Rétro-ingénierie matérielle pour les reversers logiciels : cas d'un DD externe chiffré

Joffrey Czarny & Raphaël Rigo / AGI / TX5IT

2015-06-03 / SSTIC

AIRBUS GROUP


DataLocker


APRICORN
FIPS
Level 2 Validated
140-2
USB 3.0


LOCKDOWN


Bull


CORSAIR

Introduction

Pourquoi étudier des disques chiffrés ?

- Demande initiale d'audit au sein d'Airbus Group.
- De précédents travaux ont démontré des lacunes.
- Découvrir l'analyse de produits électroniques avec microcontrôleurs.

Exemples d'*epic fail* sur ce type de périphériques

- Kingston/SanDisk FIPS 140-2 : paquet magique pour débloquer (2010)
- Corsair Padlock : données non chiffrées, accessibles sans PIN (2008)
- Corsair Padlock 2 : PIN brute-forçable (2010)

Le but *in fine*

- Analyser l'efficience d'un disque chiffré à protéger les données.
  $\Longrightarrow$ Valider l'implémentation de la sécurité et de la cryptographie au sein d'un disque chiffré.

Introduction

Objectifs de la présentation :

- Décrire l'étude d'un disque externe sécurisé ;
  - expliquer la démarche en détails ;
  - montrer nos différents échecs ;
  - donner des pistes pour continuer l'étude.

Cas d'étude présenté : boîtier Zalman ZM-VE400 :

- disque dur interchangeable ;
- chiffrement AES-256-XTS optionnel (clavier physique) ;
- peut "monter" des ISO en tant que lecteur optique USB.

Contexte, premiers éléments

Vérifications génériques

- vérifier la crypto de base :
  - mode ECB ?
  - tests statistiques corrects ?
  - clé fixe ?
- tests un peu plus avancés (clé constante ?) :
  - vérifier que le même PIN, sur deux boîtiers, donne un chiffrement différent ;
  - vérifier que le même PIN, sur le même boîtier, donne un chiffrement différent.

Constats sur le VE400

- chiffrement indépendant du boîtier : **un disque chiffré placé dans un boîtier Zalman neuf est accessible une fois le bon code PIN entré** ;
- Activation du chiffrement $\Longrightarrow$ 10 secteurs à la fin du disque :
  - inaccessibles une fois le chiffrement actif,
  - *blob* de 768 octets à forte entropie, en double ;
- mises à jour du *firmware* chiffrées : *reverse* impossible.

Suite de l'étude

Résultat : échec de conception

L'ensemble des informations nécessaires au déchiffrement est sur le disque.
$\Longrightarrow$ Attaque efficace possible (*bruteforce*, récupération de clé).

Nouvel objectif global

Comprendre le format et les données du bloc situé à la fin du disque, pour implémenter une attaque hors-ligne.

Comment ?

En essayant d'accéder au *firmware* ou en analysant les communications.

Analyse du matériel

Réalisé suivant différentes étapes :

Analyse du PCB (carte électronique)

- identification des composants
- identification des pistes et *vias* (trous)

$\Longrightarrow$ Permet l'obtention d'une vision logique

Étude des flash

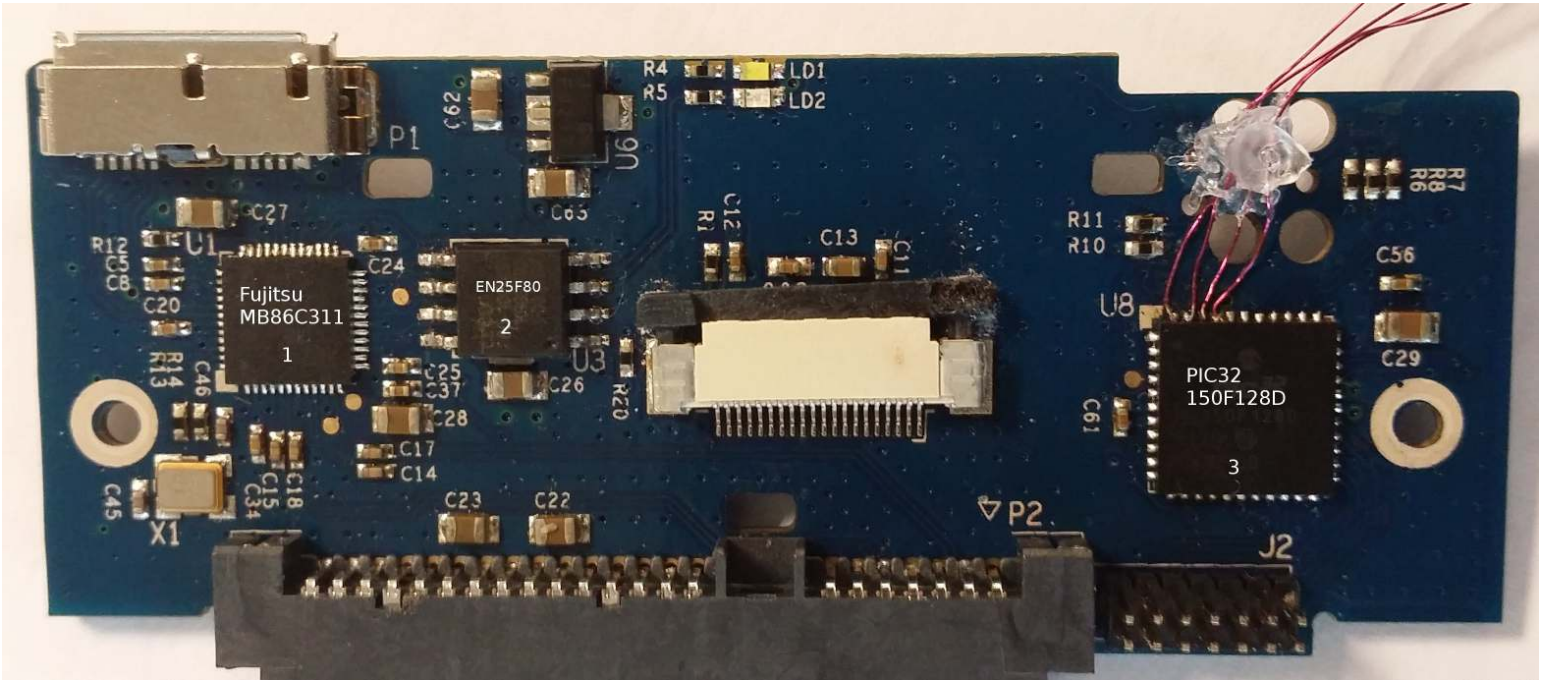
- identification des bus de communication
- récupération du contenu des flash

$\Longrightarrow$ Permet l'analyse du contenu des flash

PCB : identification des composants 1/2

Face avant du PCB

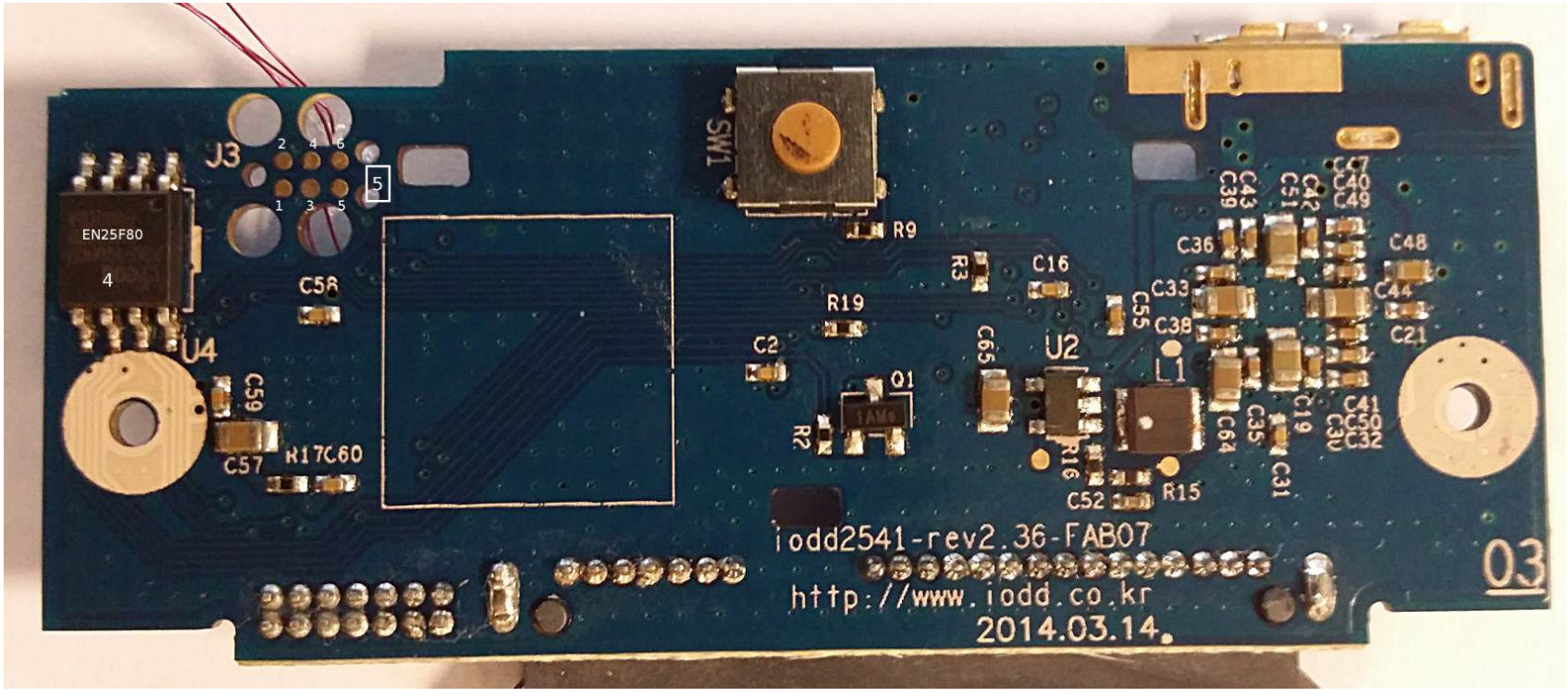
- *System on Chip* (SoC) Fujitsu MB86C311 USB3-SATA (ARM)
- Flash EN25F80
- Microcontrôleur PIC32MX 150F128D (MIPS)



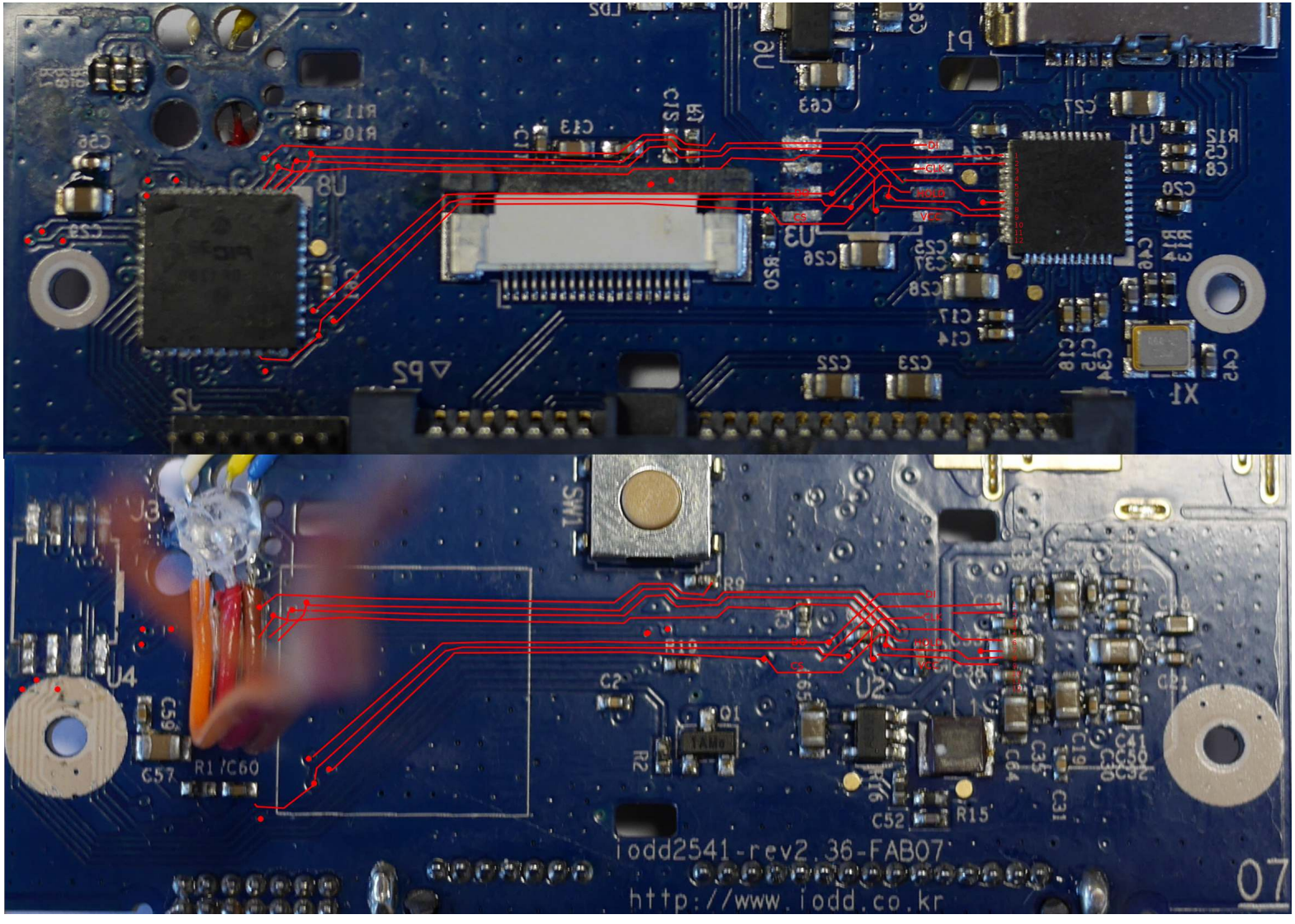
PCB : identification des composants 2/2

Face arrière du PCB

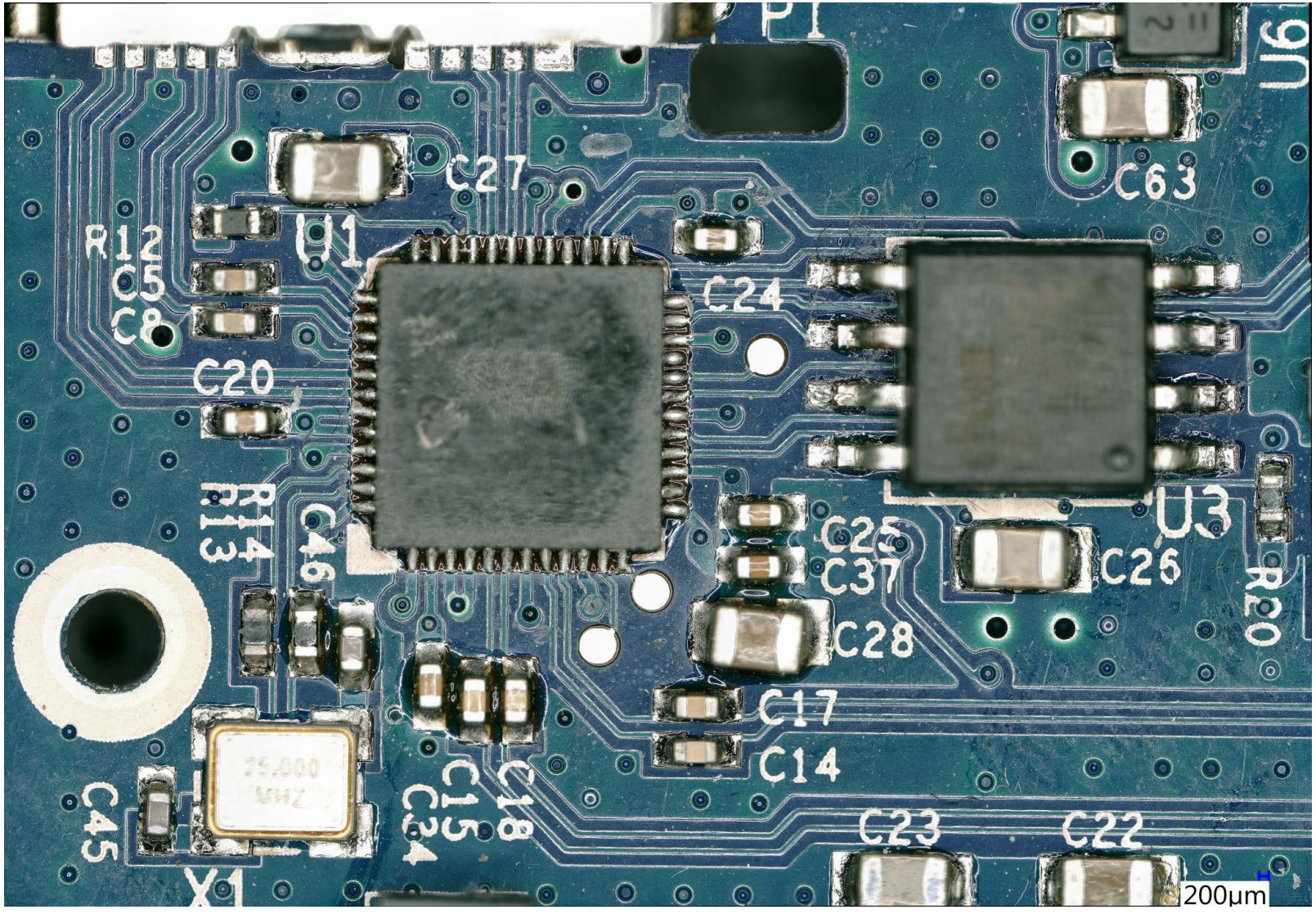
- Flash EN25F80



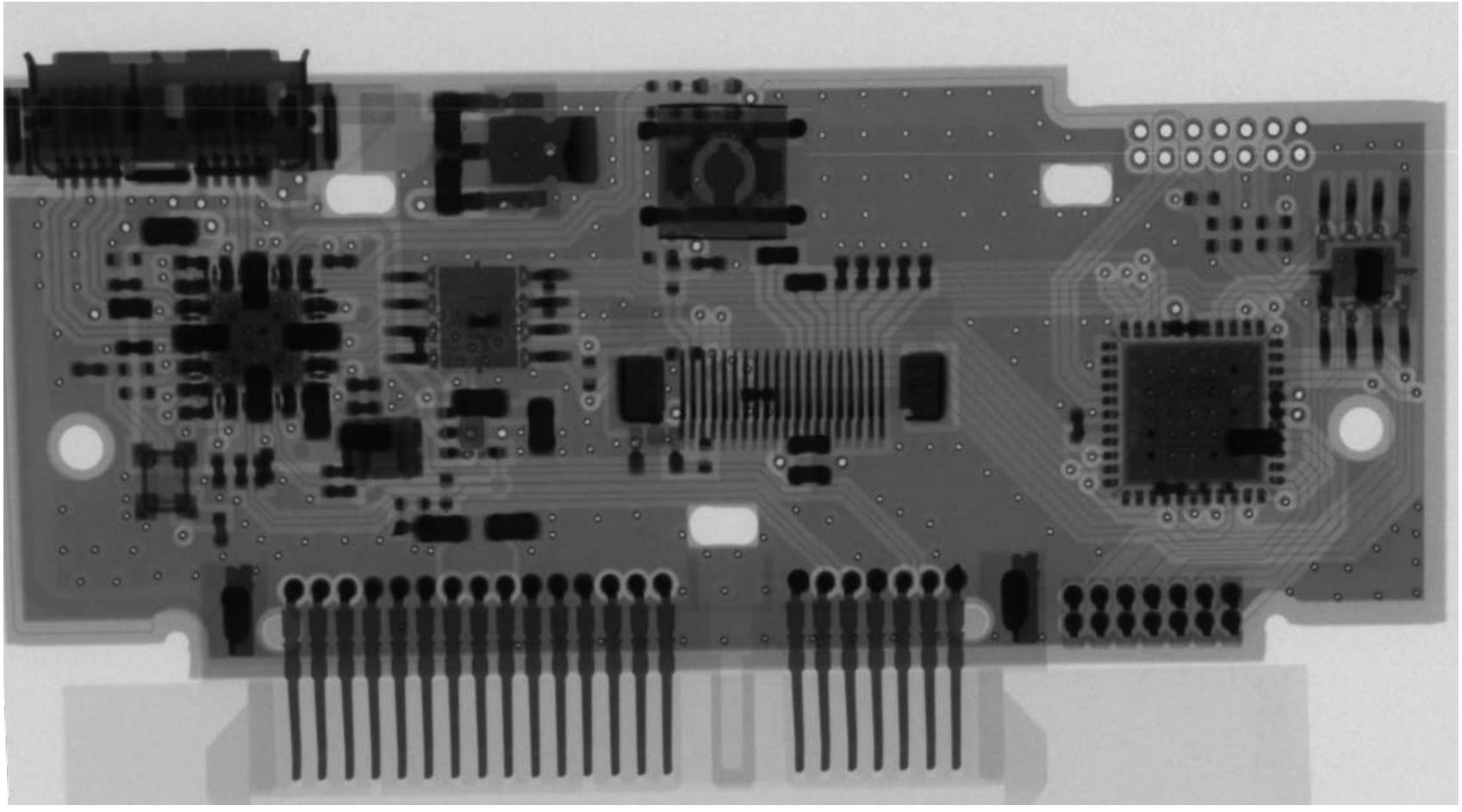
PCB : analyse des traces (1/4)



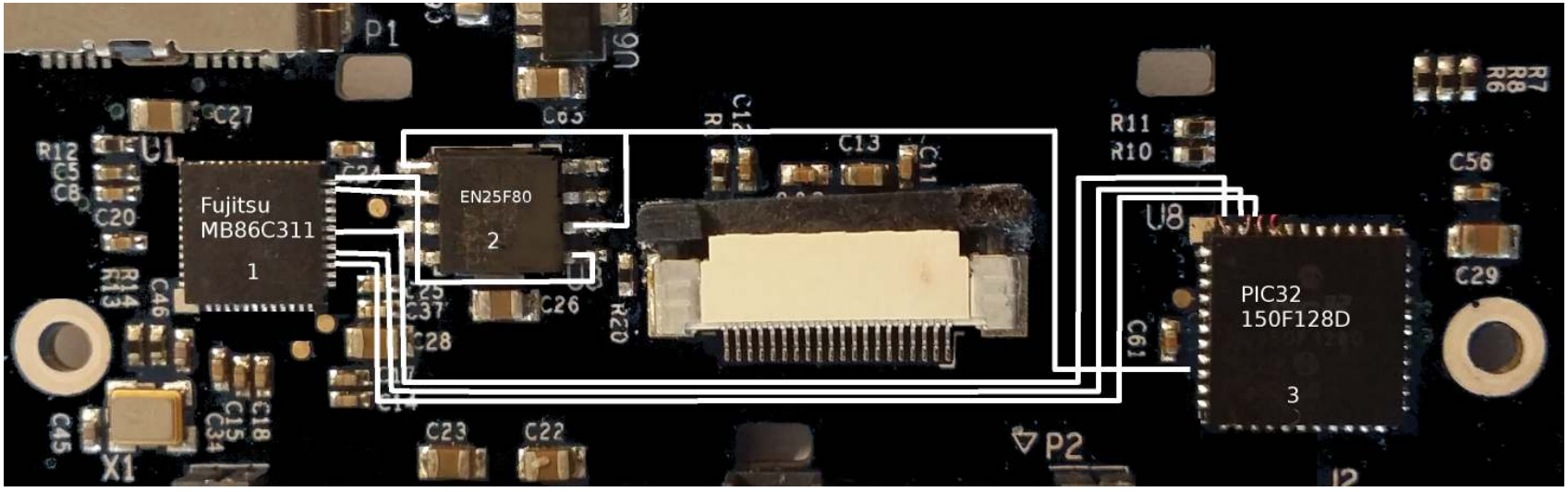
PCB : analyse des traces (2/4)



PCB : analyse des traces (3/4)



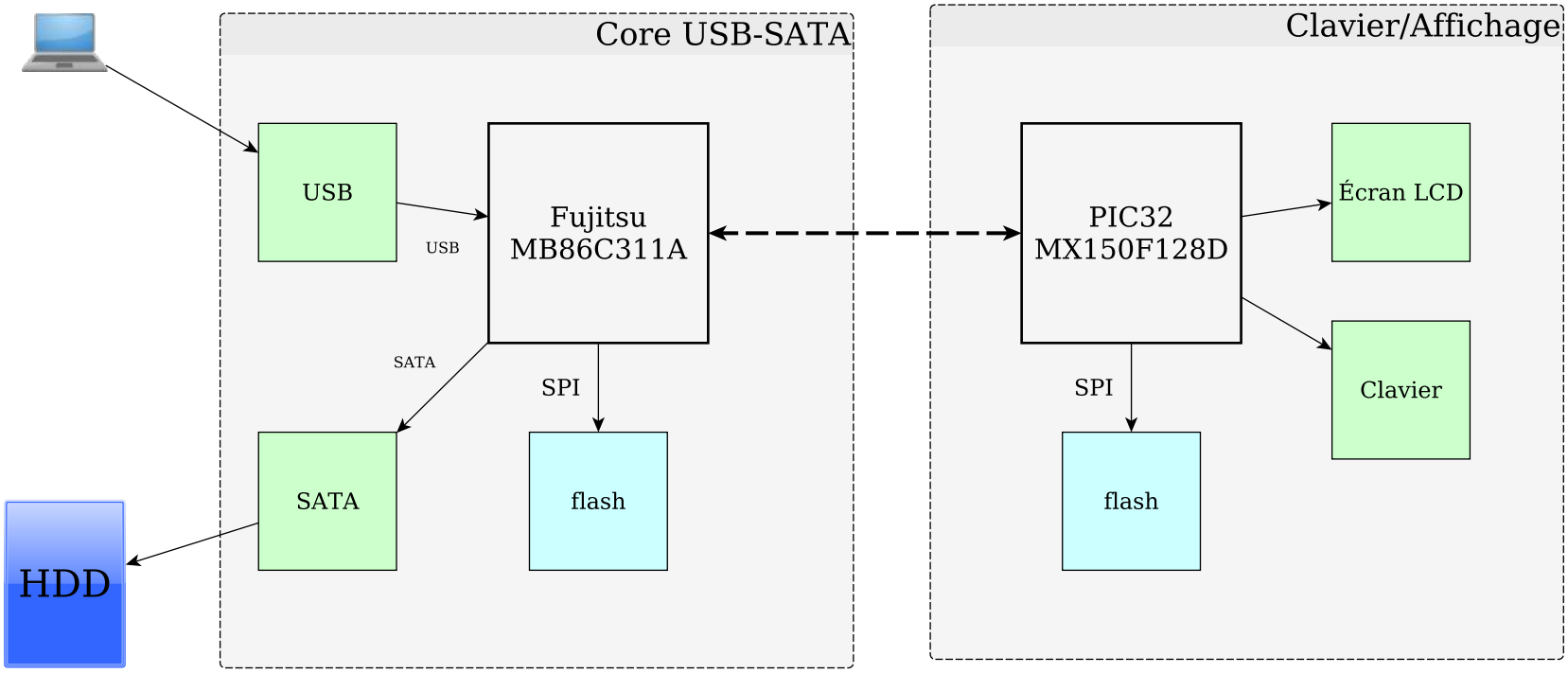
PCB : analyse des traces (4/4)



Au final :

- une flash dédiée pour le contrôleur USB-SATA (SoC)
- une flash dédiée pour le PIC32
- un lien entre le SoC et le PIC, partagé avec la flash du SoC

PCB : vision logique



Que contiennent les flash ?

Peut-être le code en clair ?

$\Longrightarrow$ Récupérons leur contenu.

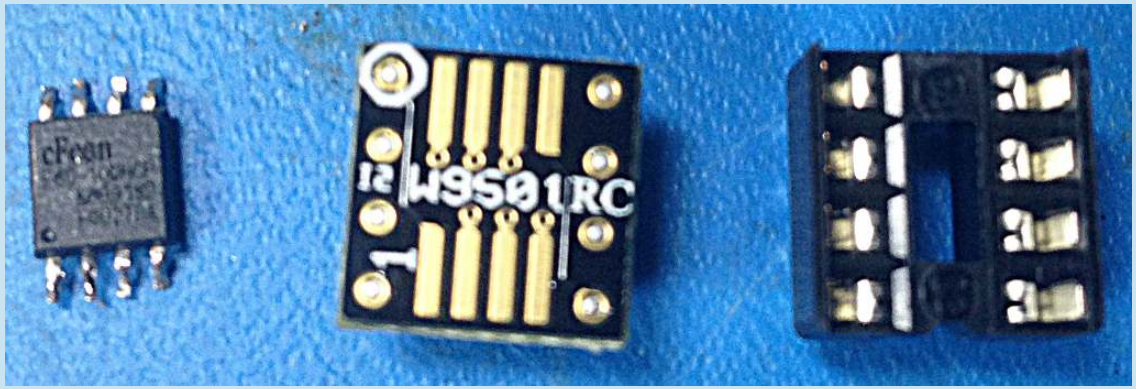
Récupération du contenu des flash (1/2)

SPI : *Serial Peripheral Interface*

- communication série synchrone ;
- spécifie 4 signaux logiques :
  - SCLK : Serial Clock
  - MOSI : Master Output, Slave Input
  - MISO : Master Input, Slave Output
  - SS : Slave Select

Lire le contenu

- nécessite de déssouder la flash
- utilisation d'un support SOIC↔DIP pour garder la board fonctionnelle



Récupération du contenu des flash (2/2)

L'outillage pour parler SPI

- GoodFET avec `goodfet.spiflash` (recommandé)
- Bus Pirate *via* SPI port
- RaspberryPI avec spi dev

Contenu des flash

Contrôleur USB-SATA :

- des données de configuration en clair
- le code, chiffré

Microcontrôleur PIC32 :

- une police de caractères
- le code, chiffré

Résultats

Échec de l'accès au code

Tout le code est chiffré, on ne peut donc pas reverser les firmwares.

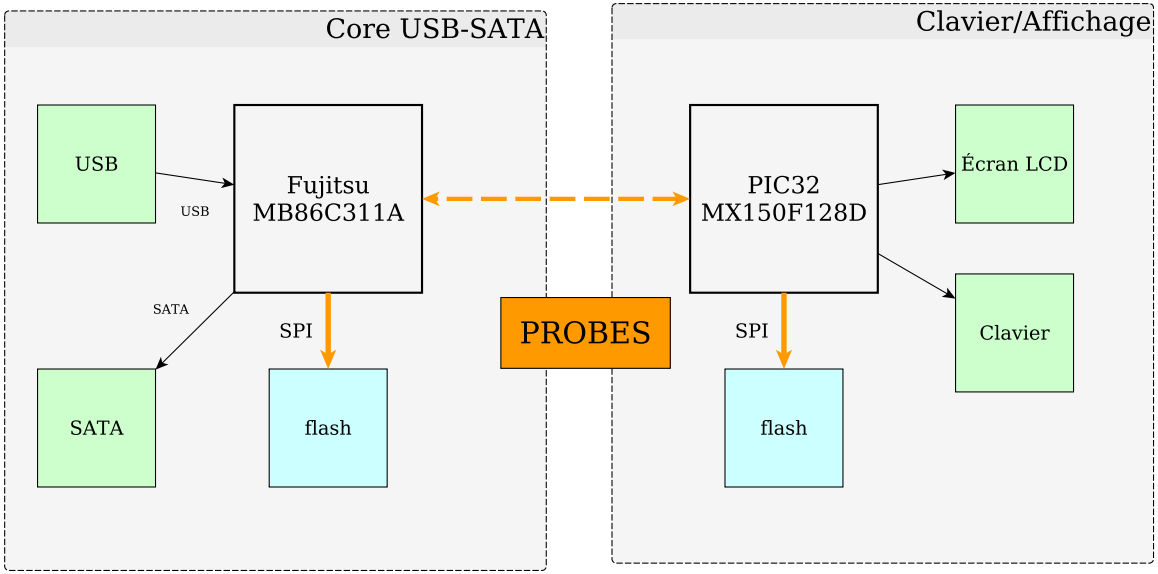
Que faire ?

Comme pour des échanges réseau, on va analyser les communications en boîte noire.

Comment ?

En utilisant un analyseur logique afin de capturer les communications.

Matériel et placement des sondes



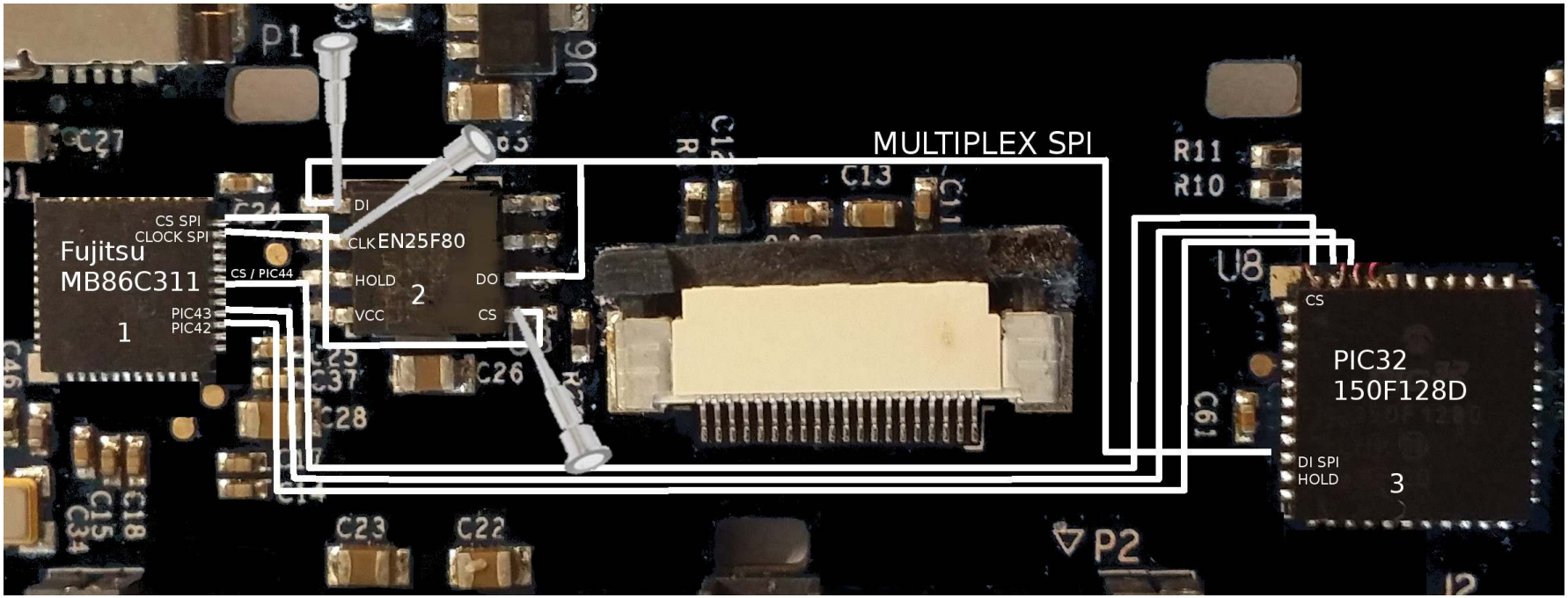
Comment ?

- analyseur logique Saleae Logic Pro 16

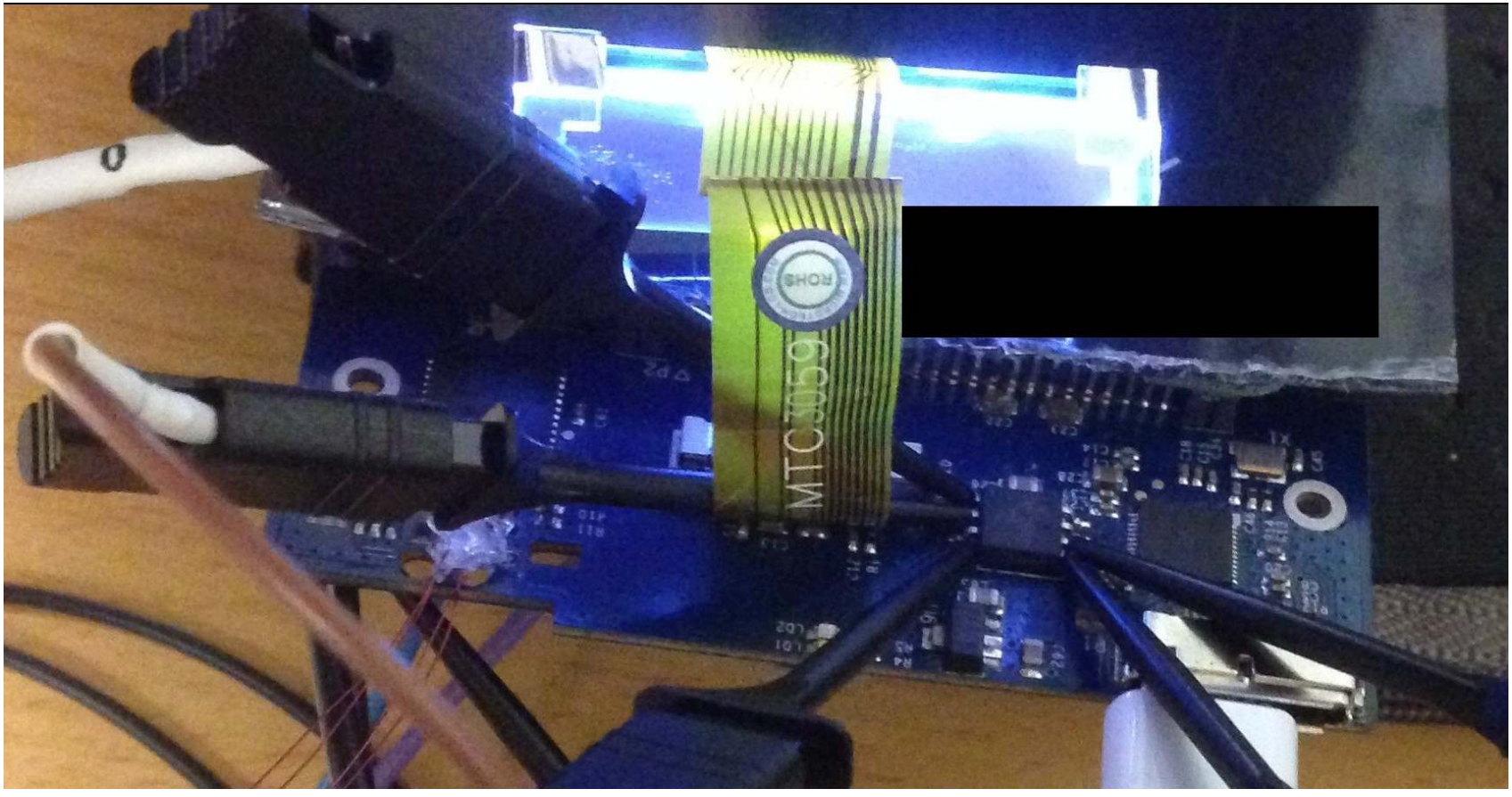
Analyseur logique ?

- "oscilloscope" pour signaux numériques
- multiples voies, souvent avec décodage de protocoles (voir [4])
- important : vitesse d'échantillonage (recommandée : 4x l'horloge [5])

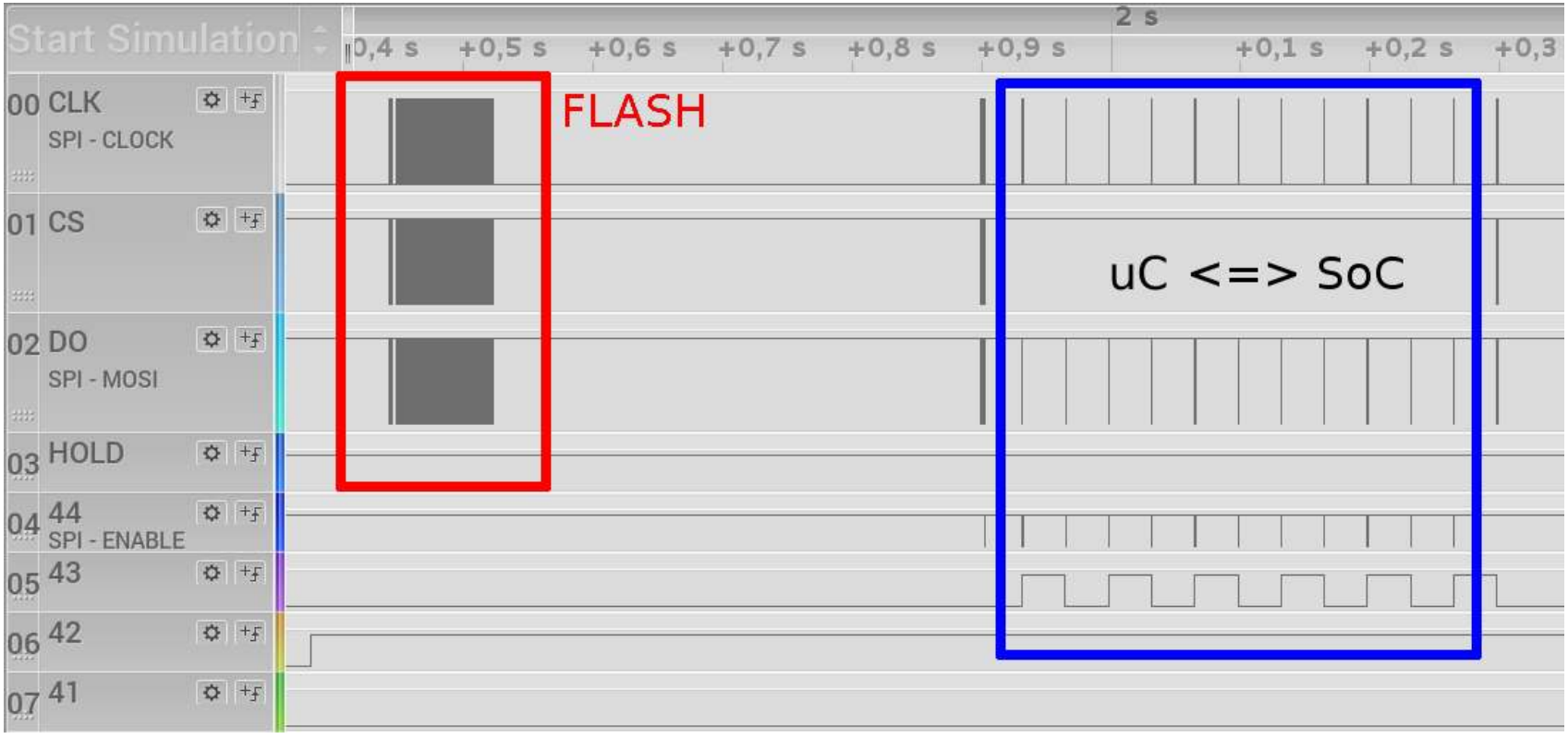
Traces PCB et Pin-out des composants



Placement des sondes



Capture d'écran



Analyse des communications flash SPI

Contrôleur USB-SATA et PIC vers flash

- placement des 4 sondes : simplement sur les pattes des mémoires
- paramètres du décodage SPI : "standard" (cf. datasheet)
- vitesse d'échantillonage : 50MS/s **mini**, 100MS/s recommandée (quartz à 25MHz)

Exploitation des résultats

- export CSV du décodage SPI
- script Ruby pour décoder les commandes flash :
  - affichage texte
  - création de *dump* binaire

Résultats

- le PIC n'écrit pas dans sa flash
- le contrôleur USB-SATA écrit des données lors de la validation du PIN

Analyse des communications SoC $\leftrightarrow$ PIC

Contrôleur USB-SATA $\leftrightarrow$ PIC

- placement des sondes : pattes SPI SATA (cf. analyse du PCB)
- vitesse : 50MS/s **mini**, 100MS/s recommandée
- protocole : inconnu

Exploitation des résultats

Protocole basé sur SPI :

- décodage bas niveau avec Saleae : export CSV
- nécessité de reverser la couche applicative

Protocole série

Reverse du protocole

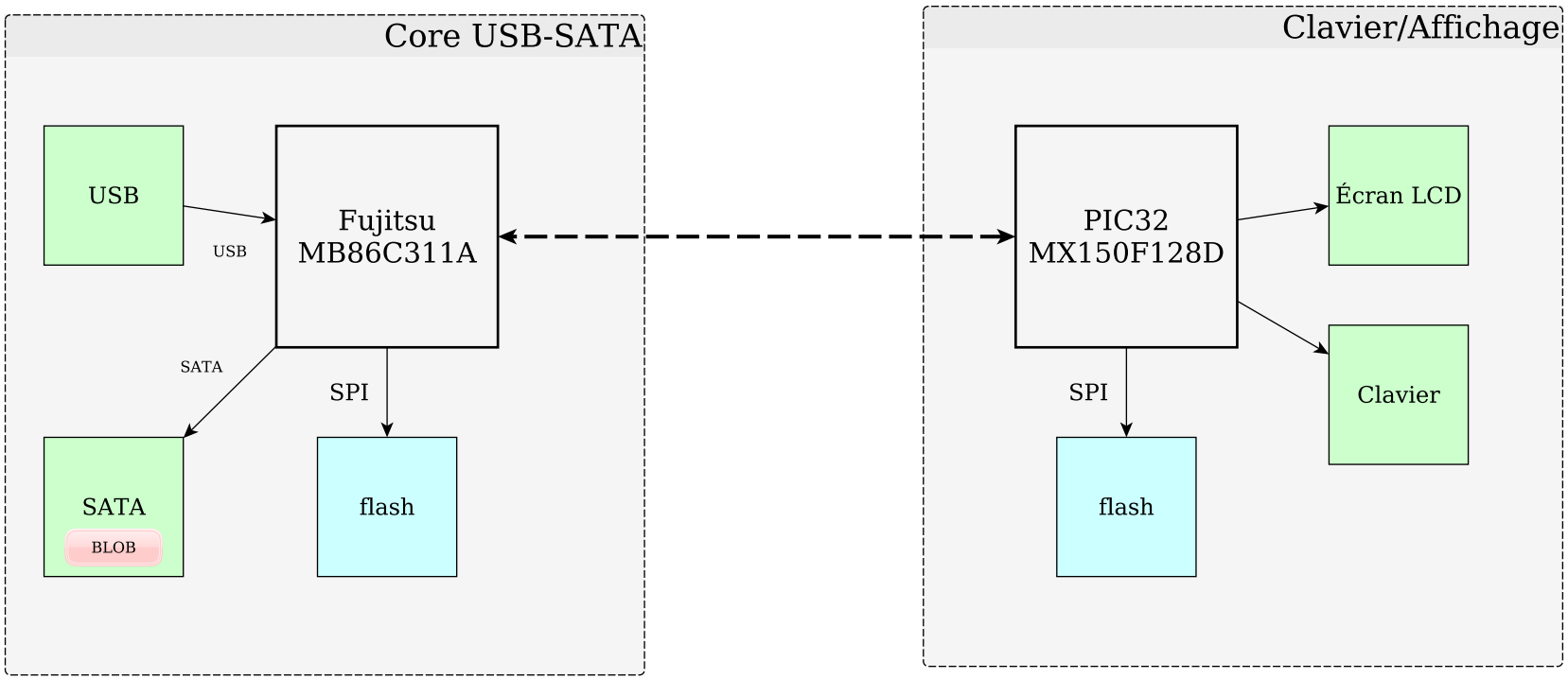
- préambule : `AA AA AA AA 55` (SoC → PIC) et `A5 A5 A5 5A` (PIC → SoC)
- format *Type, Length, Value*
- trames numérotées et acquittées
- *checksum* 16bits inconnu

⟹ Création d'un script Ruby de décodage à partir du CSV de Saleae

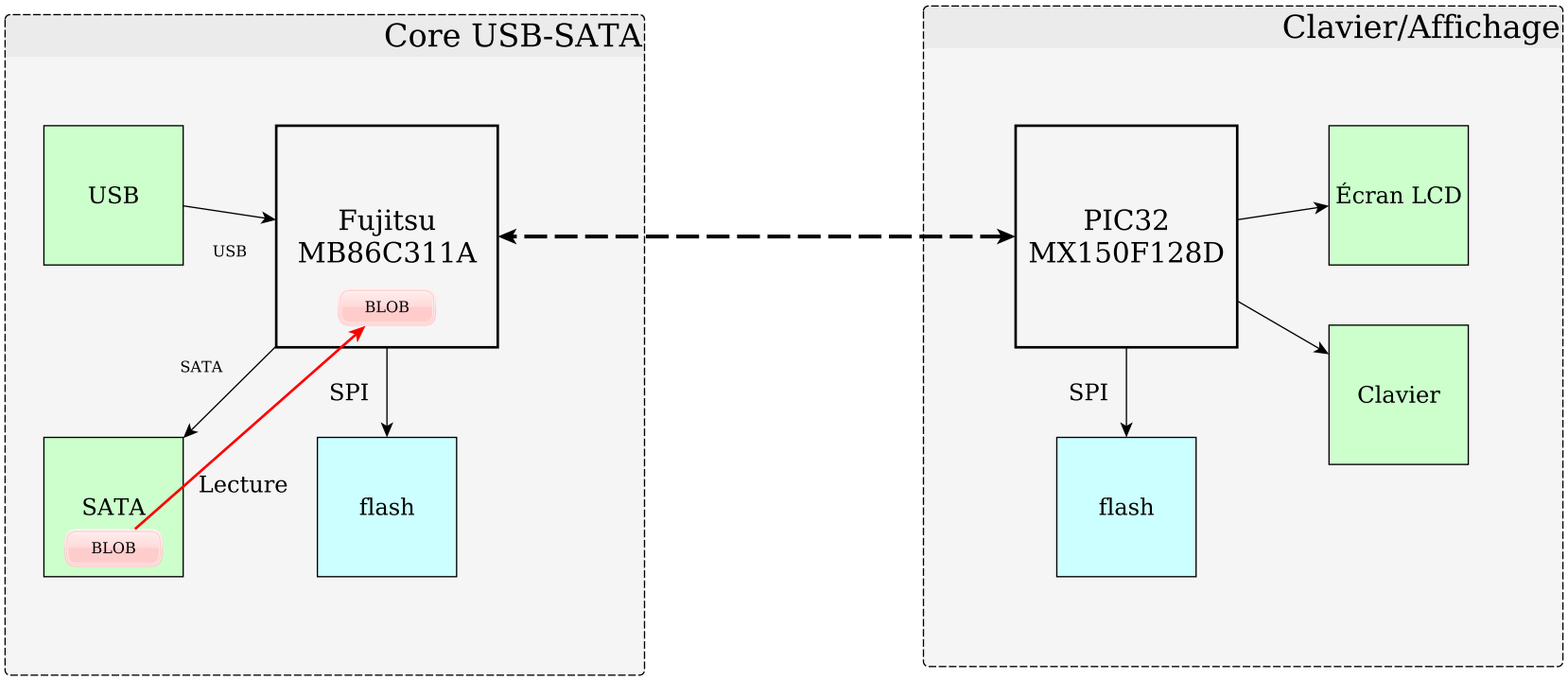
Exemple décodé : demande du PIN

```
0.00000000 SoC->PIC T: 0x33, ID: 0x14 | 01,01,10,01
0.00003861 PIC->SoC        RESP: 0x14 | 06,00,01,00,09,4d,01,cb,
                                        0e,00,00,00,89,0f,3a,7a
```

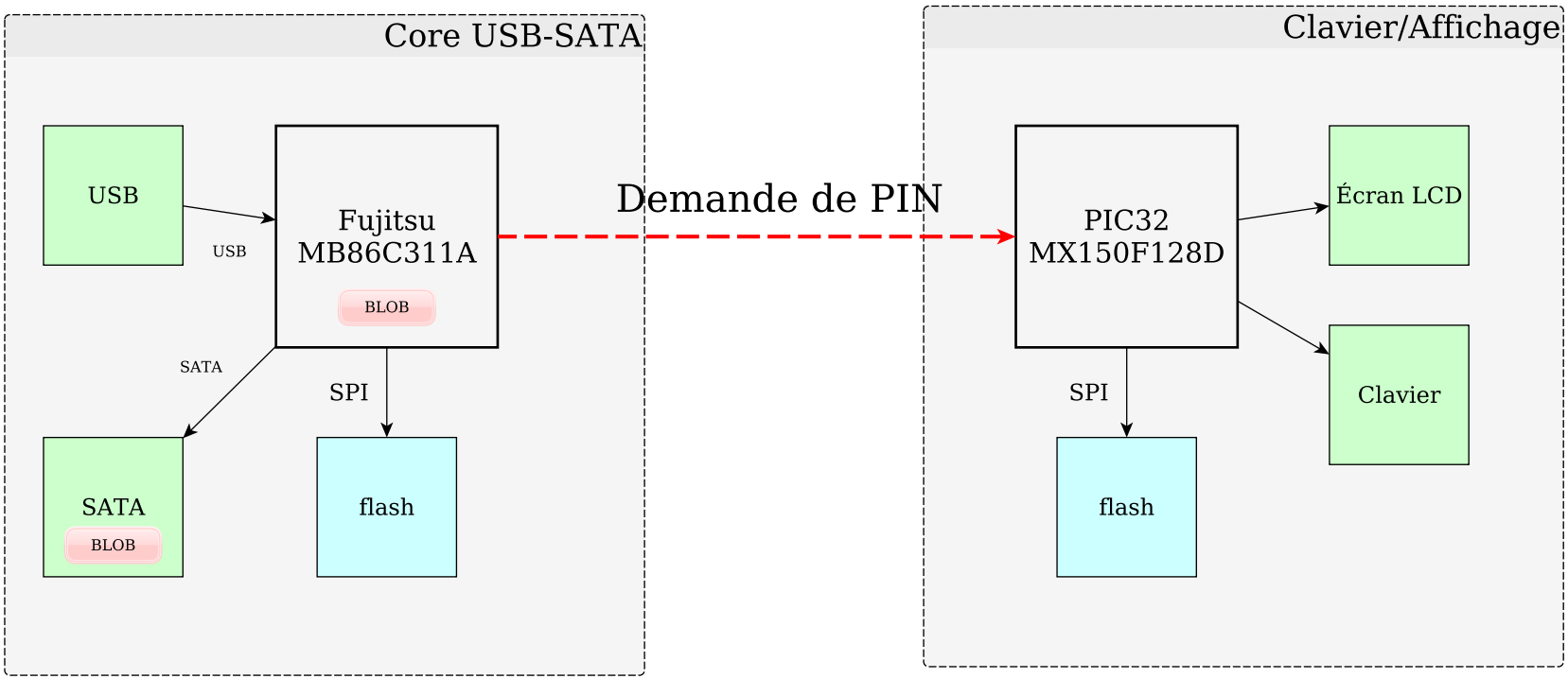
Résumé : séquencement des échanges



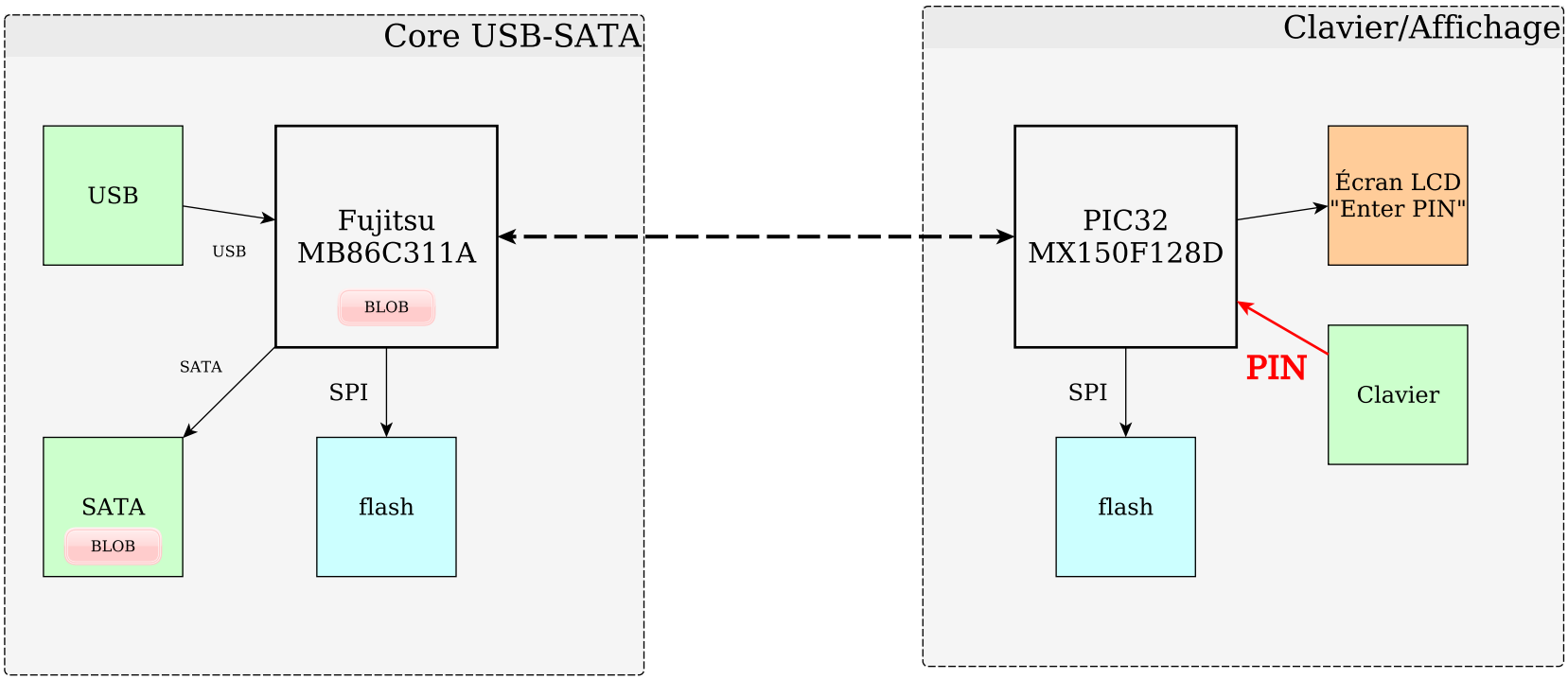
Résumé : séquencement des échanges



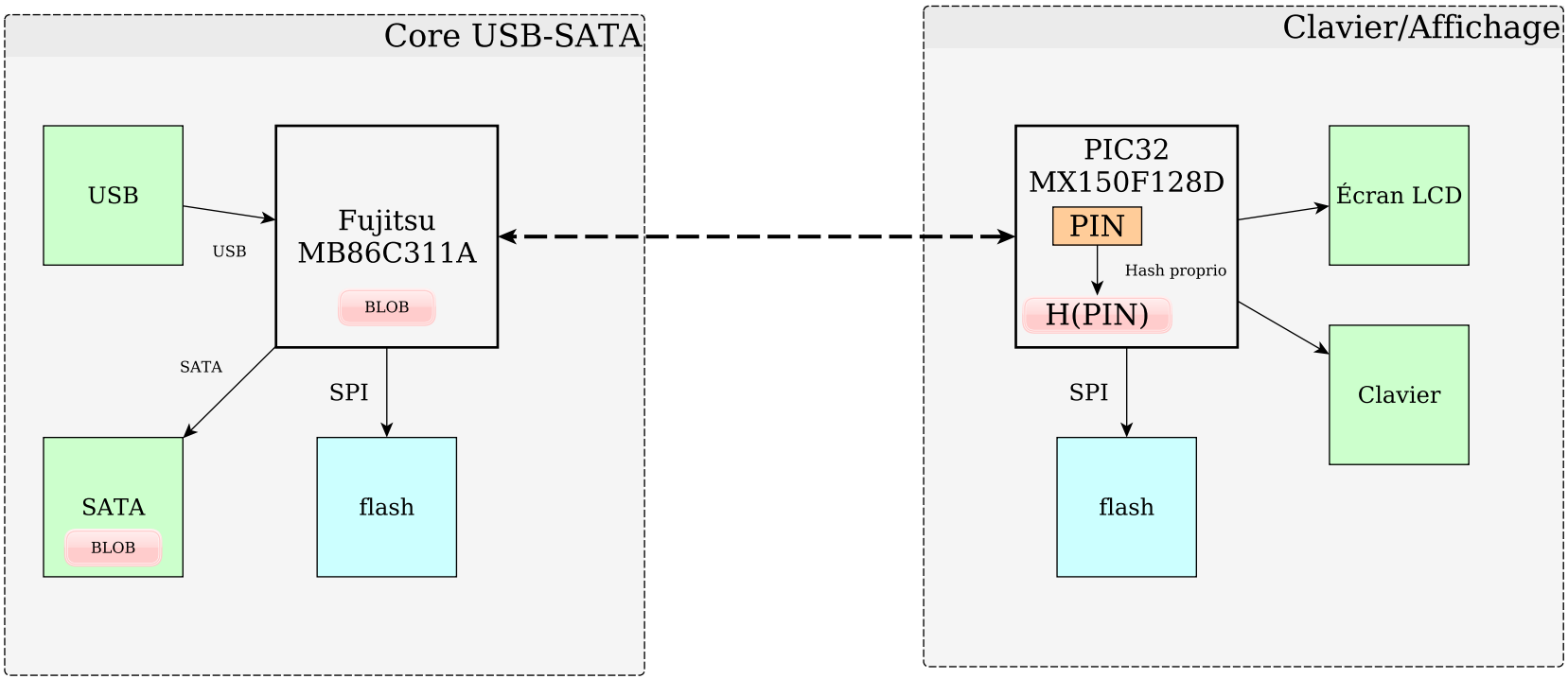
Résumé : séquencement des échanges



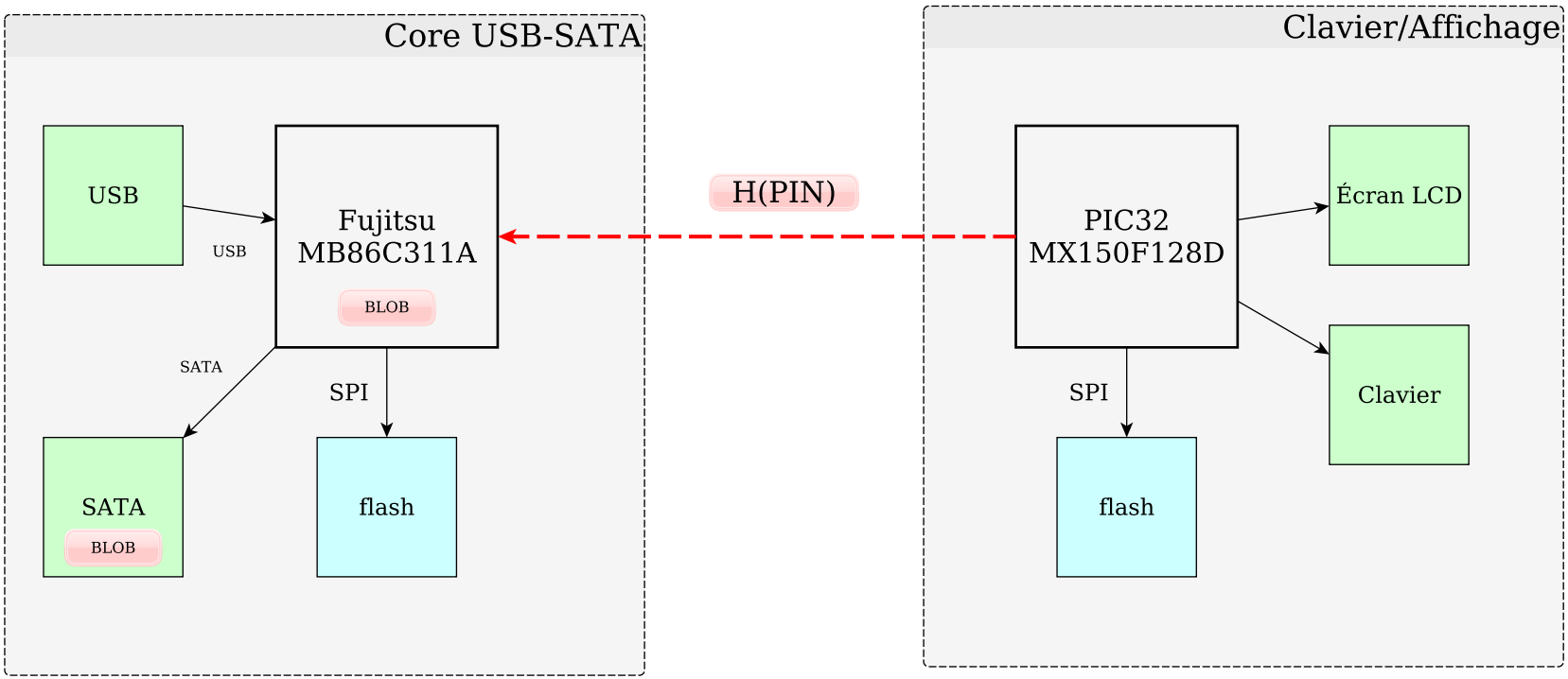
Résumé : séquencement des échanges



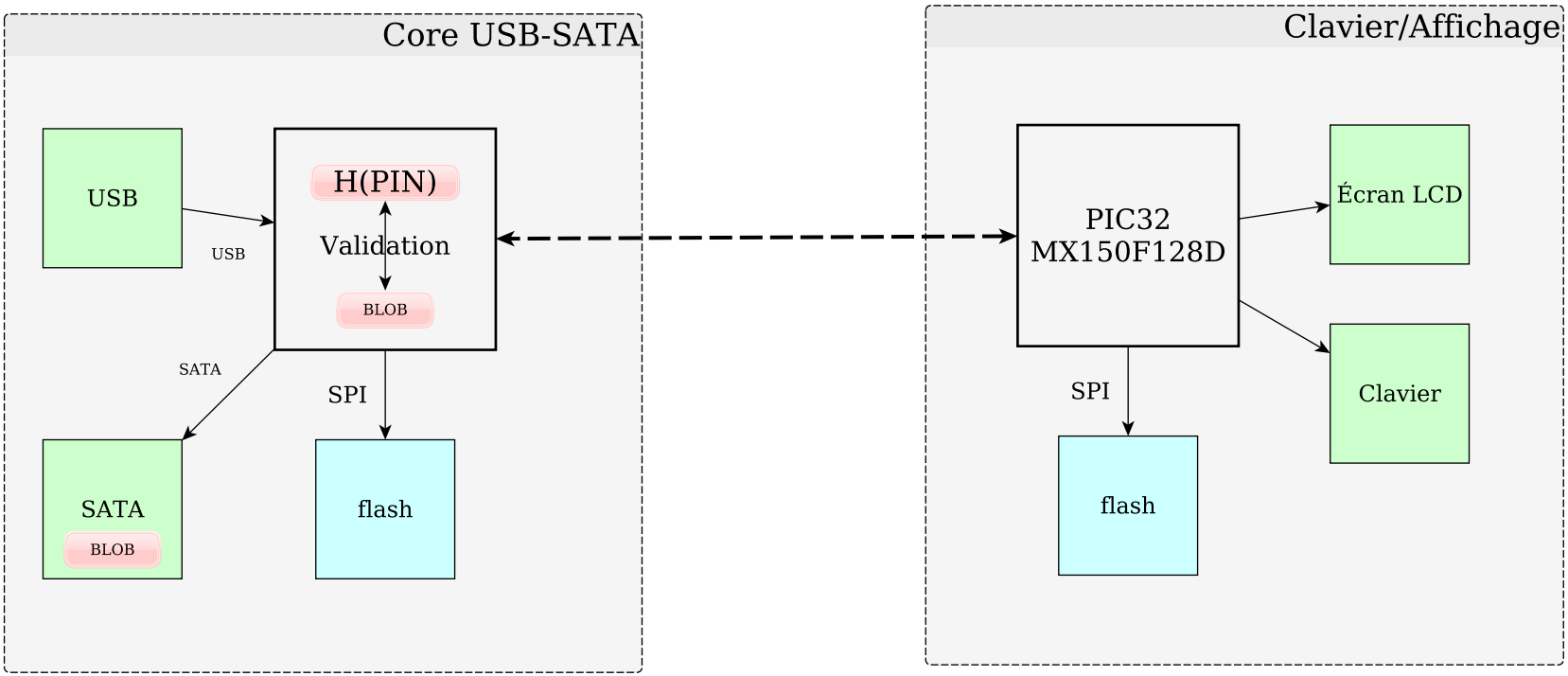
Résumé : séquencement des échanges



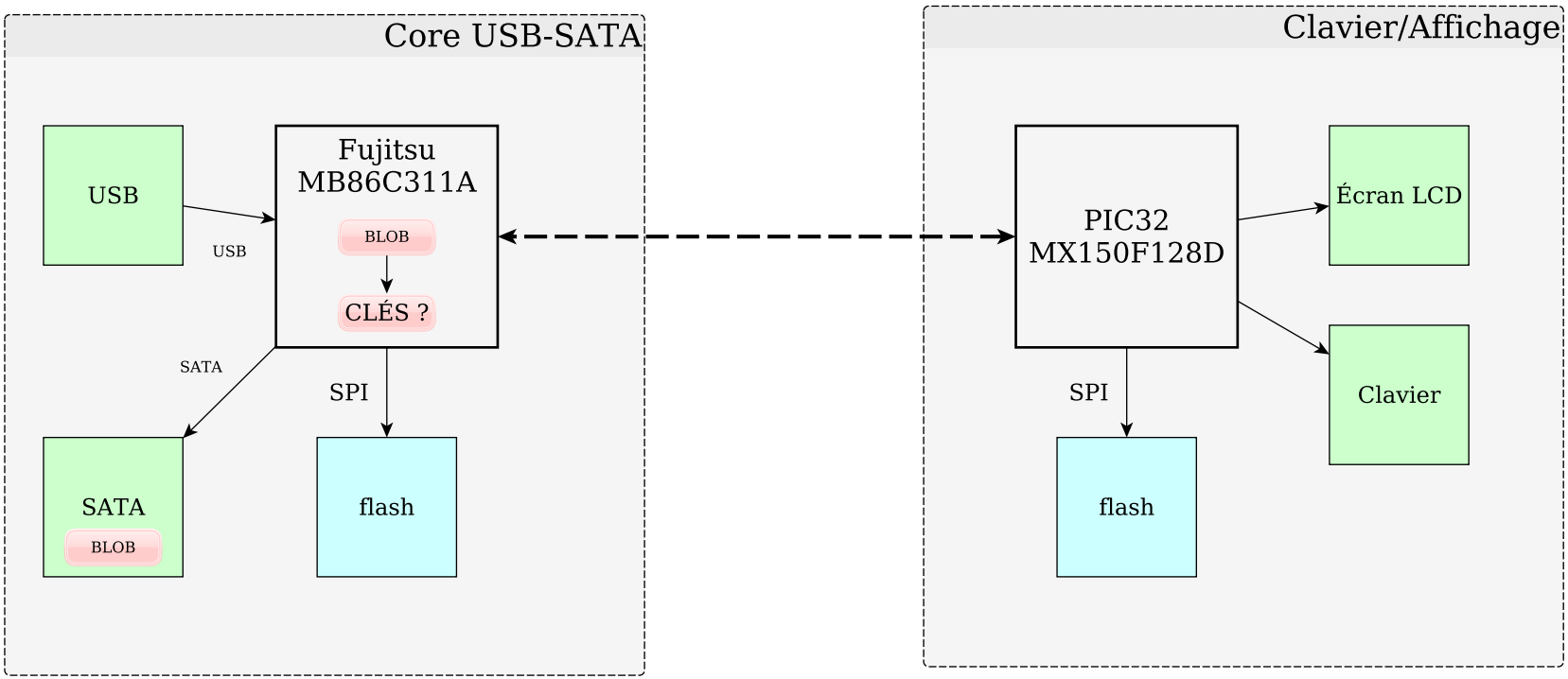
Résumé : séquencement des échanges



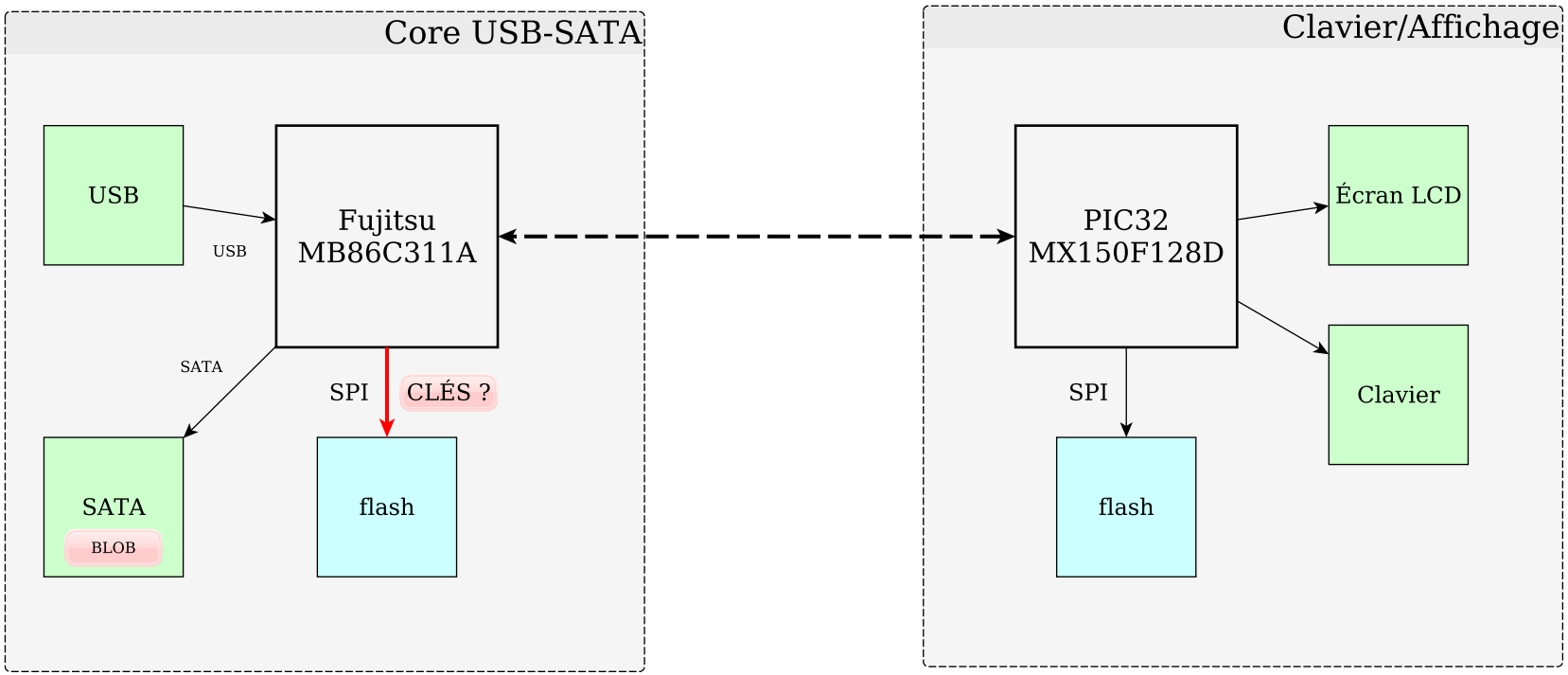
Résumé : séquencement des échanges



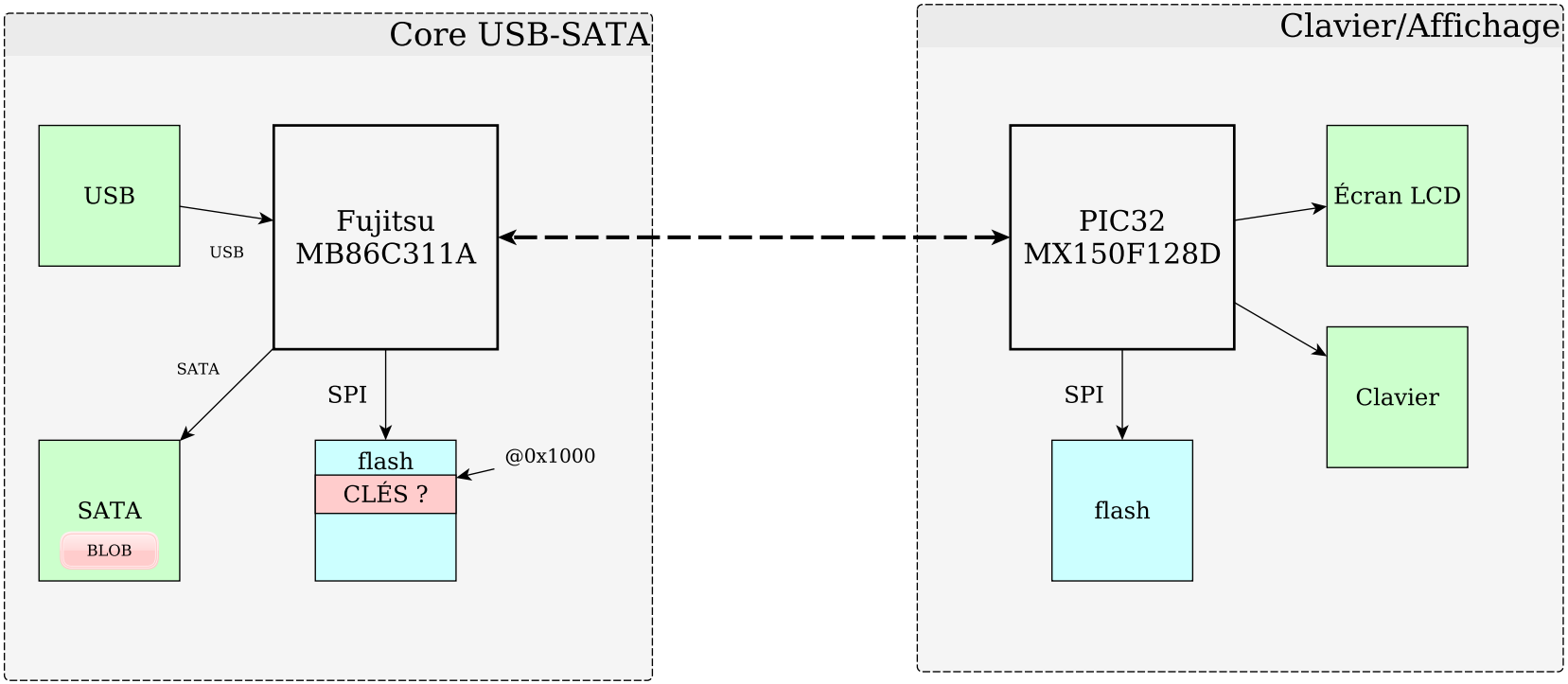
Résumé : séquencement des échanges



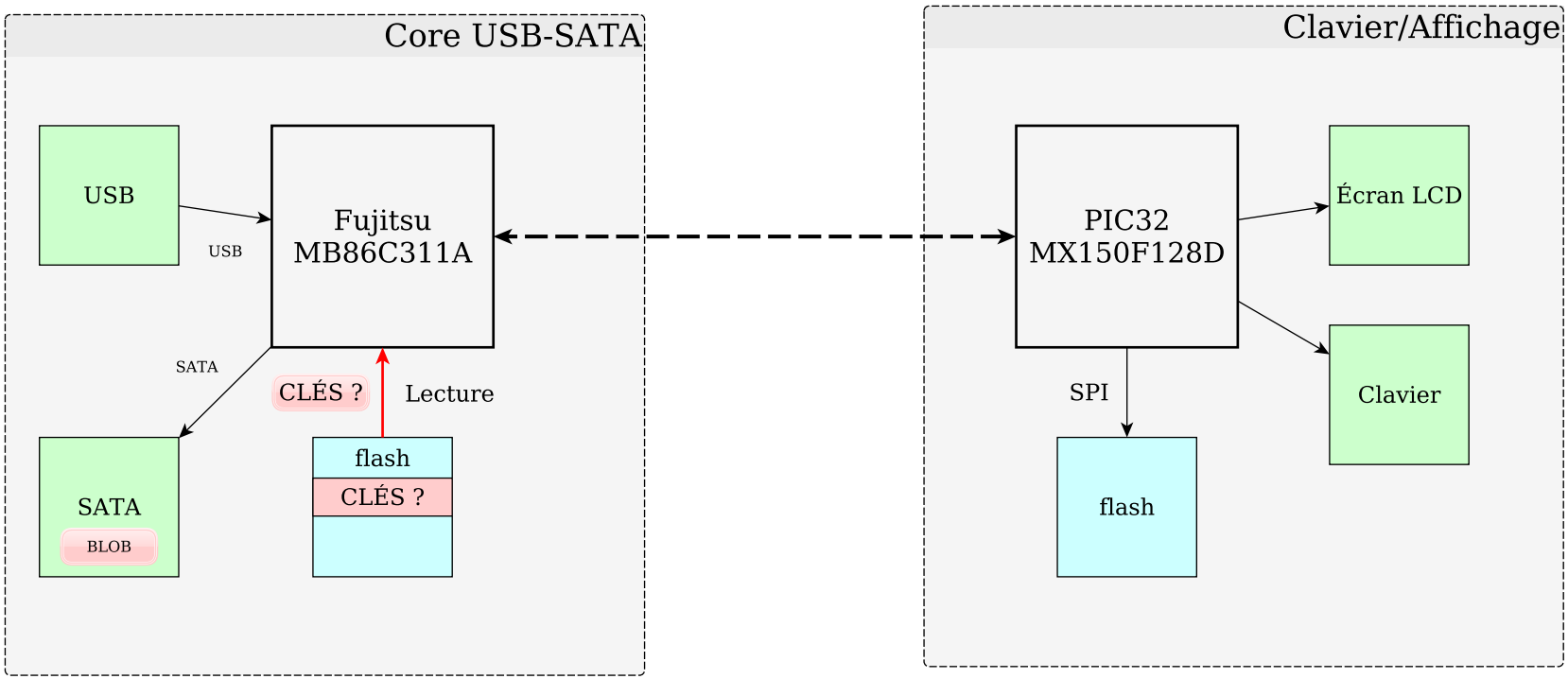
Résumé : séquencement des échanges



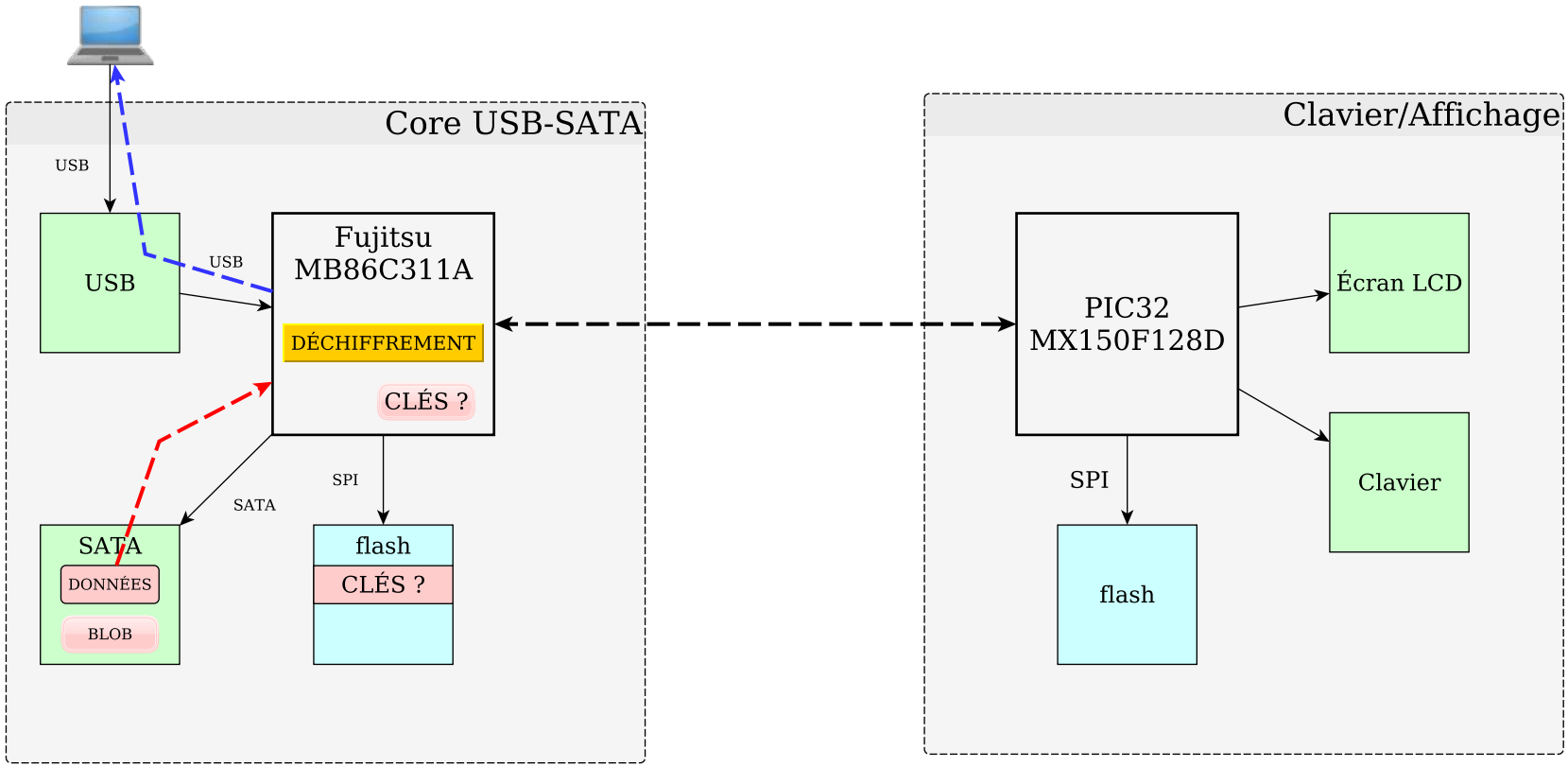
Résumé : séquencement des échanges



Résumé : séquencement des échanges



Résumé : séquencement des échanges



Et maintenant ?

Questions en suspens

- Peut-on réaliser un bruteforceur matériel (émulateur clavier/PIC) ?
  - non car l'algorithme de hachage est inconnu
- Que contient le bloc à 0x1000 ?

Bloc flash @ 0x1000

Propriétés :

- écrit lors :
  - de l'activation du chiffrement,
  - de la validation d'un PIN correct ;
- effacé lors de la désactivation du chiffrement ;
- contient 3 blocs séparés de données à forte entropie :
  - 512 bits, clé AES-256 n°1 chiffrée ?
  - 512 bits, clé AES-256 n°2 chiffrée ?
  - le SHA256 des données précédentes.

Conception d'une attaque

Hypothèse

Le bloc @ 0x1000 semble **contenir les clés de chiffrement AES-XTS**, sous forme chiffrée ou obfusquée (les tests de déchiffrement n'ont rien donné).

Conséquence ?

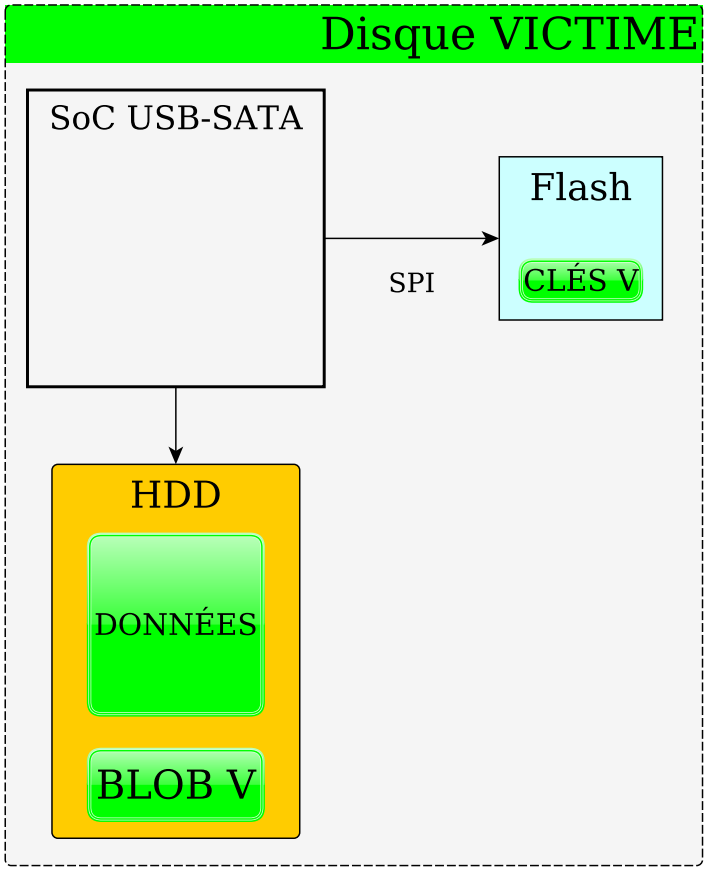
Peut-on utiliser ce bloc pour mener une attaque ?

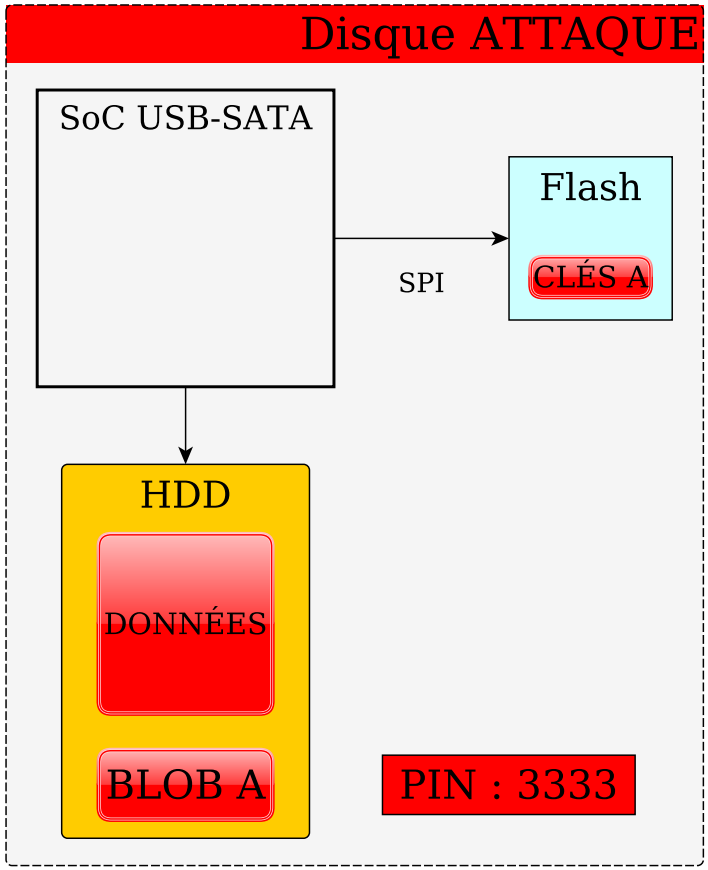
Idée de base de l'attaque

Le bloc à 0x1000 contient les clés de déchiffrement :

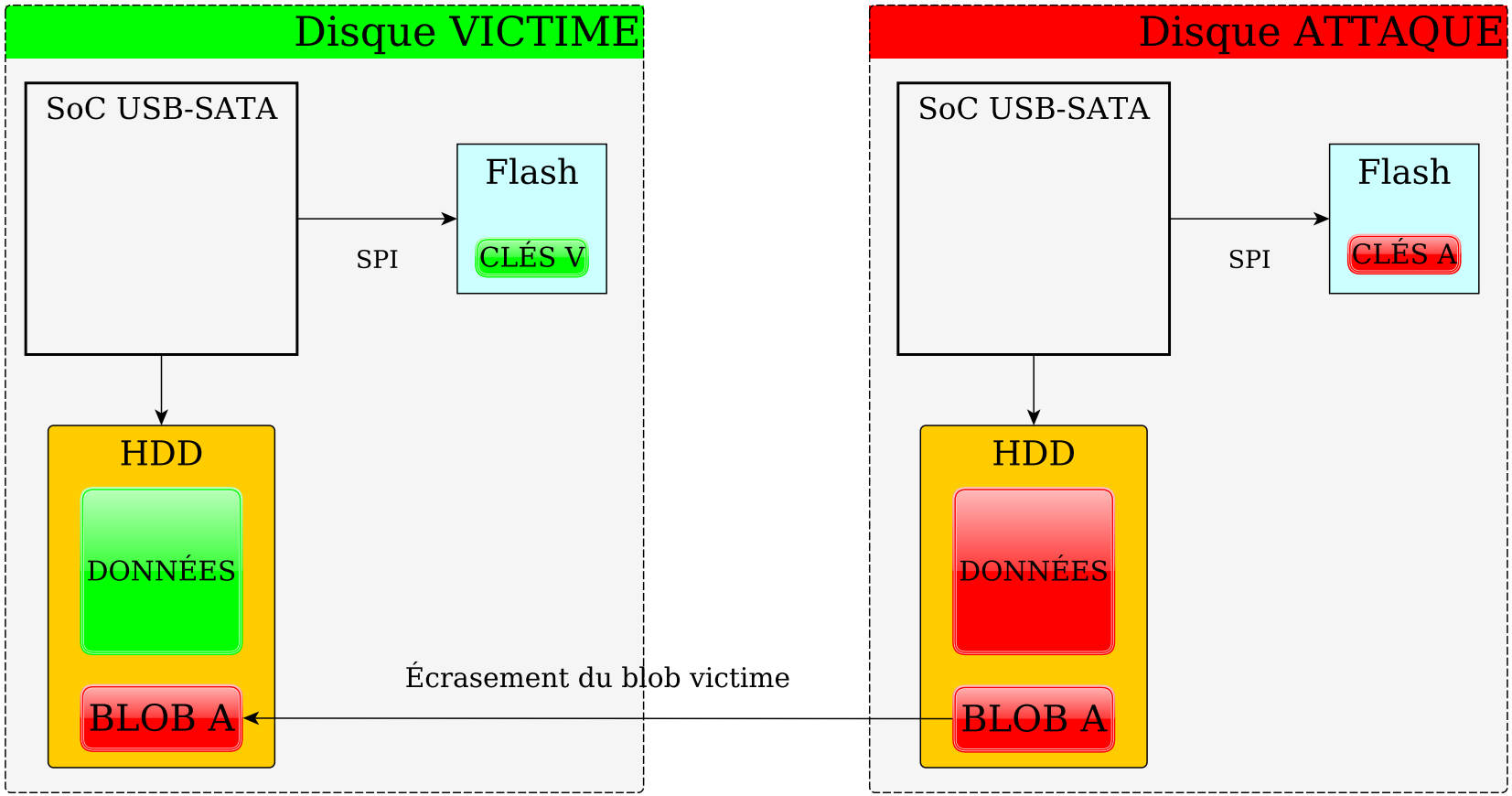
- on va tenter de garder celui du disque cible ;
- en utilisant un PIN connu dans le *blob* du HDD.

Déroulement théorique

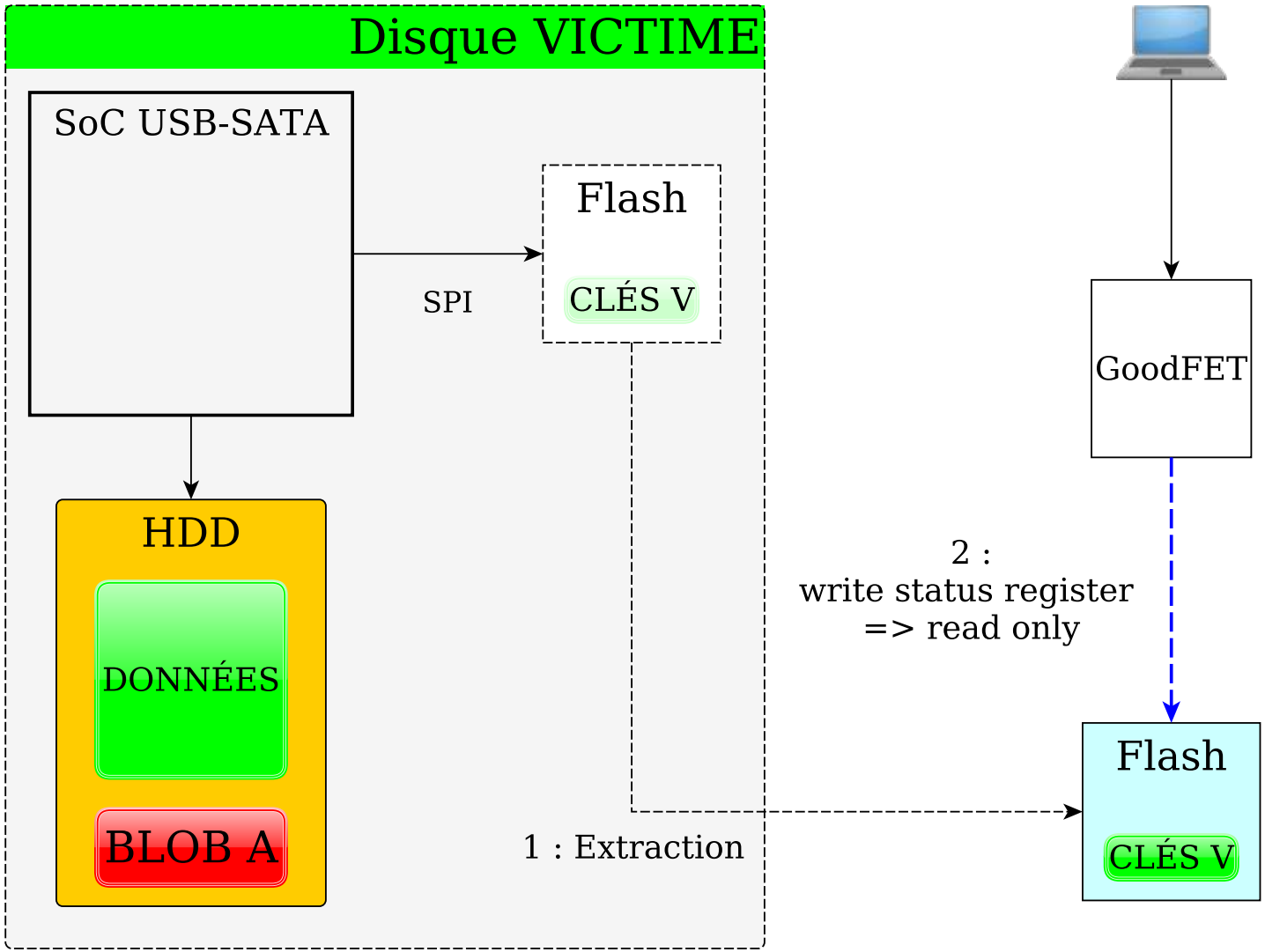




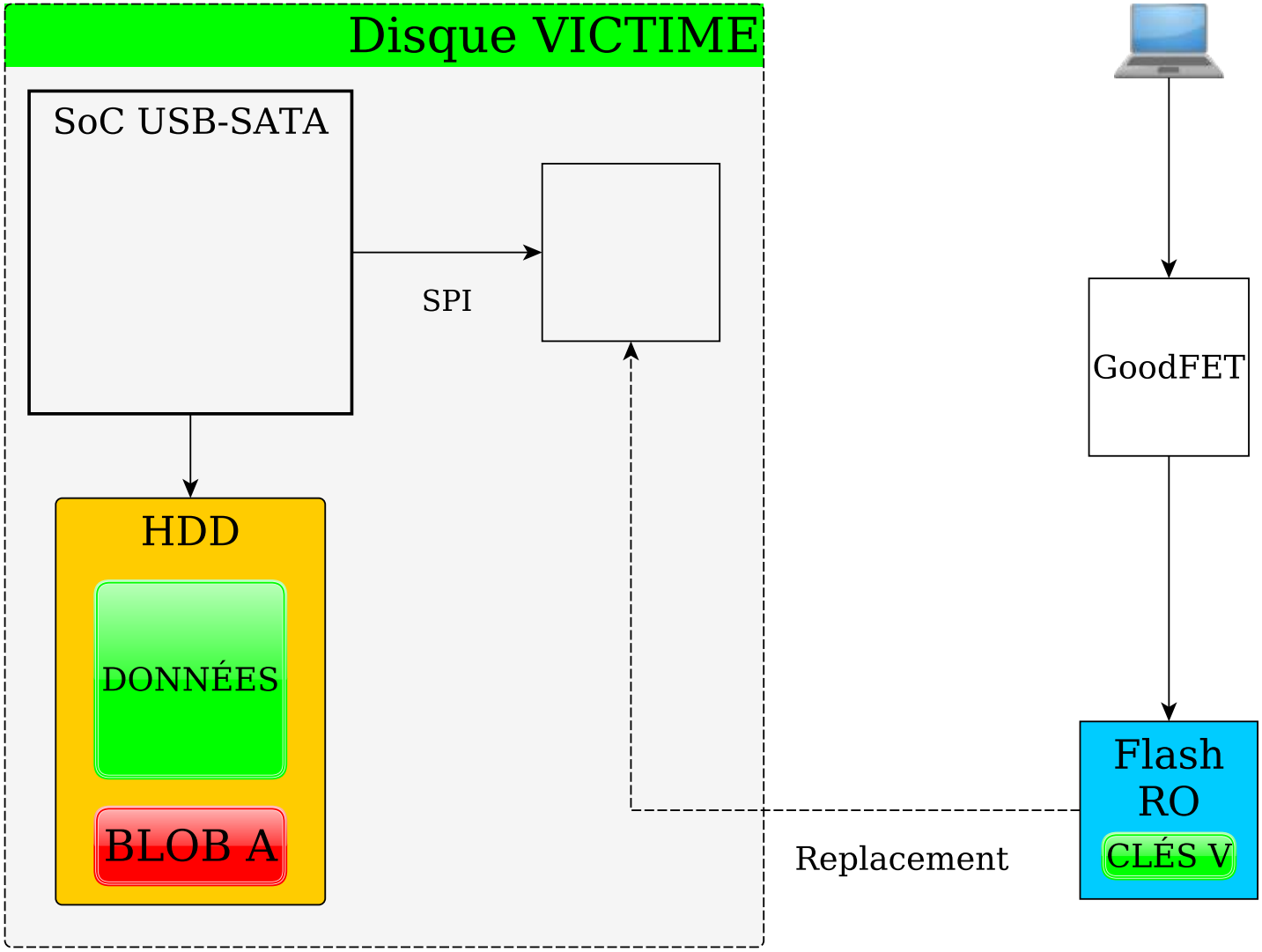
Déroulement théorique



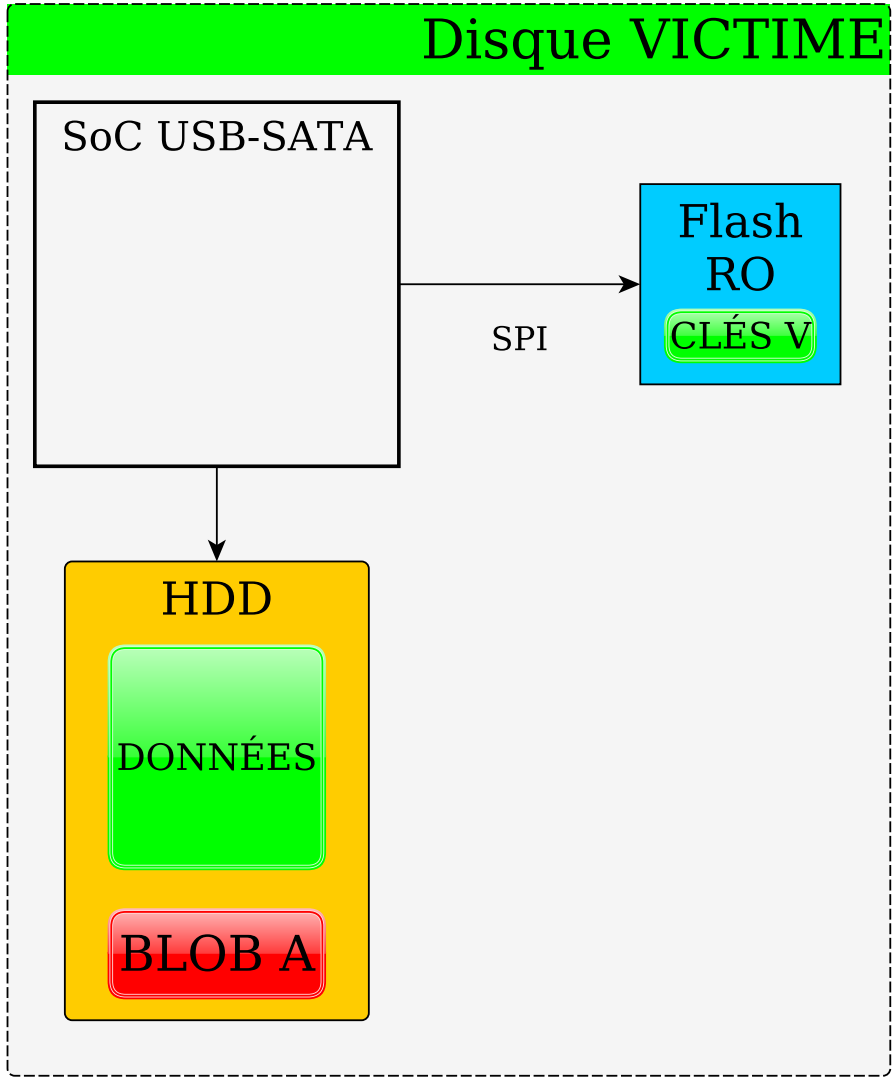
Déroulement théorique



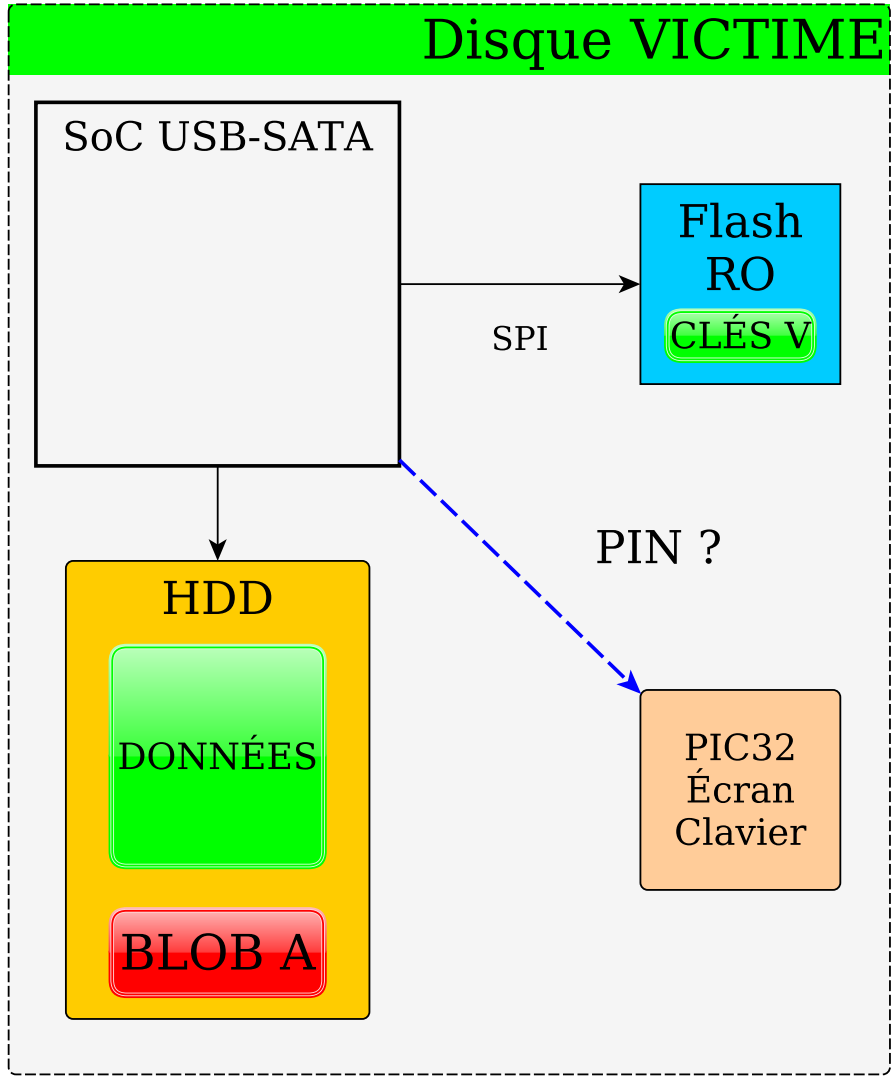
Déroulement théorique



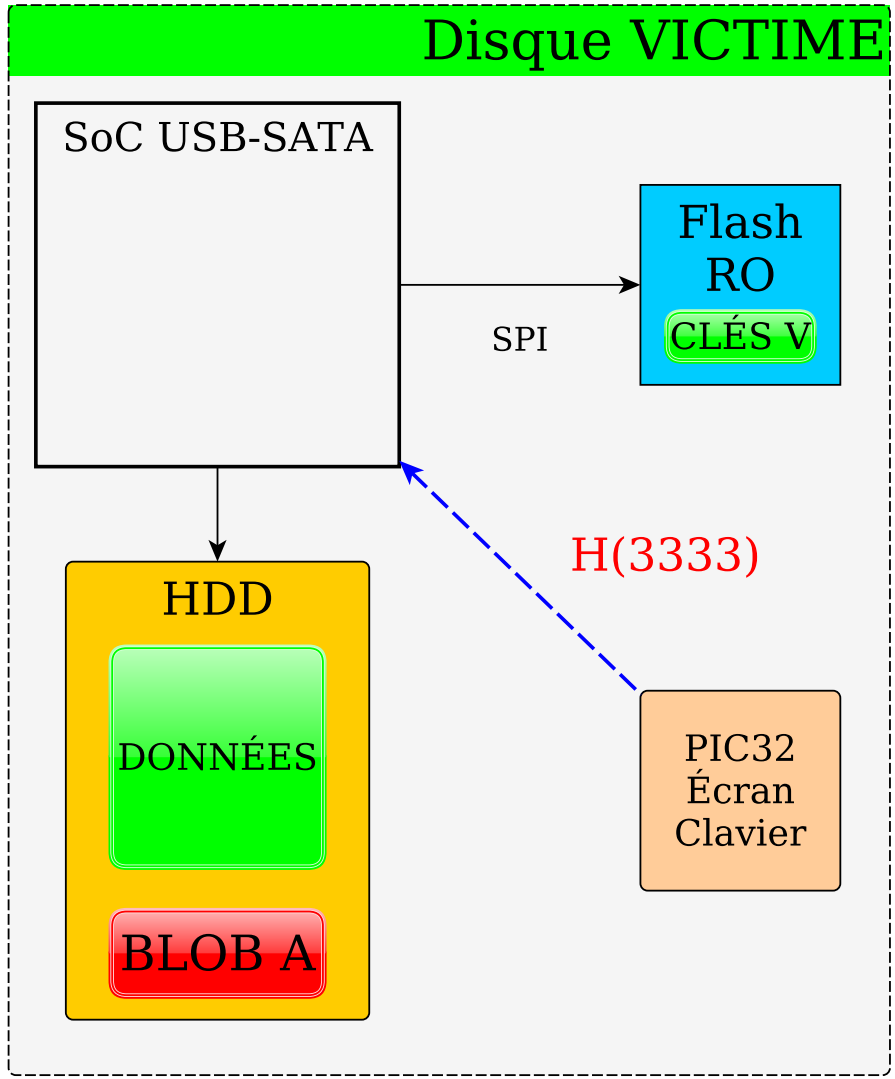
Déroulement théorique



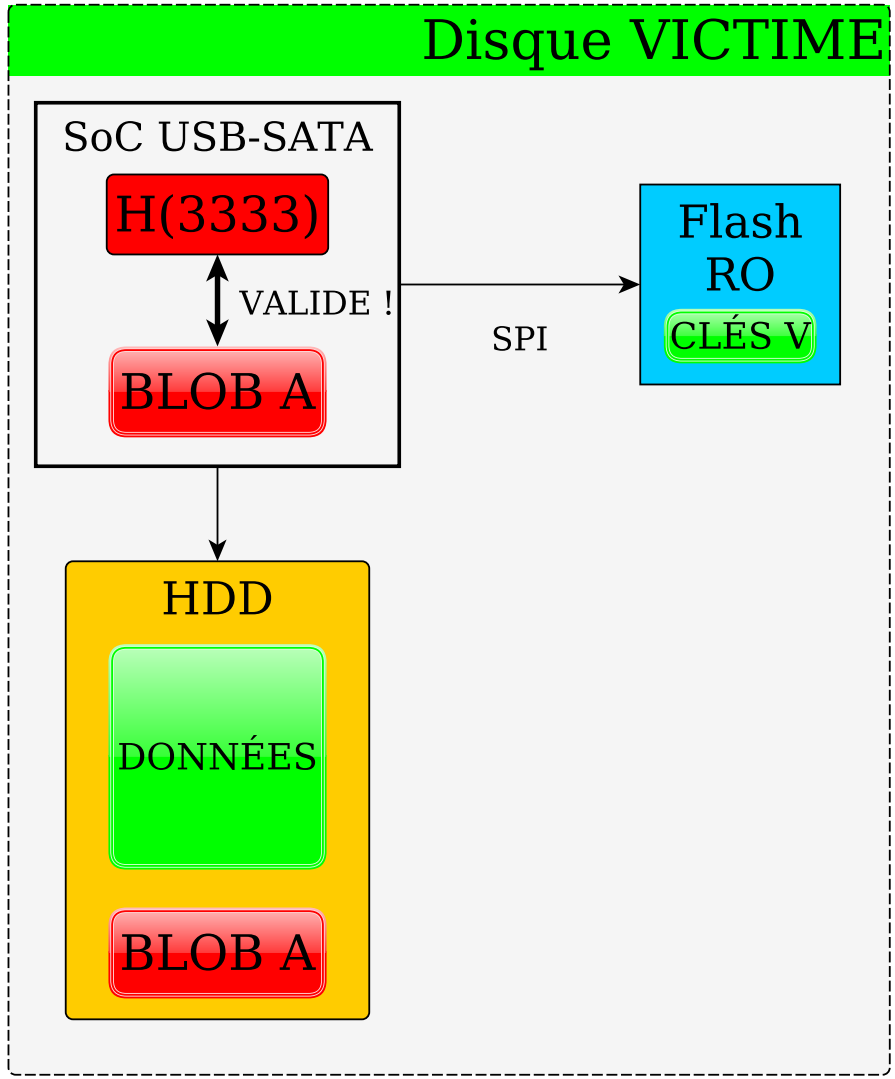
Déroulement théorique



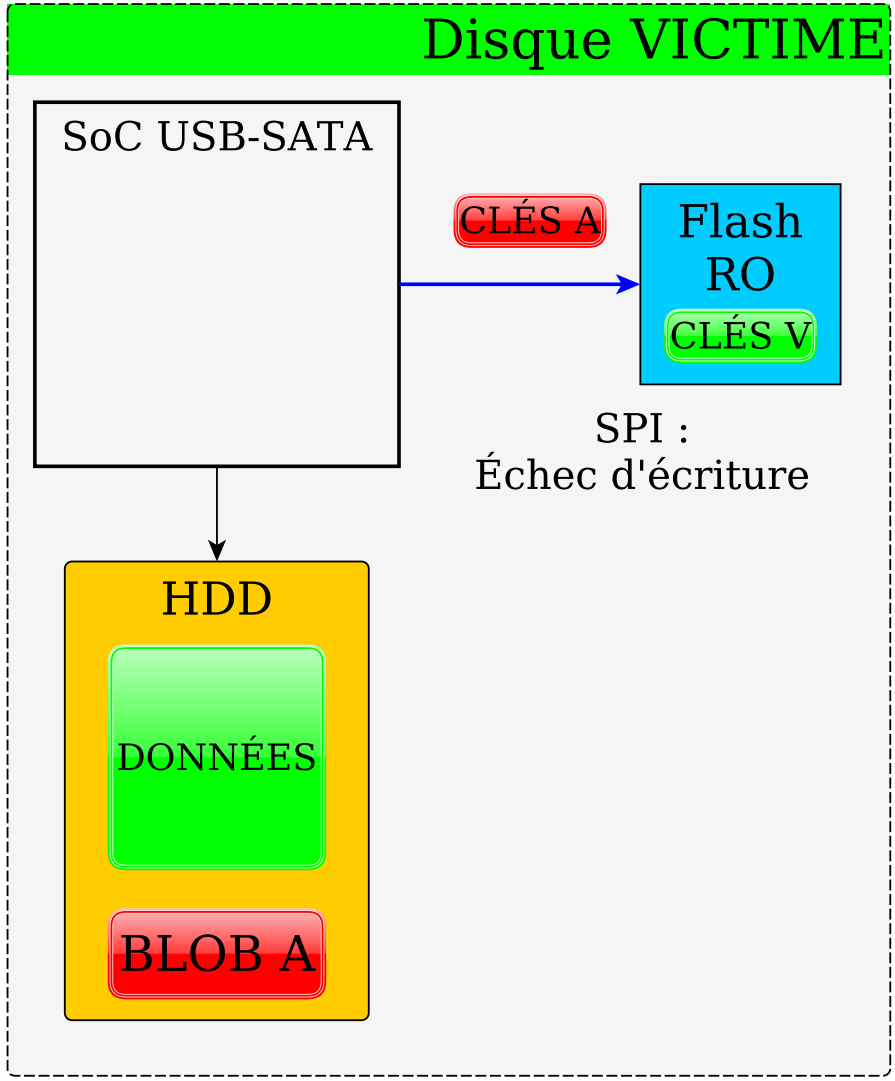
Déroulement théorique



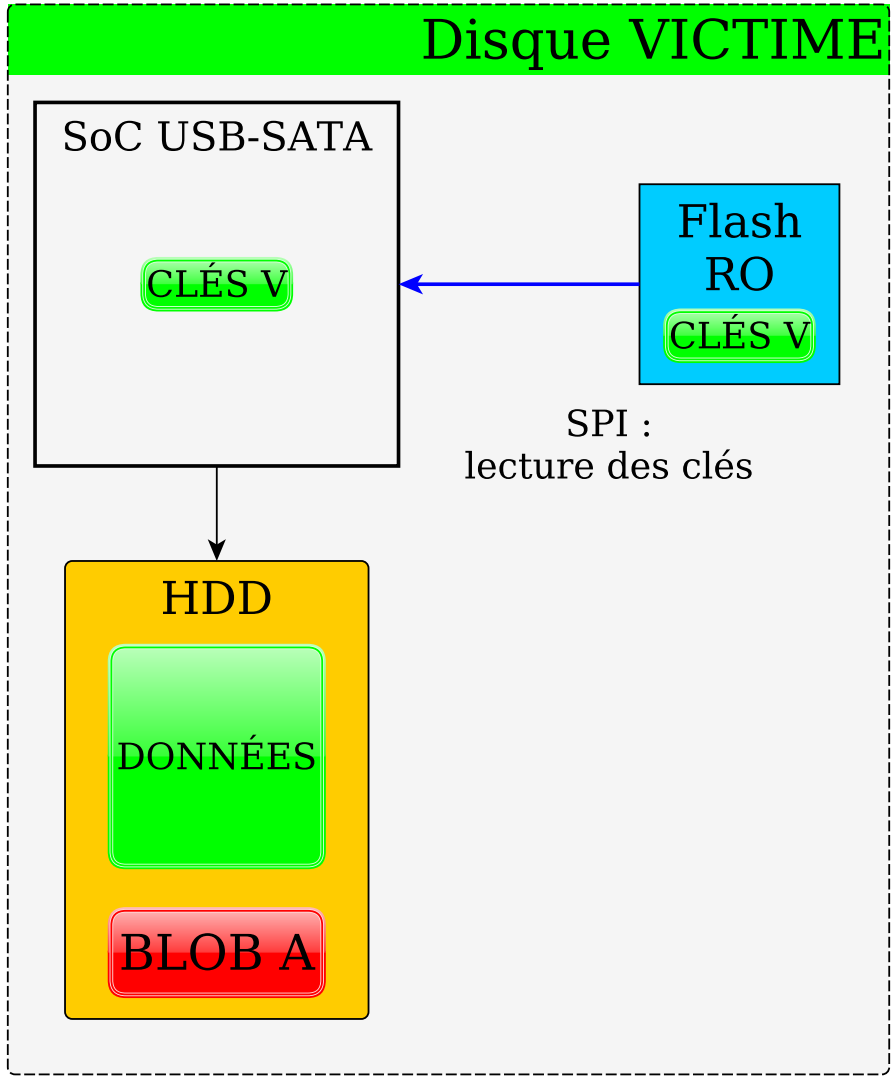
Déroulement théorique



Déroulement théorique



Déroulement théorique



En pratique

Premier échec

status register remis à zéro au démarrage par le disque

Évolution de l'attaque

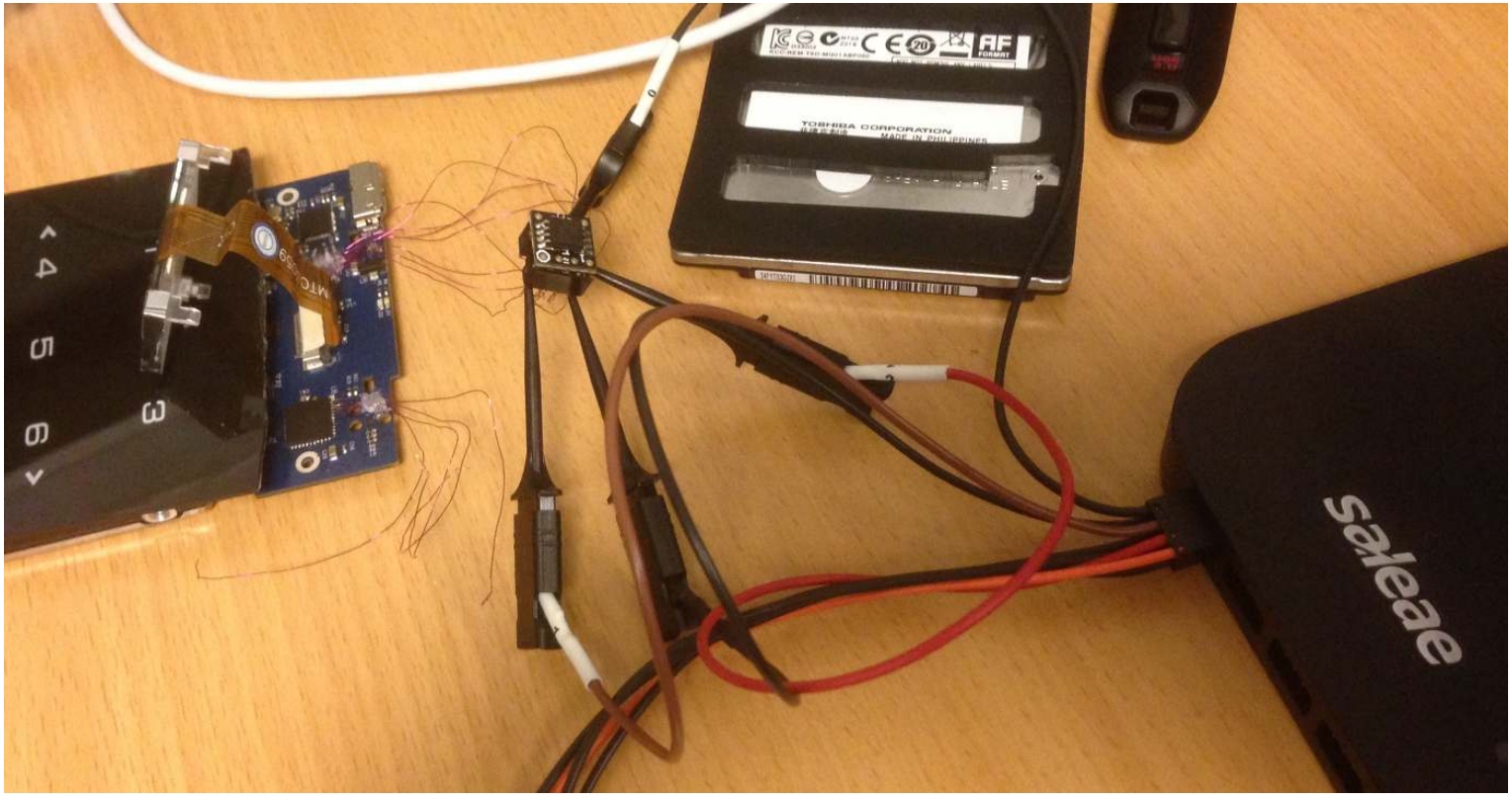
Mise en lecture seule de la flash à chaud, après le démarrage :

1. booter le disque
2. débrancher la flash
3. mettre en lecture seule
4. rebrancher
5. continuer l'attaque : entrer le PIN maîtrisé

Résultat final

Échec. Le PIN n'est pas validé (*Not match*).
$\Longrightarrow$ Il y a probablement une vérification, inconnue.

Attaque finale : démo



Conclusion

Sécurité des données chiffrées

La sécurité repose entièrement sur :

- la sécurité du *blob* de la fin du disque
- la sécurité du bloc @ 0x1000 dans la flash

$\Longrightarrow$ Tout repose sur le fait que le firmware Fujitsu est "secret".

Échanges avec le développeur

Évolution du *firmware* :

- hash du PIN non déterministe

Problème :

- gestion opaque du *blob* de la fin du disque : code binaire Fujitsu

Conclusion : suite

Étude du VE400

- intéressante à réaliser ;
- permet d'avoir une idée du niveau de sécu assez simplement.

La suite ?

Accéder au code du contrôleur USB-SATA :

- trouver un JTAG ?
- protection du firmware (sur la flash) identique sur **toutes** les puces :
  - "acheter" le SDK (attention au NDA)
  - trouver une âme généreuse :)
- *reverser* le SoC au niveau matériel :
  - *decap*,
  - image au microscope,
  - analyse des transistors.

Fin

Questions ?

References

[1] http://support.ironkey.com/article/AA-02513/

[2] http://www.h-online.com/security/features/USB-stick-with-PIN-code-746169.html

[3] https://www.exploit-db.com/papers/15424/

[4] http://sigrok.org/wiki/Main_Page

[5] http://support.saleae.com/hc/en-us/articles/200672010

Comparaison de blobs

```
bloc_ssd
0000 0000: 6E D1 40 A2 74 48 51 93  D1 5B 96 13 18 25 F7 67  n.@.tHQ. .[...%.g
0000 0010: CF 73 BB 45 0A 4F 02 11  35 34 C9 39 45 31 BE 44  .s.E.O.. 54.9E1.D
0000 0020: 03 93 32 E1 8A 64 69 2E  D6 1B 21 9F A5 51 88 8C  ..2..di. ..!..Q..
0000 0030: 16 57 FF 71 32 CA E8 82  69 68 6A 3A DE 77 EC 06  .W.q2... ihj:.w..
0000 0040: DB D9 35 2F 47 32 FC D8  30 9F 06 B7 87 C0 F3 87  ..5/G2.. 0.......
0000 0050: 66 22 4D 32 C0 58 98 65  6C 50 29 E2 FE CE A5 30  f"M2.X.e lP)....0
0000 0060: 23 D5 11 42 87 38 F5 8E  11 36 D1 8D 0C C6 67 63  #..B.8.. .6....gc
0000 0070: C1 7B 80 63 54 21 9C 7D  61 CB 33 5C 29 8C 1D DE  .{.cT!.} a.3\)...
0000 0080: B8 00 83 E9 36 50 FB FE  01 66 B5 EB F9 26 D7 64  ....6P.. .f...&.d
0000 0090: 7F FB 61 76 42 CE C7 06  74 28 EE 58 EB 3E 8C 26  ..avB... t(.X.>.&
0000 00A0: 0E 6B 94 99 7B 48 97 A3  73 33 58 A6 EE B6 9B 47  .k..{H.. s3X....G
0000 00B0: C8 81 F4 F8 C9 1B F5 8F  FB 2F 0C 73 B5 C9 CC 8E  ........ ./.s....
0000 00C0: AC B7 1E 03 FD 6D 9D E6  46 28 7F 2A 80 E1 17 01  .....m.. F(.*....
0000 00D0: 97 A7 D2 8F 33 17 A2 9E  9E BE 1C C6 AB CE E7 FC  ....3... ........
0000 00E0: 4D A6 74 27 D7 C9 3A 03  64 2C 3D 52 A4 2E A6 89  M.t'..:. d,=R....
bloc_toshiba
0000 0000: E5 91 D4 9A FD 40 12 21  1B DA 56 6D 67 AB 07 7C  .....@.! ..Vmg..|
0000 0010: 86 03 F4 AF BA 4D C3 72  D9 F4 61 F6 CF F0 28 84  .....M.r ..a...(.
0000 0020: AB EA 02 1C 08 3F 93 DB  69 BF 06 EA 8D 52 6D 16  .....?.. i....Rm.
0000 0030: 1F 7D 0A 44 7D 47 85 15  EE 43 27 74 3B CF 12 C0  .}.D}G.. .C't;...
0000 0040: E8 DC 87 82 FE 8E 40 14  D1 AC 1C 13 3F D0 84 C3  ......@. ....?...
0000 0050: 84 33 44 E4 9F 72 C3 F1  60 53 58 43 C1 6A D6 AC  .3D..r.. `SXC.j..
0000 0060: AD C8 94 88 BF 57 23 33  D4 46 77 12 3B 4C B1 AB  .....W#3 .Fw.;L..
0000 0070: E0 C7 37 ED 40 15 9C 09  60 3C 06 56 F1 F9 88 DD  ..7.@... `<.V....
0000 0080: 94 35 66 7B 5C 3C C0 51  DE A9 0F 20 B3 71 1D 17  .5f{\<.Q ... .q..
0000 0090: 52 17 6F 88 4B CF C6 E5  B8 54 C8 75 EF 93 F9 AA  R.o.K... .T.u....
0000 00A0: A7 74 E8 3D 66 D1 FB 4C  91 3F D5 2A 98 8C 75 B3  .t.=f..L .?.*..u.
0000 00B0: 04 C7 5C 53 53 7A 8E E3  AB FB 2B 2E 44 E1 98 27  ..\SSz.. ..+.D..'
0000 00C0: 9B 96 58 07 8A A8 60 19  DB 32 DE BF 26 58 1E 2A  ..X...`. .2..&X.*
0000 00D0: F4 05 34 88 2F F6 6B A1  50 01 FE 80 BA B8 1F 26  ..4./.k. P......&
0000 00E0: CD CC DD 80 77 EC 91 50  EE 25 50 79 56 18 DC C9  ....w..P .%PyV...
```

Comparaison de firmwares : Zalman vs PS4

```
flash_controlerSATA
0000 20C0: A2 3E 19 19 F5 C8 85 41  B9 E4 92 15 9F F2 CA 77  .>.....A .......w
0000 20D0: 6C D3 BE 77 6F 17 0A 85  88 14 2E 49 3E 22 F5 05  l..wo... ...I>"..
0000 20E0: 96 B0 C1 3A 93 23 4C 51  7C 7A BB CD C3 19 13 7F  ...:.#LQ |z......
0000 20F0: B2 8F 34 59 B7 0E B4 F2  75 43 10 D5 5B 22 7D 86  ..4Y.... uC..["}.
0000 2100: 0E 93 D1 03 4E 37 BB D1  1C C9 DF 95 EC 7C 73 37  ....N7.. .....|s7
0000 2110: 83 90 A9 EF 89 A1 2B 12  BB 52 38 C2 4F 6B 8F DC  ......+. .R8.Ok..
0000 2120: 01 31 47 D6 9B 97 4F F1  3A 01 87 DC C6 50 18 95  .1G...O. :....P..
0000 2130: D7 0E 75 E0 17 83 32 A0  19 3D 46 5A DC 44 88 DF  ..u...2. .=FZ.D..
0000 2140: E4 D0 84 89 86 FC 9B BD  FA D7 F1 BE C5 79 EF C4  ........ .....y..
0000 2150: 96 2D D2 5C 5C F4 4C E8  24 83 93 CB 12 B1 18 04  .-.\\.L. $.......
0000 2160: 94 BD 16 44 49 C3 54 36  76 A6 4A D1 5D 4C BE E0  ...DI.T6 v.J.]L..
0000 2170: FF 60 7D 96 D3 DD 9C C7  9A 69 C0 60 C7 7F EB 8F  .`}..... .i.`....
0000 2180: DE F1 0E CB 7F C9 55 28  D7 23 7E 1F 98 10 00 4D  ......U( .#~....M
0000 2190: 53 8D CF 14 50 32 6C 6E  82 C6 E1 06 2B C6 22 B4  S...P2ln ....+.".
0000 21A0: 8A 23 ED EB F4 46 0F 15  02 EF 45 0A 77 59 A3 9B  .#...F.. ..E.wY..

PS4_dump.bin
0000 20C0: C0 15 19 19 81 19 85 41  09 6D 92 15 9F F2 CA 77  .......A .m.....w
0000 20D0: 60 EC BF 77 E7 90 0A 85  88 14 2E 49 3E 22 F5 05  `..w.... ...I>"..
0000 20E0: 96 B0 C1 3A 93 23 4C 51  7C 7A BB CD C3 19 09 7F  ...:.#LQ |z......
0000 20F0: B2 8F 34 59 B7 0E B4 F2  75 43 10 D5 5B 22 7D 86  ..4Y.... uC..["}.
0000 2100: 0E 93 D1 03 74 37 BB D1  1C C9 DF 95 EC 7C 73 37  ....t7.. .....|s7
0000 2110: 83 90 A9 EF 89 A1 2B 12  BB 52 38 C2 FB 08 8F DC  ......+. .R8.....
0000 2120: 55 52 47 D6 9B 97 4F F1  3A 01 87 DC C6 50 18 95  URG...O. :....P..
0000 2130: D7 0E 75 E0 17 83 32 A0  19 3D 46 5A DC 44 88 DF  ..u...2. .=FZ.D..
0000 2140: E4 D0 84 89 86 FC 9B BD  FA D7 F1 BE C5 79 EF C4  ........ .....y..
0000 2150: 96 2D D2 5C 5C F4 4C E8  24 83 93 CB 12 B1 18 04  .-.\\.L. $.......
0000 2160: 94 BD 16 44 49 C3 54 36  76 A6 4A D1 5D 4C BE E0  ...DI.T6 v.J.]L..
0000 2170: FF 60 7D 96 D3 DD 9C C7  9A 69 C0 60 C7 7F EB 8F  .`}..... .i.`....
0000 2180: DE F1 0E CB 7F C9 55 28  D7 23 7E 1F 98 10 00 4D  ......U( .#~....M
0000 2190: 53 8D CF 14 50 32 6C 6E  82 C6 E1 06 2B C6 22 B4  S...P2ln ....+.".
0000 21A0: 8A 23 ED EB F4 46 0F 15  02 EF 45 0A 77 59 A3 9B  .#...F.. ..E.wY..
```