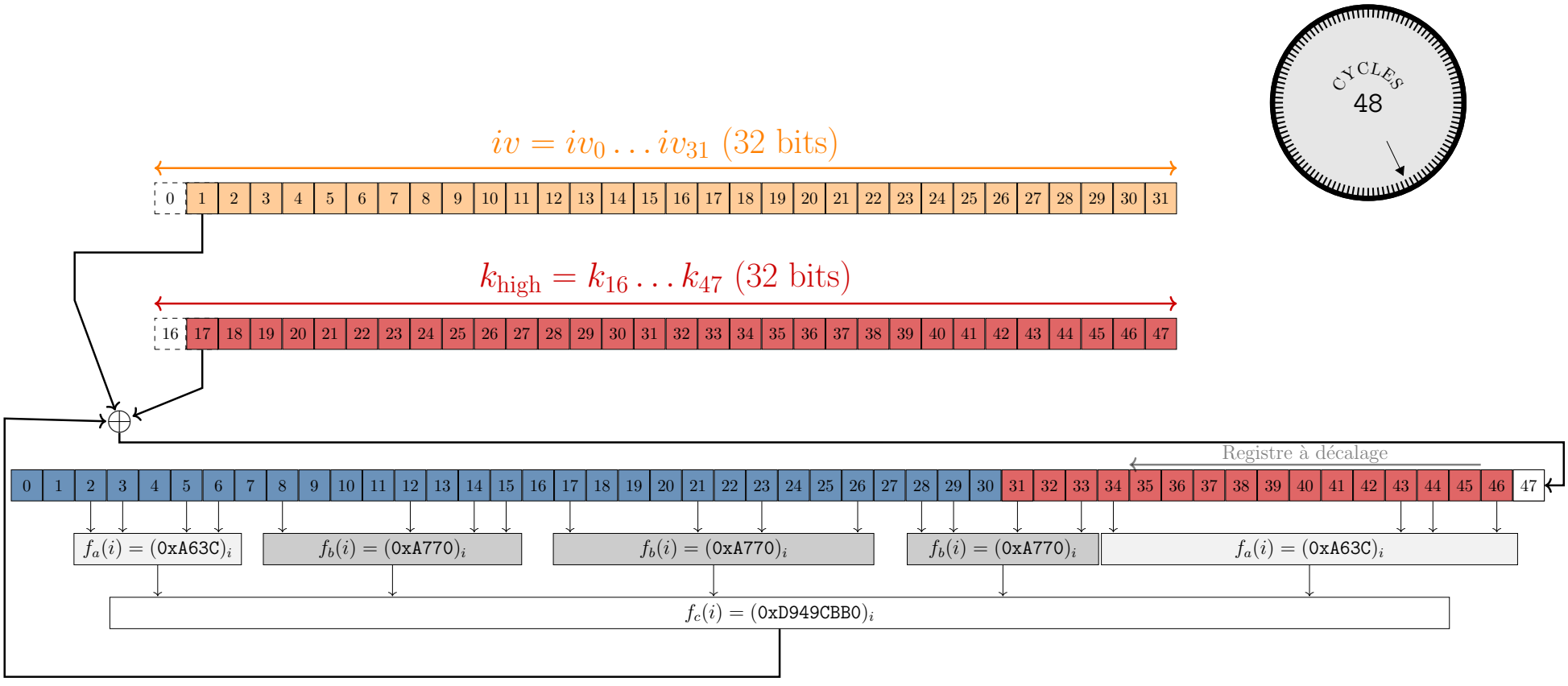
L'algorithme Hitag-2 : phase de randomisation



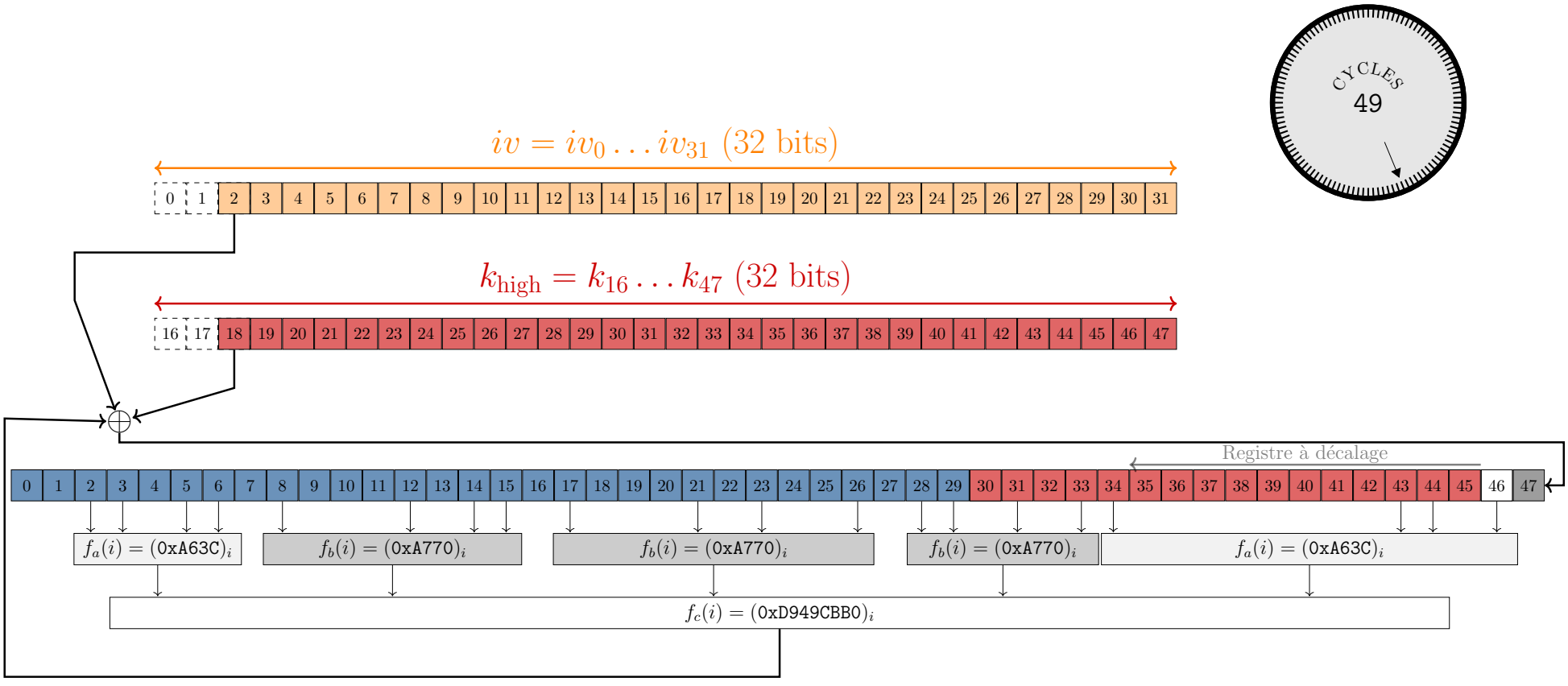
L'algorithme Hitag-2 : phase de randomisation



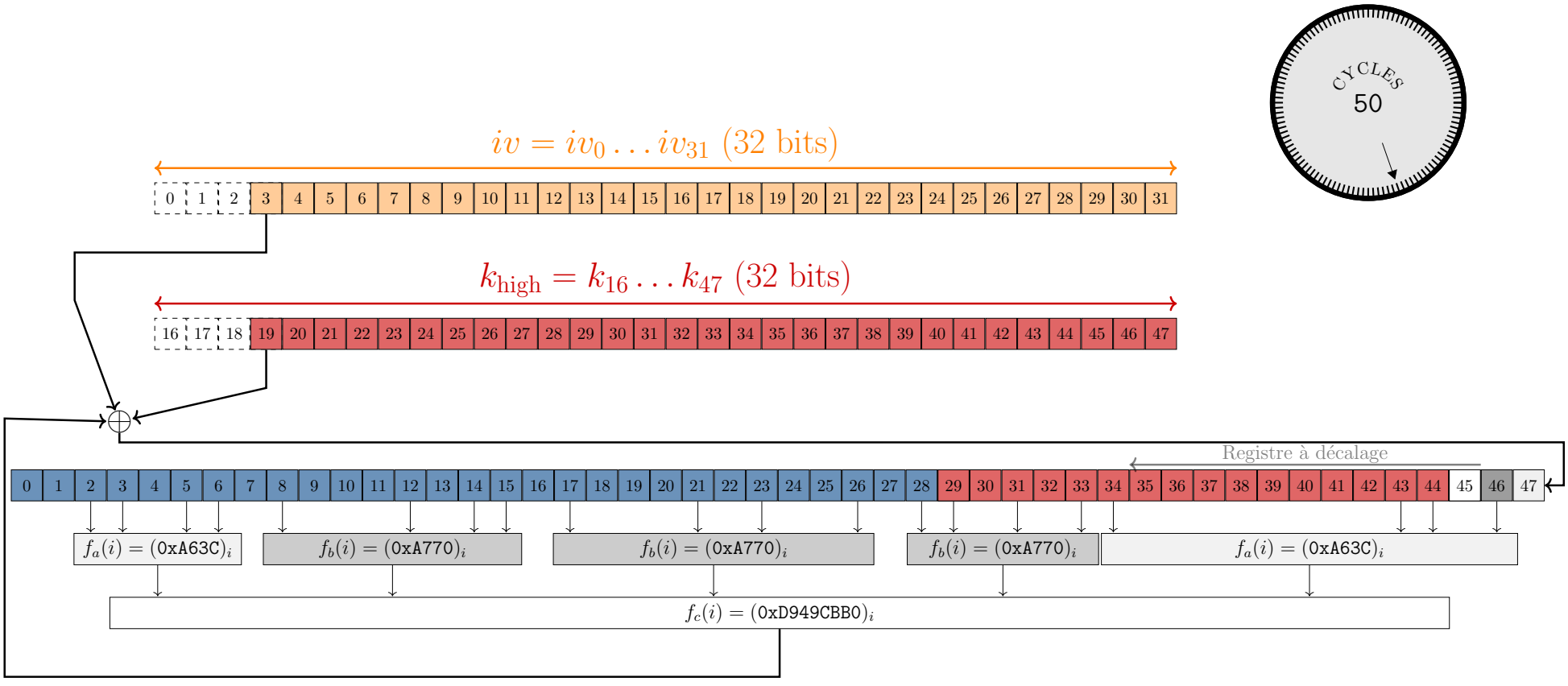
L'algorithme Hitag-2 : phase de randomisation



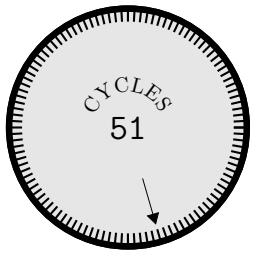
L'algorithme Hitag-2 : phase de randomisation

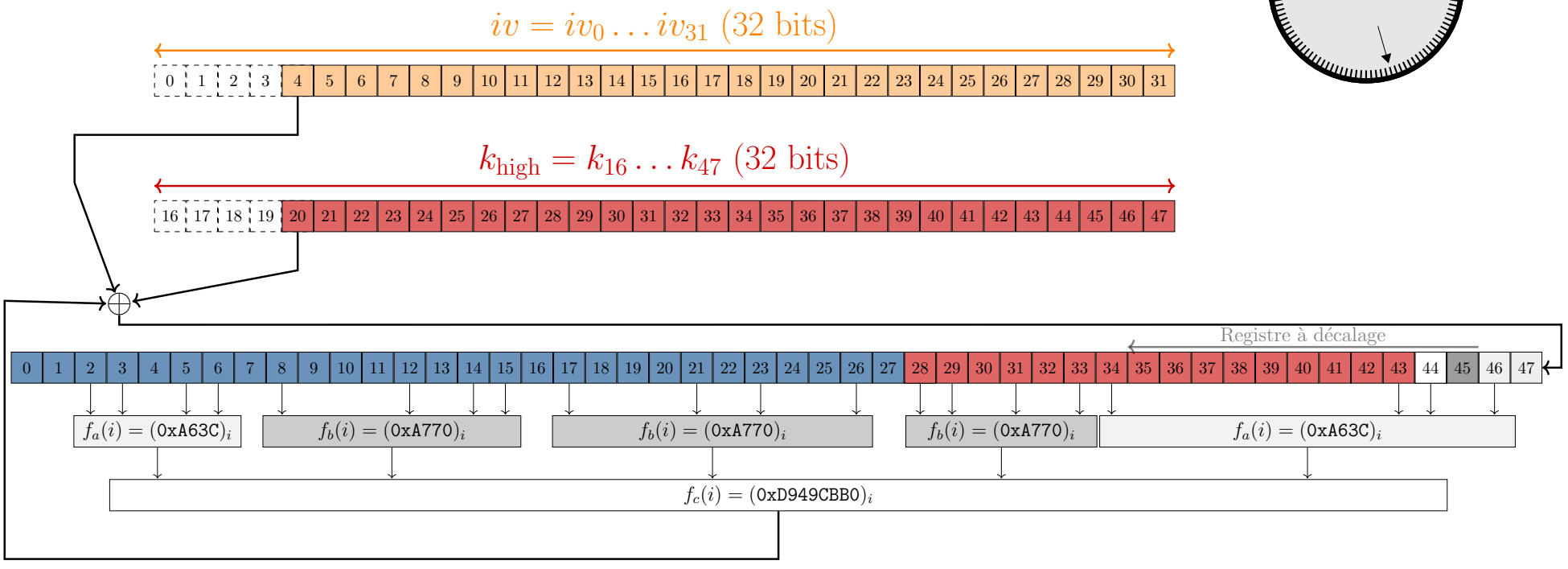


L'algorithme Hitag-2 : phase de randomisation

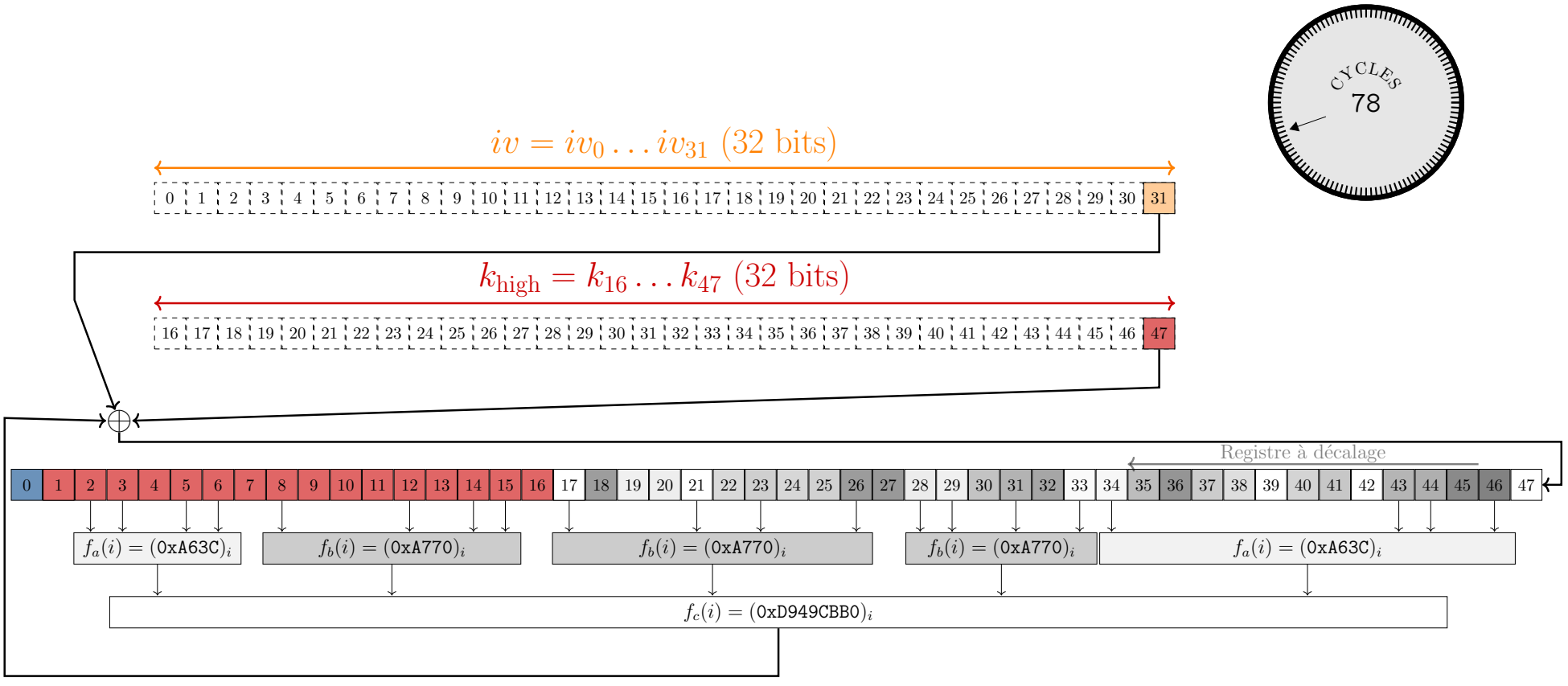


L'algorithme Hitag-2 : phase de randomisation

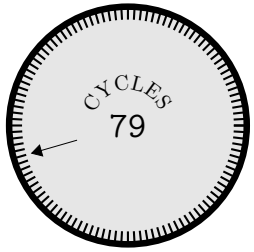




L'algorithme Hitag-2 : phase de randomisation



L'algorithme Hitag-2 : phase de randomisation

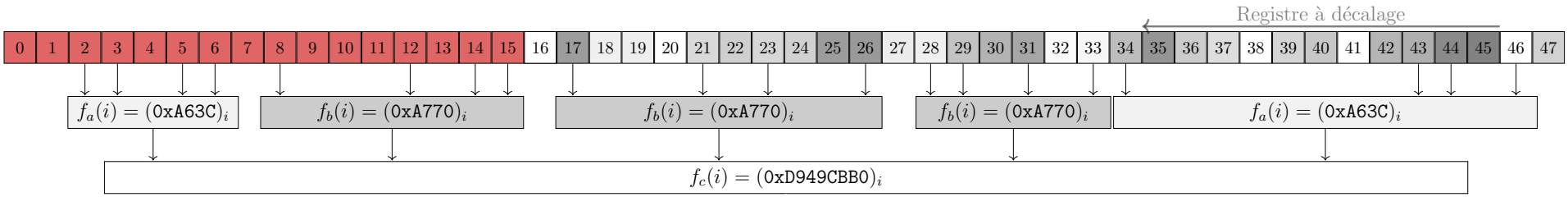


$iv = iv_0 \dots iv_{31}$ (32 bits)

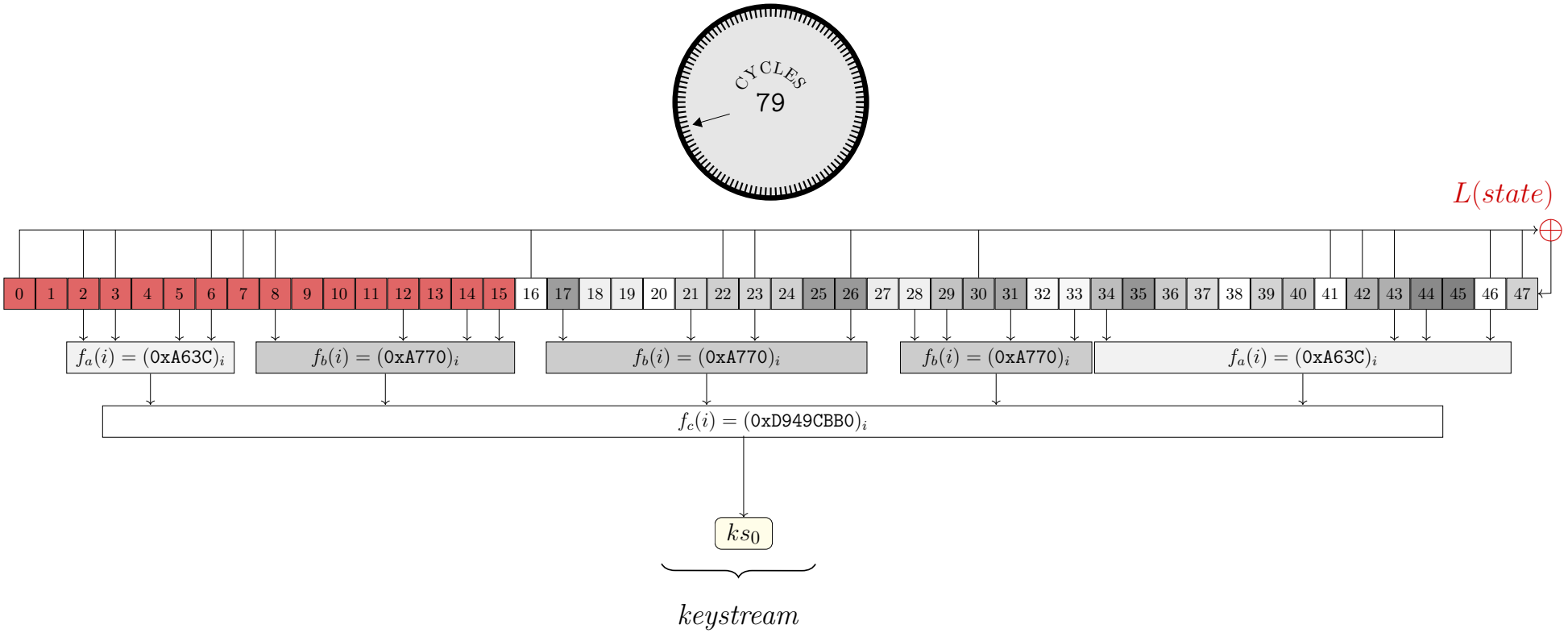
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

$k_{\text{high}} = k_{16} \dots k_{47}$ (32 bits)

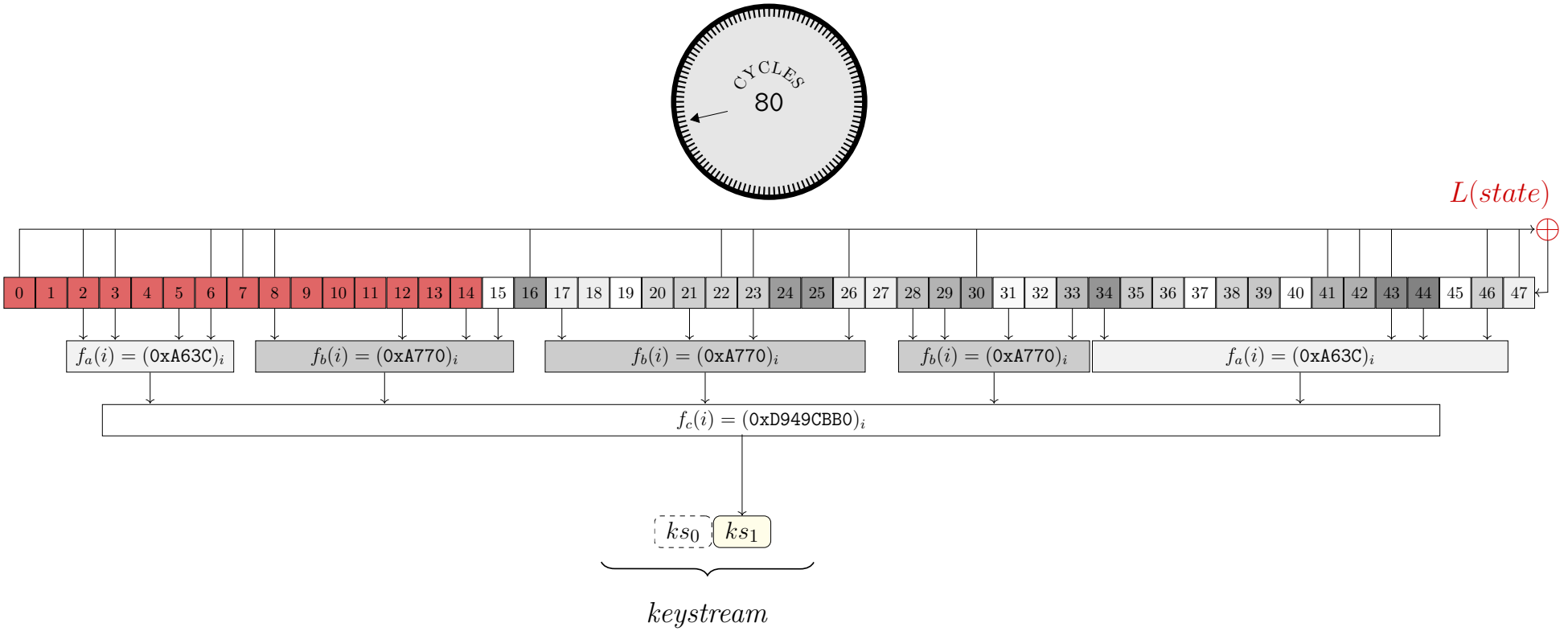
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47



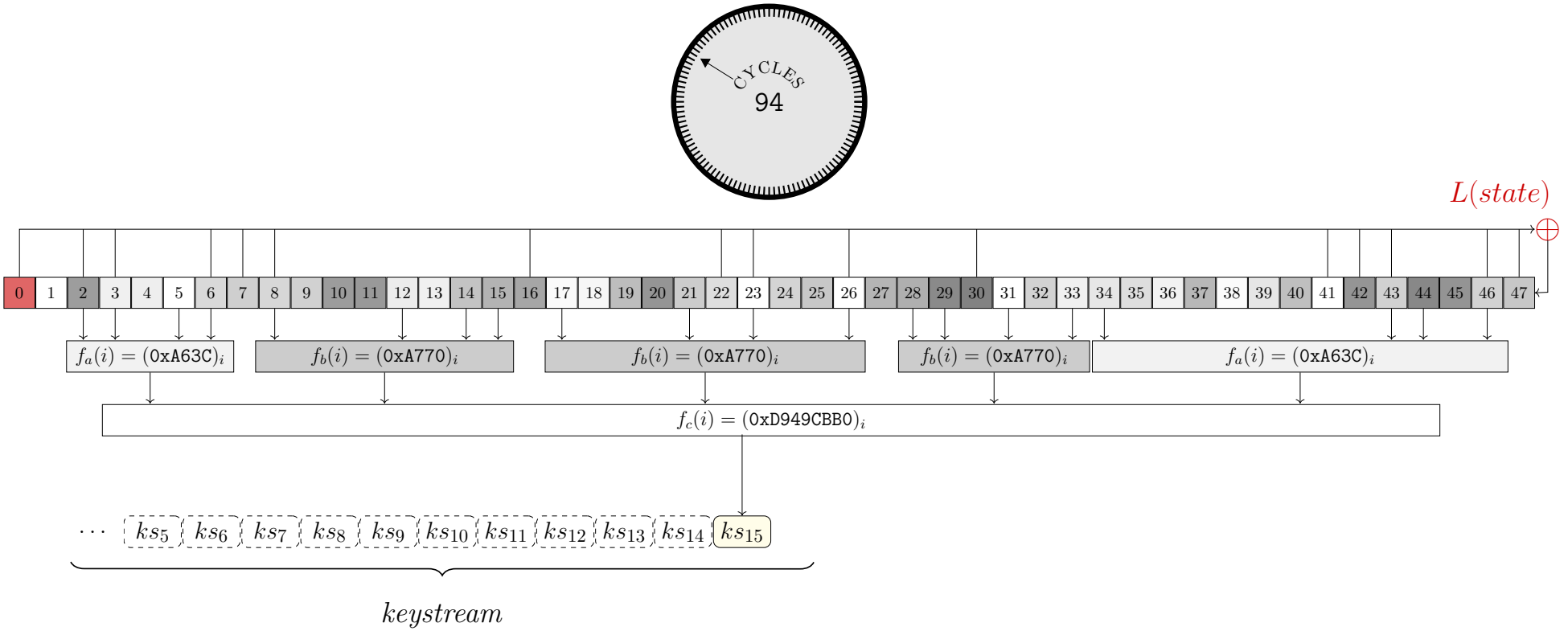
L'algorithme Hitag-2 : phase nominale



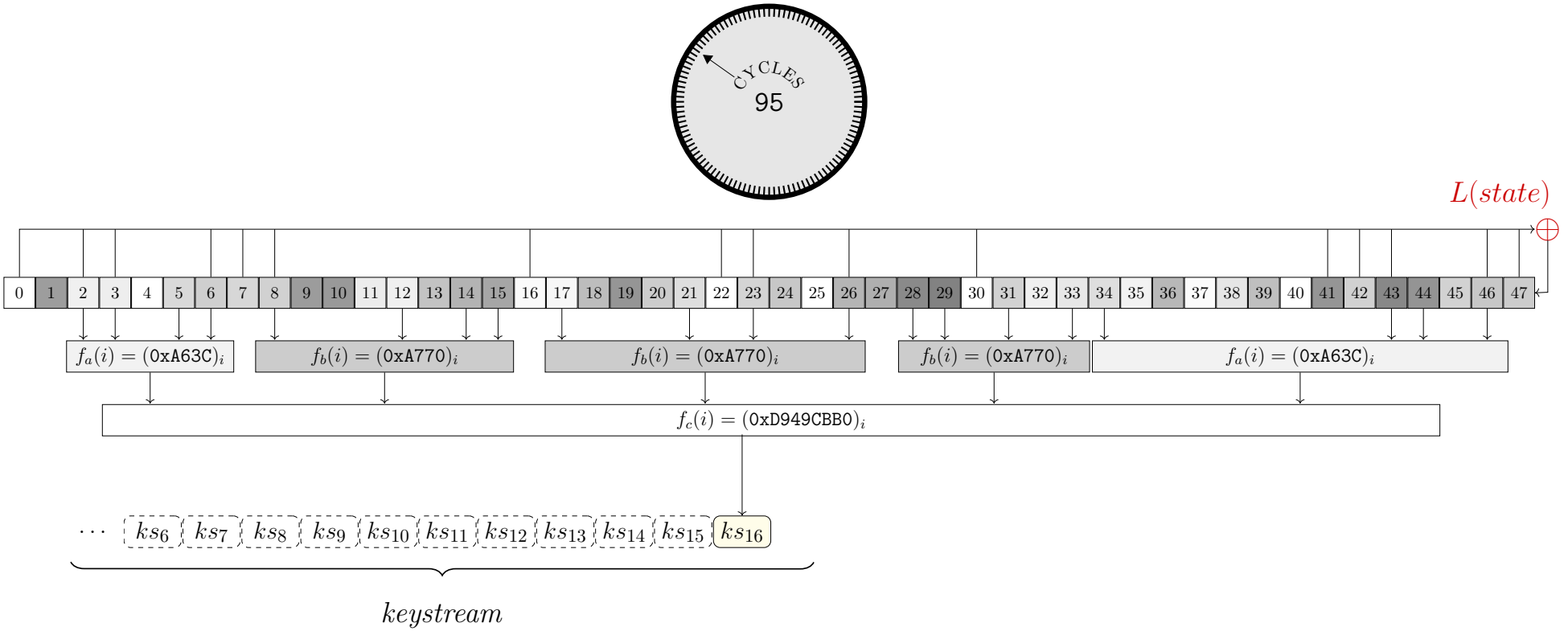
L'algorithme Hitag-2 : phase nominale



L'algorithme Hitag-2 : phase nominale



L'algorithme Hitag-2 : phase nominale



L'algorithme Hitag-2 : attaque par corrélation

- Introduite par l'article de Usenix 2016 :
  - Retrouve la clé avec 4 à 8 trames.
  - Réduit fortement l'espace de recherche des clés.
  - *Scoring* des candidats lié au *keystream* observé.

L'algorithme Hitag-2 : attaque par corrélation

- Introduite par l'article de Usenix 2016 :
  - Retrouve la clé avec 4 à 8 trames.
  - Réduit fortement l'espace de recherche des clés.
  - *Scoring* des candidats lié au *keystream* observé.
- Problème de CNTRH inconnu :
  - *A priori* à zéro en sortie d'usine.
  - Les auteurs proposent d'estimer l'âge du véhicule.

Implémentation de la cryptanalyse par corrélation

- Test sur des trames émulées.
  - Notre implémentation fonctionne.
  - La clé est retrouvée en quelques minutes.

Implémentation de la cryptanalyse par corrélation

- Test sur des trames émulées.
  - Notre implémentation fonctionne.
  - La clé est retrouvée en quelques minutes.
- Test sur des trames réelles (`CNTRH` inconnu).
  - Ne converge pas vers une bonne clé sur les trames décodées ...

Implémentation de la cryptanalyse par corrélation

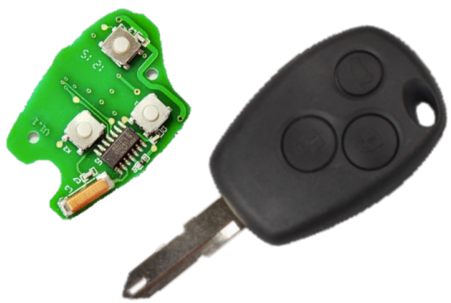
- Test sur des trames émulées.
  - Notre implémentation fonctionne.
  - La clé est retrouvée en quelques minutes.
- Test sur des trames réelles (`CNTRH` inconnu).
  - Ne converge pas vers une bonne clé sur les trames décodées ...
  - Nécessité d'un recalage cryptographique.

Le recalage cryptographique

- Comment faire ?
  - Accès au véhicule mais accès difficile à l'ECU.
  - Pas de NDA avec NXP donc pas de *datasheet* ni de SDK.

Le recalage cryptographique

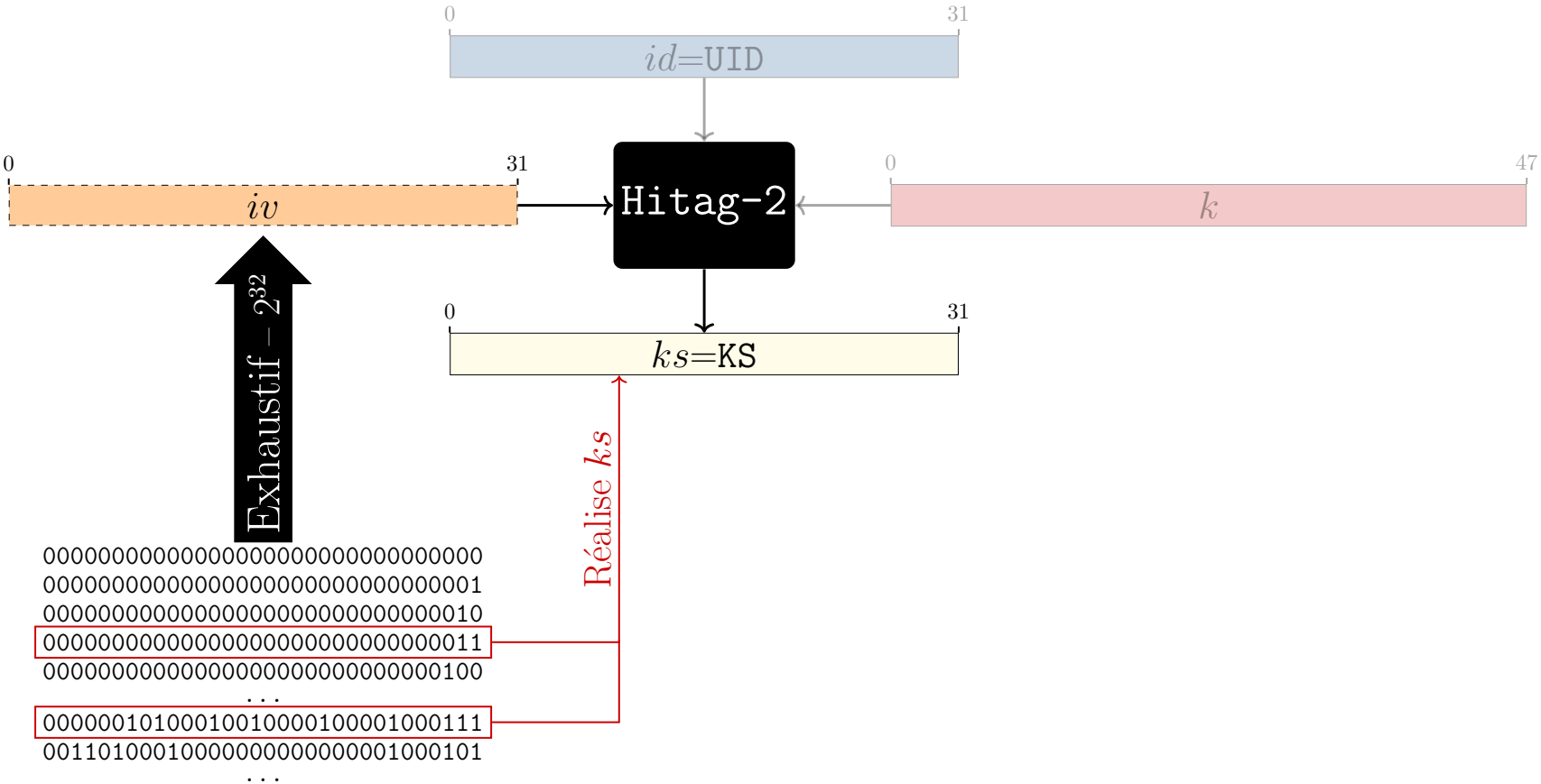
- Comment faire ?
  - Accès au véhicule mais accès difficile à l'ECU.
  - Pas de NDA avec NXP donc pas de *datasheet* ni de SDK.
- Accès à des clés vierges programmables contenant le PCF7946 !



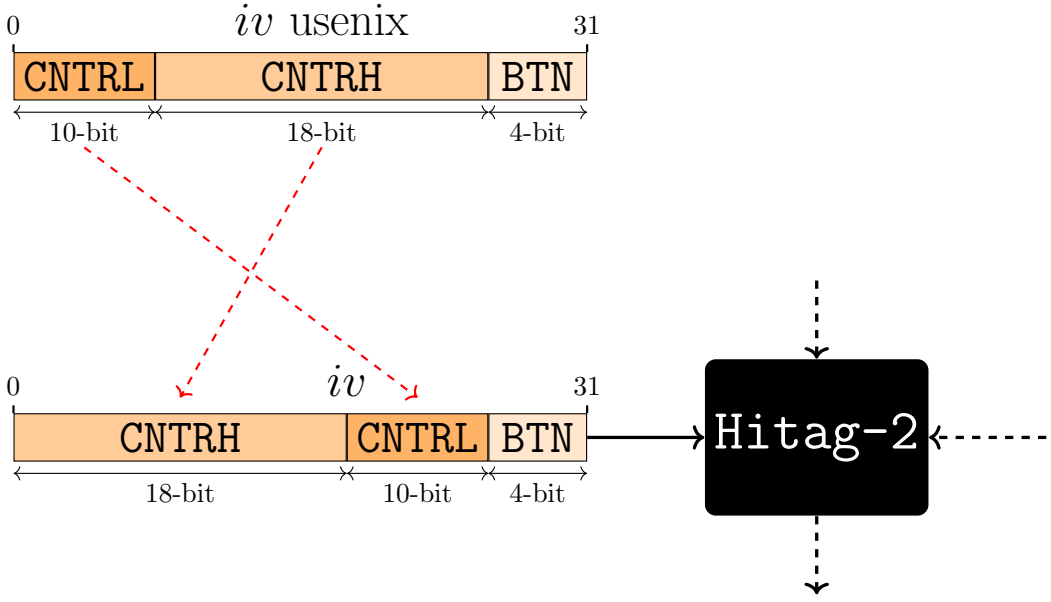
  - Elles utilisent la clé usine par défaut `0x4f4e4d494b52`.

Recherche du format de *iv*

- Recherche exhaustive d'un *pattern* pour les 2^{32} *iv* qui réalise *ks* à *id* et *k* fixes.

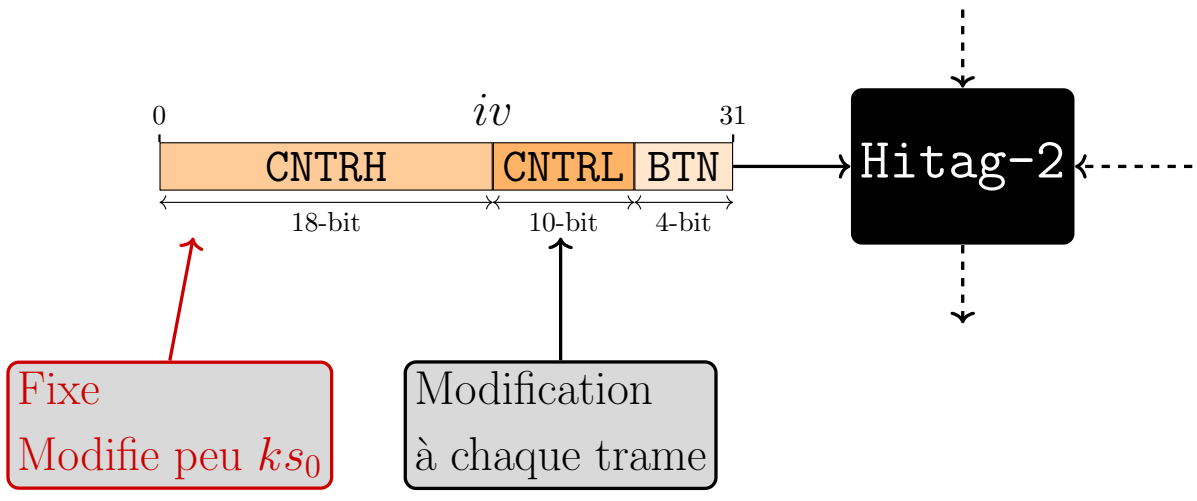


Différences trouvées



Différences trouvées

- Explique pourquoi la cryptanalyse par corrélation fonctionne mal.



Découverte d'une contremesure ECU

ECU

UID	Clé	Compteur	Bouton
UID1	K1	10	0,1,5,6
UID2	K2	200	0,1,2
...	...	...	...

Découverte d'une contremesure ECU

ECU

UID	Clé	Compteur	Bouton
UID1	K1	**10**	0,1,5,6
UID2	K2	200	0,1,2
...	...	...	...

CNTR1 = 9 < 10

Découverte d'une contremesure ECU

- Resynchronisation en champ proche à 125 KHz lors du contact.

UID1, CNTRL1 = 9, BTN1, AUTH1

ECU

UID	Clé	Compteur	Bouton
UID1	K1	10	0,1,5,6
UID2	K2	200	0,1,2
...	...	...	...

CNTR1 = 9 < 10

ALÉATOIRE

CNTRH' | CNTRL'

18-bit 10-bit

Resynchronisation ECU

Recherche exhaustive optimisée

- Nécessite deux triplets (id, iv, ks) :
  - Recherche parmi 2^{48} clés celle qui réalise les deux *keystreams*.
  - Implémentation d'un *brute-forcer* optimisé et parallélisé sur CPU et GPU en OpenCL.

Recherche exhaustive optimisée

- Nécessite deux triplets (id, iv, ks) :
  - Recherche parmi 2^{48} clés celle qui réalise les deux *keystreams*.
  - Implémentation d'un *brute-forcer* optimisé et parallélisé sur CPU et GPU en OpenCL.

- Testé sur Amazon EC2 :

Plateforme	Temps
GeForce `GTX 780Ti`	18 heures
Instance Amazon EC2†	45 minutes
3 instances Amazon EC2†	15 minutes

†`p2.16xlarge` : 16 Tesla K80, 128 CPU

Recherche exhaustive optimisée

- Nécessite deux triplets (id, iv, ks) :
  - Recherche parmi 2^{48} clés celle qui réalise les deux *keystreams*.
  - Implémentation d'un *brute-forcer* optimisé et parallélisé sur CPU et GPU en OpenCL.

- Testé sur Amazon EC2 :

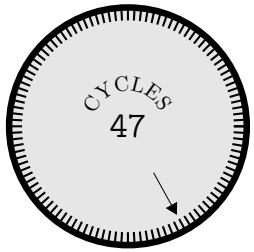
Plateforme	Temps
GeForce `GTX 780Ti`	18 heures
Instance Amazon EC2†	45 minutes
3 instances Amazon EC2†	15 minutes

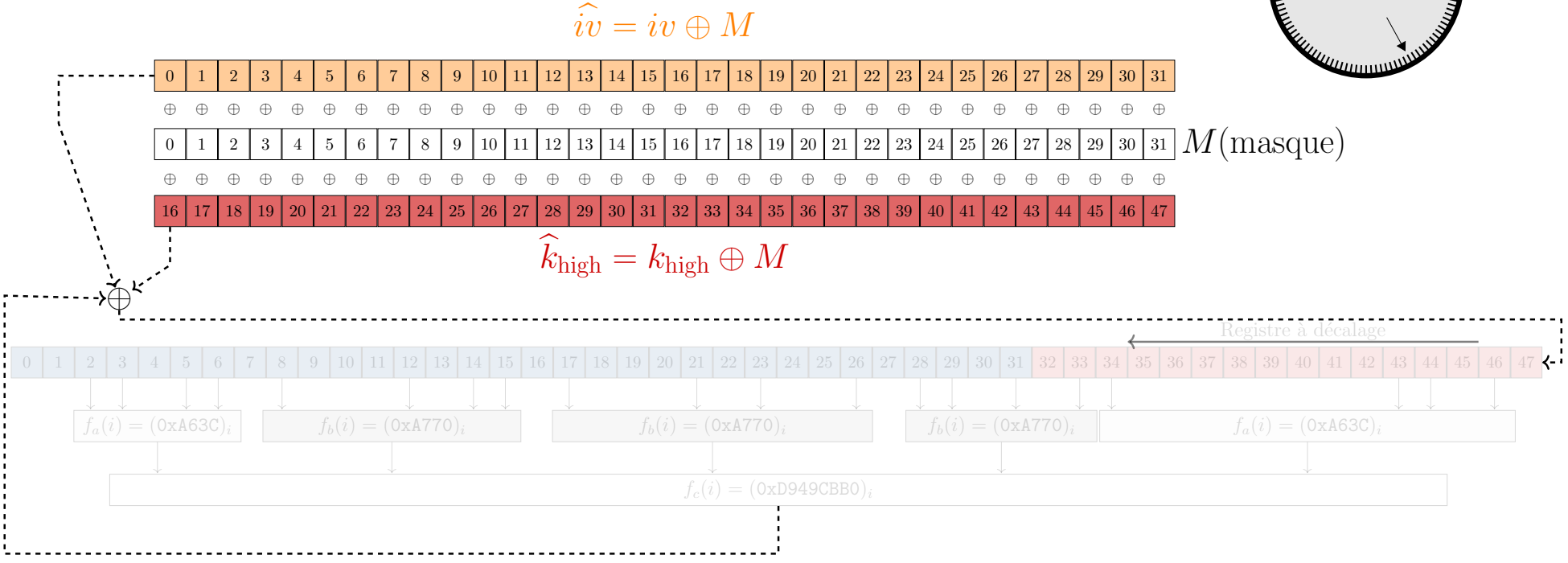
†`p2.16xlarge` : 16 Tesla K80, 128 CPU

- *Quid* de la partie inconnue `CNTRH` ?

Clés Hitag-2 équivalentes

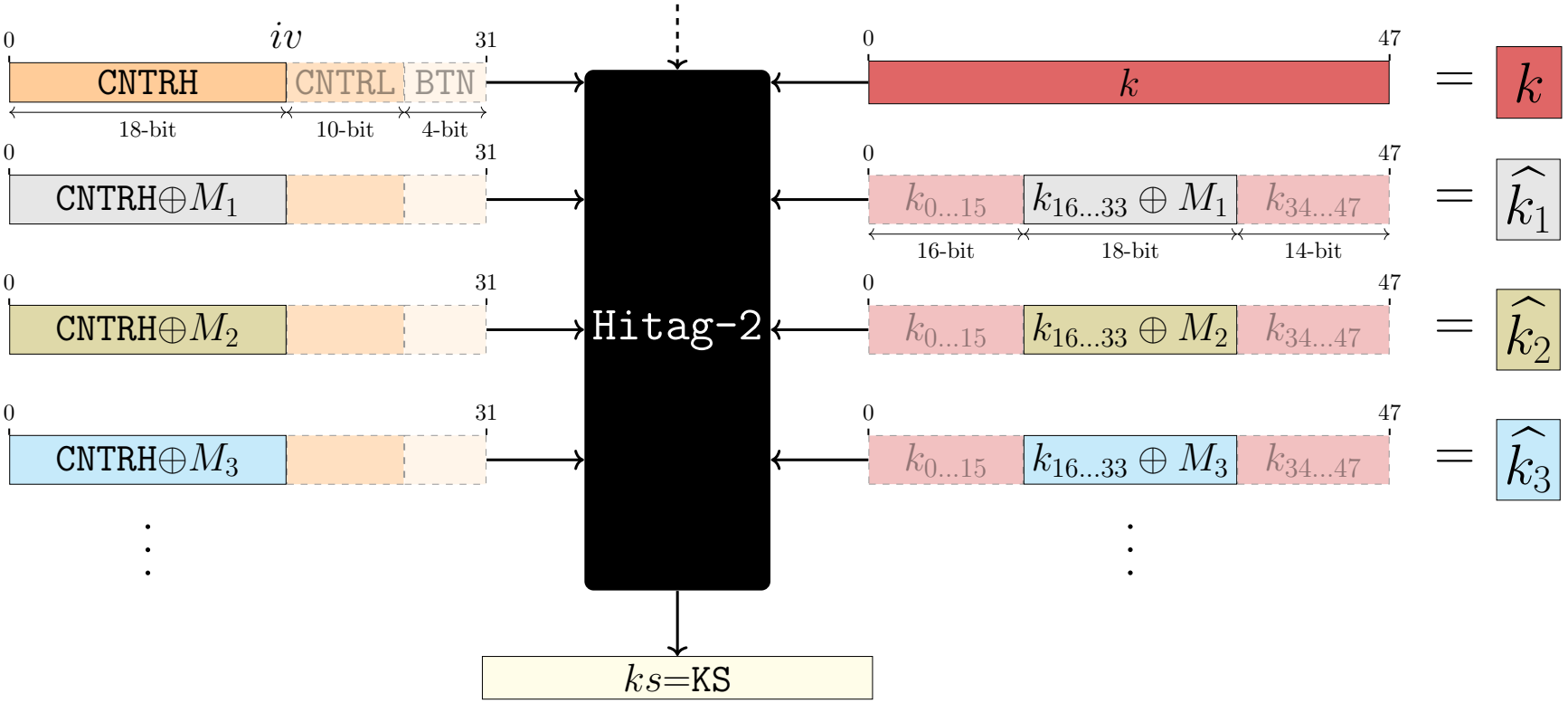
- Compensation par masquage durant la phase de randomisation.





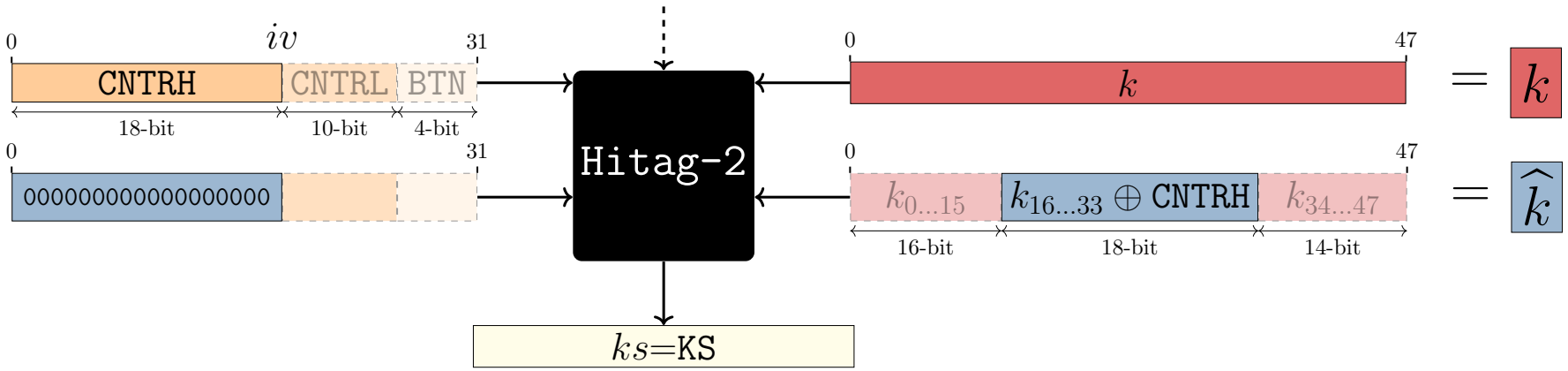
Clés Hitag-2 équivalentes

- Existence de multiples clés équivalentes produisant le même *keystream* à iv masqué.



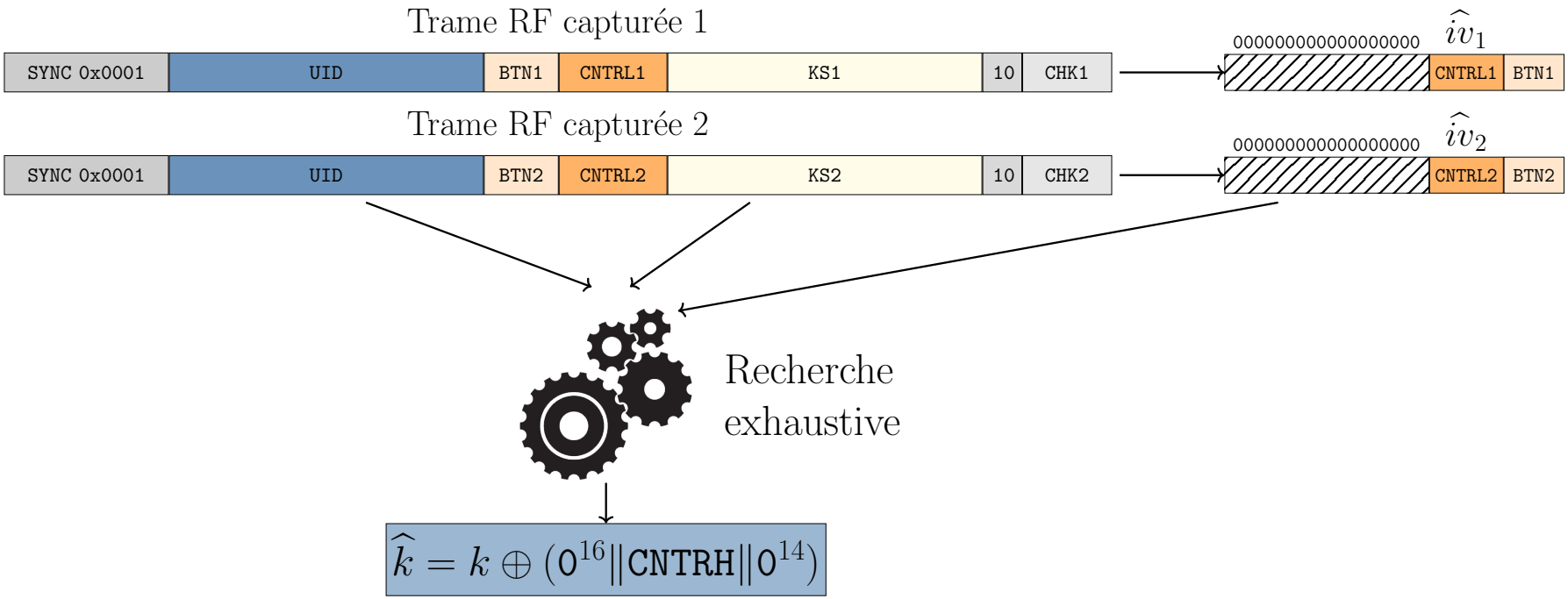
Clés Hitag-2 équivalentes

- Existence de multiples clés équivalentes produisant le même *keystream* à iv masqué.
- Cas particulier où le masque est CNTRH.



Clés Hitag-2 équivalentes

- Existence de multiples clés équivalentes produisant le même *keystream* à iv masqué.
- Une recherche exhaustive avec des $\widehat{iv}$ équivalents produit une clé équivalente $\widehat{k}$ masquée avec CNTRH.

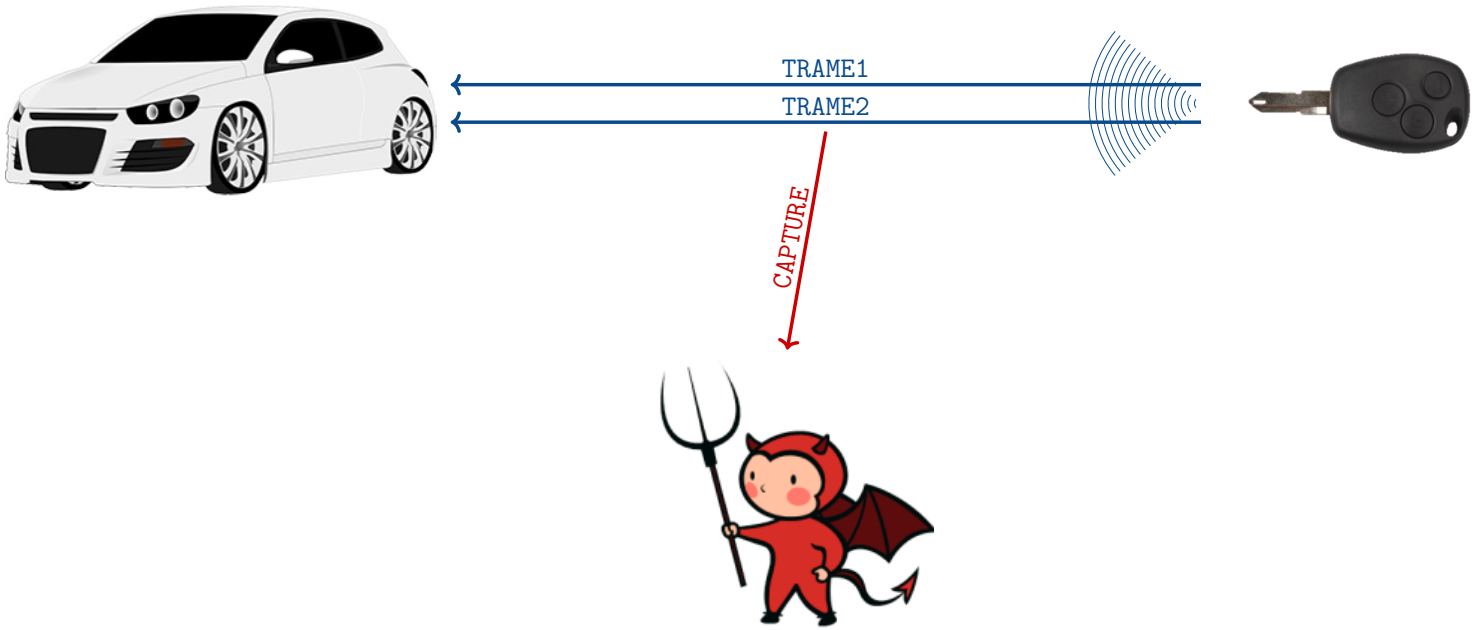


Clés Hitag-2 équivalentes

- Existence de multiples clés équivalentes produisant le même *keystream* à iv masqué.
- Une recherche exhaustive avec des $\widehat{iv}$ équivalents produit une clé équivalente $\widehat{k}$ masquée avec CNTRH.
- Nul besoin de retrouver la vraie clé k pour forger des trames valides !

Nouvelles attaques 1/2

- **Sans** resynchronisation ECU.



Nouvelles attaques 1/2

- **Sans** resynchronisation ECU.

$\widehat{k}$ = Recherche exhaustive 2^{48}
$\Rightarrow$ 15 minutes EC2

Nouvelles attaques 1/2

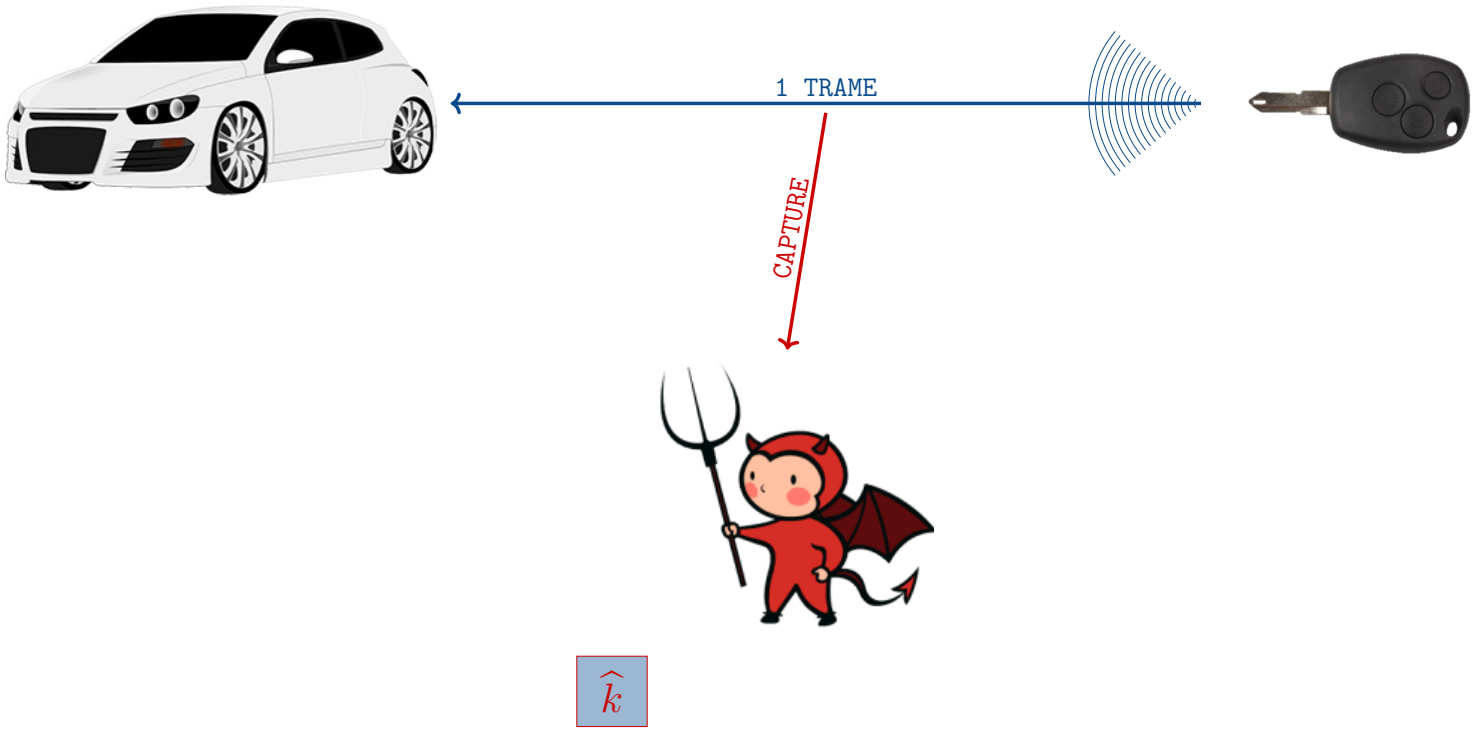
- **Sans** resynchronisation ECU.

$\widehat{k}$

$\leq (1024 - \texttt{CNTRL})$ trames OK
$> (1024 - \texttt{CNTRL}) \Rightarrow$ incrémentation

Nouvelles attaques 2/2

- **Avec** resynchronisation ECU.



Nouvelles attaques 2/2

- **Avec** resynchronisation ECU.

$\widehat{k}$ ⇢ $\widehat{k'}$

$\widehat{k'}$ = Recherche echaustive 2^{18}
⇒ 15 minutes sur PC

Nouvelles attaques 2/2

- **Avec** resynchronisation ECU.

$\widehat{k'}$

$\leq (1024 - \texttt{CNTRL})$ trames OK
$> (1024 - \texttt{CNTRL}) \Rightarrow$ incrémentation

Conclusion

- Résultats :
  - Différentes implémentations RKE `Hitag-2`.
  - Contremesure par resynchronisation avec l'ECU.
  - Coût de l'attaque $= 10 + 90 + 45$ €.
  - 2 trames RF, +1 avec la contremesure ECU.

Conclusion

- Résultats :
  - Différentes implémentations RKE `Hitag-2`.
  - Contremesure par resynchronisation avec l'ECU.
  - Coût de l'attaque $= 10 + 90 + 45$ €.
  - 2 trames RF, +1 avec la contremesure ECU.

- Cryptographie propriétaire obsolète et cassée à proscrire.