AGENCE
DU NUMÉRIQUE
EN SANTÉ

Volet DESFire
CPS V4

Statut : Validé | *Classification : Publique* | *Version : 0.2*

Agence du Numérique en Santé
2 - 10 Rue d'Oradour-sur-Glane – 75015 Paris
T. 01 58 45 32 50
esante.gouv.fr

Destinataires

Prénom / Nom	Entité / Direction
Tous les collaborateurs	

Documents de référence

Historique du document

Version	Rédigé par		Vérifié par		Validé par	
0.1	ANS	21/10/2024		Le JJ/MM/AA	P.NOM	Le JJ/MM/AA
	Motif et nature de la modification : **Création du document**					
0.2	ANS	21/02/2025				
	Mise à jour / remontées terrain					
	Motif et nature de la modification :					
	Motif et nature de la modification :					
	Motif et nature de la modification :					
	Motif et nature de la modification :					

SOMMAIRE

1. INTRODUCTION

1.1. Objet du document

La technologie MIFARE DESFire est un standard de-facto pour des usages de contrôle d'accès physique (parking, locaux ...) et/ou logiques (accès au SI, restauration …). Cette technologie, présente sur la carte CPS V3, est très utilisée d'où la nécessité de la reconduire sur la carte CPS V4.

Le but de ce document, dans le cadre de la carte CPS V4, est de :

- Décrire les éléments de personnalisation électriques de l'application DESFire
- D'énoncer des recommandations (ANS) par rapport à l'exploitation de cette application sur le terrain.

Avant de décrire ces éléments, le chapitre suivant fait un rappel des éléments de cette même application sur le masque actuel (CPS V3) ainsi que les choix réalisés pour la CPS V4.

Ce document s'applique à la CPS V4 version R4V2 ou plus.

1.2. Éléments de référence

N° Document	Nom du document	Description
[ANS 1]	ASIP-PUSC-PSCE_NP_CPx-Sans-Contact_20190425_v3.3.0	

1.3. Terminologie

Terme, sigle, acronyme	Définition
AA	Active Authentication
ADF	Application Dedicated File
AID	Application Identifier
ANS	Agence Numérique pour la Santé
APDU	Application Protocol Data Unit
ATR	Answer To Reset
CPS	Carte de professionnel de santé
DDA	Dynamic Data Authentication
DF	Directory File
EF	Elementary File
PA	Passive Authentication
PICC	Proximity Integrated Circuit(s) Card
MAC	Message Application Cryptogram
MF	Master File
SDA	Static Data Authentication
SN	Serial Number

Notations :

- 0 à 9 : caractères décimaux.
- ‘0’ à ‘9’ et ‘A’ à ‘F’ : caractères hexadécimaux.
- ‘xxx’ : chaîne de caractères hexadécimaux

2. ELEMENTS DE CONTEXTE

2.1. Masque CPS V3

2.1.1. Généralités

Sur le masque CPS V3, deux technologies MIFARE sont embarquées :

- MIFARE CLASSIC
- DESFire EV1

Le MIFARE CLASSIC n'est pas reconduit car obsolète en termes de sécurité. Pour ce qui est du DESFire, on passe de DESFire EV1 à DESFire EV3.

 **Les solutions terrain qui s'appuient sur MIFARE CLASSIC doivent donc migrer, dans le cadre du déploiement du masque CPS V4, vers du DESFire.**

 Les solutions terrain qui s'appuient sur DESFire EV1 sont portables car NXP garanti une compatibilité ascendante EV1 vers EV3.

2.1.2. Personnalisation

Sur le masque V3, la politique de l'ANS est de fournir une application DESFire « vierge ». La configuration est la suivante :

- Une MASTER KEY de type 2K3DES 128 bits initialisée avec des '00'
- Modification configuration protégée par authentification avec la MASTER KEY
- Modification MASTER KEY, type et/ou valeur protégée par authentification avec la MASTER KEY
- Création application DESFire possible **sans authentification** avec la MASTER KEY
- Suppression protégée par authentification avec la MASTER KEY
- Format PICC protégée par authentification avec la MASTER KEY
- UID fixe sur 7 octets
- DESFire 8K

2.1.3. ATR

Il existe un seul ATR sans contact pour le masque CPS V3 :

‘3B8F8001 0031B86404B0ECC173940180829000 0E’

Interprétation des octets historiques :

Historical bytes	00 31 B8 64 04 B0 EC C1 73 94 01 80 82 90 00
Category indicator byte: 0x00	(compact TLV data object) Tag: 3, Len: 1 (card service data byte) Card service data byte: 184 - Application selection: by full DF name - BER-TLV data objects available in EF.DIR - BER-TLV data objects available in EF.ATR - EF.DIR and EF.ATR access services: by READ BINARY command - Card with MF Tag: 6, Len: 4 (pre-issuing data) Data: 04 B0 EC C1 "...." Tag: 7, Len: 3 (card capabilities) Selection methods: 148 - Short EF identifier supported - DF selection by file identifier - DF selection by full DF name Data coding byte: 1 - Behaviour of write functions: one-time write - Value 'FF' for the first byte of BER-TLV tag fields: invalid - Data unit in quartets: 1 Command chaining, length fields and logical channels: 128 - Command chaining - Logical channel number assignment: No logical channel - Maximum number of logical channels: 1 Mandatory status indicator (3 last bytes) LCS (life card cycle): 130 (Proprietary) SW: 90 00

3. DESFIRE CPS V4

3.1. Introduction

Comme déjà indiqué, le MIFARE CLASSIC n'est pas reconduit car obsolète en termes de sécurité. Pour ce qui est du DESFire, on passe de DESFire **EV1** à DESFire **EV3**.

3.2. ATR

Pour le masque CPS V4, le choix a été fait de d'avoir des octets historiques « neutres » afin de rendre le volet DESFIRE complètement indépendant des volets « métiers ».

Carte	ATR
CPS V3 CL	'**3B** 8F8001 0031B8**6404B0ECC1**73940180829000 **0E**'
CPS V4 CONTACT	'**3B** DC18FF00 00122500**6480000401**009000'
CPS V4 CONTACTLESS	'**3B** 818001 80 **80'** **octets historique H1 :** compact TLV data object

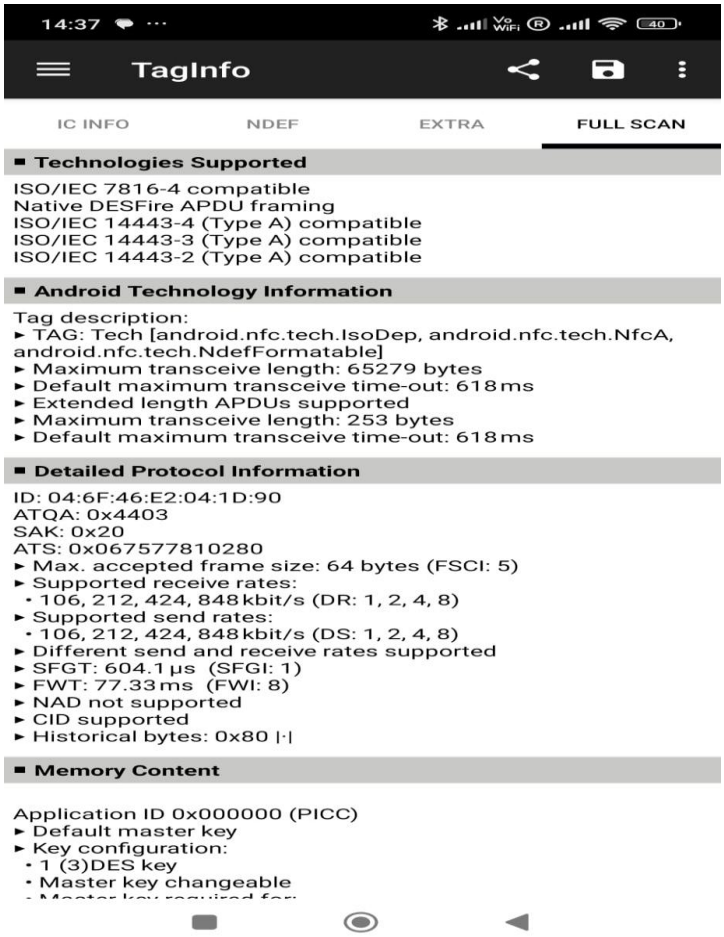
3.3. Configuration

Afin d'être compatible avec les solutions actuelles, on garde la même configuration, à savoir :

- Une MASTER KEY de type 2K3DES 128 bits initialisée avec des '00'
- Modification configuration protégée par authentification avec la MASTER KEY
- Modification MASTER KEY, type et/ou valeur protégée par authentification avec la MASTER KEY
- Création application DESFire possible **sans authentification** avec la MASTER KEY
- Suppression protégée par authentification avec la MASTER KEY PICC ou MASTER KEY application
- Format PICC protégée par authentification avec la MASTER KEY
- UID fixe sur 7 octets
- DESFire 8K
- AID applet DESFIRE = '00A4040007D276000085010000'

3.4. NFC Informations

Voici les informations liées à la configuration sans contact (ATQA, SAK …) lues au travers de l'application NXP TagInfo :



3.5. Reconnaissance d'une carte CPS V4

La reconnaissance d'une carte CPS V4 peut se faire au niveau électrique par :

- ✓ La présence de l'application ANS DESFIRE décrite un peu plus loin dans ce document (l'utilisation de l'ATR n'est pas conseillée car on utilise un ATR « neutre »)

ou bien au niveau graphique, sur le VERSO de la carte en haut à droite la version R4Vx avec x supérieur ou égale à 2 indique que la carte CPS est personnalisée avec le volet DESFIRE tel que décrit dans ce document.

3.6. Sélection du volet DESFIRE

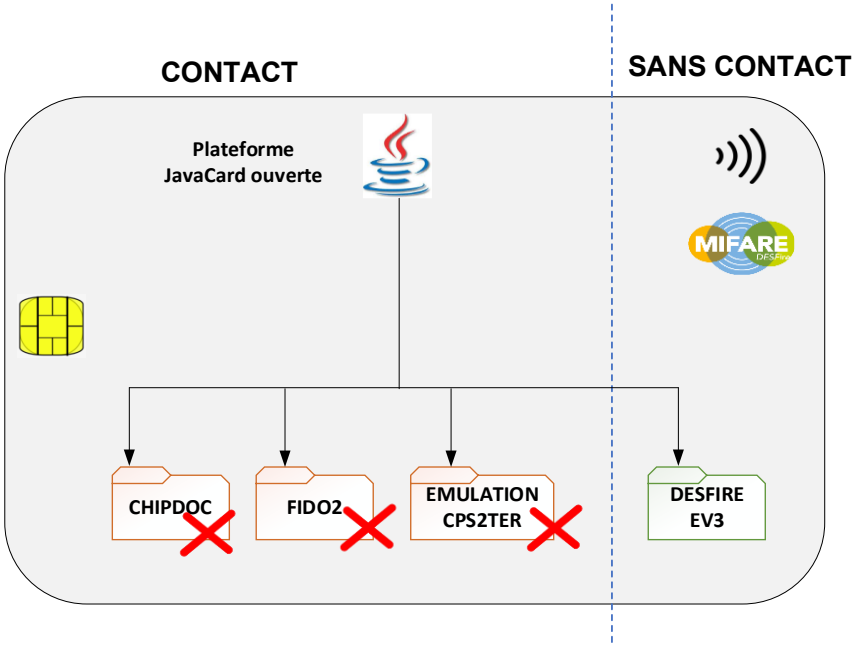
3.6.1. Introduction

L'implémentation de MIFARE DESFIRE est réalisée au travers d'une **émulation**. Comme pour la CPS V3, le chip retenu est un chip dual (contact et sans contact). Il est basé sur une plateforme JavaCard ouverte et les fonctionnalités de MIFARE DESFIRE sont disponibles au travers d'une **applet DESFIRE**.

Cette applet doit donc cohabiter avec les autres applets de la carte et notamment l'applet régalienne.

Avant de pouvoir travailler avec une applet, il faut au préalable la sélectionner. Il existe bien un mécanisme qui permet de sélectionner une applet par défaut après le reset (privilège GlobalPlateform), mais celui est dédié à une autre applet de la carte.

L'interface de communication sans contact est dédiée à l'applet DESFIRE tandis que l'interface contact est en quelque sorte partagée par trois autres applets : applet régalienne (CHIPDOC), applet FIDO2 et applet émulation CPS2TER.



Il est important de noter que toute solution qui met en œuvre l'applet DESFIRE de la CPS V4 doit s'assurer que cette applet est bien sélectionnée avant tout.

Avant d'aborder cette sélection, ci-dessous deux contraintes liées à l'implémentation de l'applet DESFIRE au sein de la carte CPS V4 :

- ✓ L'applet DESFIRE ne peut être opérationnelle (sélectionnée) que sur le canal 0
- ✓ Lorsque l'applet DESFIRE est opérationnelle, aucune autre applet ne peut être sélectionnée sur un autre canal. Et de manière réciproque si une autre applet est sélectionnée sur le canal 1, l'applet DESFIRE ne peut pas être sélectionnée sur le canal 0.

Dans le cadre du projet CPS V4, la configuration de la plateforme utilisée (JCOP 4.5) est réalisée de telle sorte qu'un traitement particulier est réalisé lors de la réception de la 1ière commande reçue sur l'interface sans contact (l'autre configuration serait d'utiliser le privilège associé à l'applet qui ferait qu'elle serait automatiquement sélectionnée après chaque reset de la carte, mais cette option ne peut pas être utilisée pour des raisons « métier »).

Le traitement effectué sur la **première commande** est le suivant :

- S'il s'agit d'un SELECT BY NAME ('00A404 …') alors la plateforme va essayer de sélectionner l'applet en question. Attention, si ce SELECT échoue, la seule manière de sélectionner l'applet DESFIRE sera de renvoyer un SELECT BY NAME avec l'AID de l'applet DESFIRE.
- S'il s'agit d'une autre commande, cette commande est envoyée à l'applet DESFIRE et :
  - Si l'applet DESFIRE sait traiter cette commande, la plateforme sélectionne l'applet DESFIRE sur l'interface sans contact (canal 0) et donne la commande pour traitement
  - Dans le cas contraire, un mot d'état '6A82' est retourné.

3.6.2. Sélection applet DESFIRE

Dans le cas de l'utilisation d'un lecteur NFC avec le protocole spécifique de NXP, le problème de la sélection de l'applet DESFIRE ne se pose pas car la plateforme redirige systématique les commandes reçues vers l'applet DESFIRE. On a donc là un comportement comme si l'on avait une puce DESFIRE dédiée (et non pas émulée).

On se place ici dans le cas d'un lecteur sans contact conforme au protocole ISO 14443, on doit donc s'assurer que l'applet DESFIRE est sélectionner avant de l'utiliser.

La sélection de l'applet peut être :

- Explicite : envoi d'une commande SELECT BY NAME
- Implicite : envoi d'une commande DESFIRE

3.6.3. Sélection explicite

La sélection explicite sur le canal de base (0) avec l'AID DESFIRE doit toujours pouvoir être réalisée. Attention, il s'agit là d'un SELECT BY NAME ('00A404 …') et non pas du SELECT DESFIRE (commande 'A5').

Ci-dessous quelques cas de sélection de l'AID DESFIRE par rapport au traitement de la première commande que réalise la plateforme.

CAS N°1: SELECT BY NAME avec AID DESFIRE (Cas nominal)

Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO]Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO]Power-on
[Event-REC]State:The card has been reset and specific communication protocols have been established.
[Event-REC]Protocol:Use of T=1 protocol

[Event-REC]ATR:3B8180018080

=1=> SELECT AID DESFIRE
APDU: 00A4040007D276000085010000 (13 bytes)
SW12: 9000 (Command successfully executed (OK))

=2=> GET VERSION DESFIRE
APDU: 9060000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04810043011A05

=3=> GET VERSION DESFIRE step 2
APDU: 90AF000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04814603001A05

=4=> GET VERSION DESFIRE step 3
APDU: 90AF000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (14 bytes)
046F46E2041D9021025000001524

[Event-ENDS]No Error.
>>>

CAS N°2: SELECT BY NAME avec AID connu (ISD = Issuer Security Domain)

```
>>> %Run cpsV4-desfire-reset.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO]Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO]Power-on
[Event-REC]State:The card has been reset and specific communication protocols have been established.
[Event-REC]Protocol:Use of T=1 protocol
[Event-REC]ATR:3B8180018080

=1=> SELECT ISD
APDU:  00A4040007A0000001510000 (12 bytes)
SW12: 9000 (Command successfully executed (OK))

=2=> GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 6E00 (Class not supported)

=3=> SELECT AID DESFIRE
APDU:  00A4040007D276000085010000 (13 bytes)
SW12: 9000 (Command successfully executed (OK))

=4=> GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04810043011A05

=5=> GET VERSION DESFIRE step 2
APDU:  90AF000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04814603001A05

=6=> GET VERSION DESFIRE step 3
APDU:  90AF000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (14 bytes)
046F46E2041D9021025000001524

[Event-ENDS]No Error.
>>>
```

=2=

Cette commande n'étant pas la 1ière commande, elle n'est pas redirigée ver l'applet DESFIRE, et donc pas de sélection de l'applet DESFIRE. Dans ce cas, c'est l'ISD qui va traiter la commande (CLA INCONNU).

=3=

Il faut refaire un SELECT BY NAME pour sélectionner l'applet DESFIRE.

CAS N°3: SELECT BY NAME avec AID inconnu

```
>>> %Run cpsV4-desfire-reset.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO]Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO]Power-on
[Event-REC]State:The card has been reset and specific communication protocols have been established.
[Event-REC]Protocol:Use of T=1 protocol
[Event-REC]ATR:3B8180018080

= 1 => SELECT Not Found
APDU:  00A4040007112233445566 77 (12 bytes)
SW12: 6A82 (File not found)

= 2 => GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 6A82 (File not found)

= 3 => SELECT AID DESFIRE
APDU:  00A4040007D276000085010000 (13 bytes)
SW12: 9000 (Command successfully executed (OK))

=4=> GET FREE MEMORY
APDU:  906E000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (3 bytes)
201C00

[Event-ENDS]No Error.
>>>
```

=2=

Cette commande n'étant pas la 1ière commande, elle n'est pas redirigée ver l'applet DESFIRE, et donc pas de sélection de l'applet DESFIRE. Dans ce cas, aucune applet n'est sélectionnée sur cette interface, la plateforme retourne '6A82'.

Il est à noter que :

- Si au lieu de l'AID inconnu on utilise l'AID de l'applet CHIPDOC (application régalienne CPS V4), FIDO2 ou émulation CPS2TER, on aura le même comportement car ces applets ont été désactivées sur cette interface.

- Aucun test de SELECT BY NAME n'est réalisé sur le canal 1 car les seules applets opérationnelles sur l'interface sans contact (DESFIRE, ISD) ne fonctionnent que sur le canal de base (canal 0).

3.6.4. Sélection implicite

La sélection implicite, quant à elle, va se reposer sur le mécanisme de la plateforme pour sélectionner automatiquement l'applet DESFIRE.


Dans la mesure du possible, il ne faudra pas utiliser ce mécanisme. En effet, il se peut que d'autres composants logiciels (OS, autres applications …) accèdent à la ressource matérielle et envoient des commandes sur l'interface sans contact. Dans ce cas, on ne peut plus se baser sur une sélection au travers de la première commande car celle-ci aura déjà été envoyée. **ATTENTION les commandes qui ne sont autorisées qu'après authentification (exemple : getCardUID) sont considérées comme commandes invalides et la sélection de l'applet DESFire n'est pas effectuée.**

Ci-dessous quelques cas de sélection de l'AID DESFIRE par rapport au traitement de la première commande que réalise la plateforme.

CAS N°4 : Commande non reconnue par applet DESFIRE

```
>>> %Run cpsV4-desfire-reset.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO]Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO]Power-on
[Event-REC]State:The card has been reset and specific communication protocols have been established.
[Event-REC]Protocol:Use of T=1 protocol
[Event-REC]ATR:3B8180018080

=1=> GET DATA Applet Name CHIPDOC
APDU:  00CA010000 (5 bytes)
SW12: 6A82 (File not found)

=2=> GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 6A82 (File not found)

=3=> SELECT AID DESFIRE
APDU:  00A4040007D276000085010000 (13 bytes)
SW12: 9000 (Command successfully executed (OK))

=4=> GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04810043011A05

[Event-ENDS]No Error.
>>>
```

=1=
Cette commande n'étant pas reconnue par DESFIRE, la sélection implicite n'a pas lieu et la commande suivante (DESFIRE) GET VERSION n'est pas reconnue.

CAS N°5 : Commande reconnue par applet DESFIRE

```
>>> %Run cpsV4-desfire-reset.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO]Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO]Power-on
[Event-REC]State:The card has been reset and specific communication protocols have been established.
[Event-REC]Protocol:Use of T=1 protocol
[Event-REC]ATR:3B8180018080

=1=> GET FREE MEMORY
APDU:  906E000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (3 bytes)
201C00

=2=> GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04810043011A05

=3=> GET VERSION DESFIRE step 2
APDU:  90AF000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04814603001A05

=4=> GET VERSION DESFIRE step 3
APDU:  90AF000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (14 bytes)
046F46E2041D9021025000001524

=5=> SELECT app ANS
APDU:  00A4000002A00000 (8 bytes)
SW12: 9000 (Command successfully executed (OK))

[Event-ENDS]No Error.
>>>
```

=1=

Cette commande est la 1ière commande envoyée sur l'interface sans contact et elle reconnue par l'applet DESFIRE ➔ cette dernière est alors implicitement sélectionnée, et les commandes DESFIRE suivantes sont correctement exécutées : GETVERSION et SELECT APPLICATION ANS.

CAS N°5 : Commande invalide

```
>>> %Run cpsV4-desfire-reset.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> GET CARd UID
APDU:  9051000000 (5 bytes)
SW12: 6A82 (File not found)

==> SELECT app ANS
APDU:  00A4000002A00000 (8 bytes)
SW12: 6A82 (File not found)

[Event-ENDS] No Error.
>>>
```

3.6.5. Recommandations sélection applet DESFIRE

Quelques recommandations quant à la sélection de l'applet de DESFire :

	Recommandation ANS
RS01	Si cela est possible, utiliser toujours la sélection explicite SELECT BY NAME avec l'AID DESFIRE
RS02	Ouvrir une session sur le lecteur sans-contact en mode EXCLUSIF (pour la carte CPS V4, c'est la seule applet opérationnelle sur l'interface sans contact)

3.7. Application DESFire ANS

3.7.1. Introduction

Lors des ateliers de cadrage de la CPS V4, la phase d'enrôlement (utilisation du sans contact) a été évoquée, et notamment la possibilité de réaliser un pré enrôlement lors de la production de la carte afin de rendre cet enrôlement plus simple.

Pour rappel, la phase d'enrôlement est l'étape qui consiste à lier une carte physique (CPx) à un titulaire afin de lui octroyer :

- L'accès à une ou plusieurs zones sécurisées (parking …)
- L'accès à un ou plusieurs services (session informatique …)

Cette association va consister en général à inscrire au sein d'un serveur (serveur de contrôle d'accès, SSO …) un couple d'identifiants :

- Un premier identifiant accessible en sans contact (lié à la carte physique)
- Un second qui identifie le titulaire de la carte

Une fois ce lien inscrit, le titulaire peut ainsi accéder aux zones/services :

- Présentation de la carte sur un lecteur sans contact, lecture de l'identifiant accessible en sans contact
- Remontée vers le serveur qui retrouve le lien inscrit et donc le titulaire de la carte
- Accès ou non aux ressources selon les droits inscrits pour ce titulaire

 Pour rappel, les règles de sécurité induites par le sans contact imposent à ce que les données pouvant être lues à l'insu du porteur ne permettent pas de relier cette carte à son propriétaire (**Protection des données Personnelles**). C'est la raison pour laquelle, la carte CPS V3 n'est pas liée à une personne physique (données accessibles en sans contact : numéro de série carte, certificat anonyme) d'où la nécessité de cette phase d'enrôlement.

Avant de voir ce qu'il est possible de faire sur le masque V4, un point sur les différents identifiants qui existent dans le cadre des cartes CPx.

3.7.2. Identifiants

Afin de faciliter la phase d'enrôlement, il est important de lister l'ensemble des identifiants pouvant être utilisés dans le cadre de la CPS :

- UID identifiant unique associé à la carte (UID en type A, sur 7 octets)
- SN numéro de série de la carte
- IDCARD identifiant de la carte
- IDNAT identifiant unique associé au titulaire de la carte
- ID LOCAL géré en local par l'établissement / éditeur de solution

Pour chaque masque, les colonnes DESFIRE, Contact et Sans contact correspondent respectivement à l'application DESFIRE, l'application régalienne contact (IAS ou CHIPDOC) et l'application régalienne sans contact (IAS ou CHIPDOC). Lorsqu'une case est grisée, cela signifie que l'application concernée n'est pas activée pour cette interface. Il n'existe pas de colonne DESFire Contact car le DESFire est toujours sur le sans contact. Lorsqu'une case est non grisée, cela indique que l'identifiant est accessible, et on indique alors dans cette case que cet identifiant est :

- Privé c'est-à-dire non accessible de manière publique (via site ANS ou autre)
- Public c'est-à-dire que cet identifiant est publié dans l'annuaire ANS (et présent dans certificat)

Identifiant	Masque CPS V3				Masque CPS V4			
	DESFire	Contact	Sans contact	Imprimé	DESFire	Contact	Sans contact	Imprimé
UID	Privé			Non	Privé			Non
SN		Privé	Privé	Non	Oui	Privé		Non
IDCARD		Public		Oui	Configurable	Public		Oui
IDNAT		Public		Oui	Configurable	Public		Oui
ID LOCAL	(1)			(2)	(1)			(2)

REMARQUES

- (1) indique qu'il est possible que les établissements gèrent un identifiant local qu'ils inscrivent dans l'application DESFire durant la phase d'enrôlement.

- (2) certains établissements réalisent une seconde personnalisation graphique, ils pourraient du coup imprimer un identifiant local.

- La différence entre IDNAT et IDCARD est que IDNAT identifie le titulaire alors que IDCARD identifie une carte associée à ce titulaire. Si un titulaire perd une carte, l'ANS lui renverra une autre carte avec le même IDNAT mais avec un IDCARD différent.

- Pour information, le code barre présent sur les enveloppes d'expédition CPS est codé avec l'identifiant IDCARD afin de permettre l'enrôlement sans avoir à ouvrir l'enveloppe.

Proposition de pré-enrôlement

On parle de pré-enrôlement car la phase qui consiste à inscrire le couple des identifiants :

Identifiant privé	accessible en sans contact
Identifiant public	permet de retrouver toutes les informations liées à la carte (titulaire …) dans l'annuaire ANS

Devra se faire au sein du serveur d'accès, SSO ….

Identifiant privé

On se propose d'utiliser, comme sur le masque V3 l'identifiant SN. L'application DESFire du masque V4 aura un fichier SN (identique à la partie contact). Ce fichier sera accessible en lecture libre. Pour information, sur le masque V3, des lecteurs très utilisés tel que STID utilisent soit l'UID soit cet identifiant. Mais sur la carte CPS V3, SN n'est pas accessible au travers du DESFire mais de la partie régalienne (IAS) sans contact.

Identifiant public

Bien qu'un seul identifiant suffise, on propose d'inscrire les deux identifiants IDCARD et IDNAT. Comme pour le SN (identifiant privé), ces données sont en lecture libre.

Afin d'être en conformité avec la CNIL, la solution ANS se rapproche des solutions « cartes bancaires » et pour ce faire, il faut :

- Indiquer au titulaire la liste des données personnelles accessibles en sans contact, à savoir IDNAT et IDCARD
- Indiquer au titulaire que par défaut ces données sont accessibles mais avec la possibilité :
    - De les désactiver (cela se fera au travers de l'outil public CPS GESTION, disponible sur les stores Microsoft et Apple)
    - De les protéger (authentification avec clé de lecture à gérer par les établissements, données chiffrées lors de la lecture). Cette protection pourra aussi être activée/désactivée à l'aide de CPS GESTION

SOLUTION PROPOSEE

Afin de garder une compatibilité avec l'existant, on propose de mettre tous les identifiants sous une application **DESFire ANS**. Chaque identifiant est stocké dans un fichier standard (binaire), en accès libre par défaut.

La configuration (taille, accès …) de cette application/fichiers est détaillée dans la suite de ce document.

3.7.3. Fichier DATA

Le choix de supprimer l'interface sans contact des applications autres que DESFire à un effet de bord : en effet, une des utilisations du masque V3 est la suivante :

- Insertion carte en contact et signature d'un jeton
- Inscription de ce jeton dans le fichier DATA, partagé en contact et sans contact (régalien)
- Lecture de jeton en sans contact (fichier DATA) …. Qui donne accès alors à des services informatiques …

Ce use case jeton CT/CL n'est plus possible du fait de la suppression de l'interface sans contact sur la partie régalienne.

Nous proposons une solution alternative qui va consister à inscrire le jeton dans l'application DESFire ANS. En fait le fichier DATA est aussi conservé dans la partie régalienne car certains éditeurs nous ont indiqué utiliser le jeton dans un cas CT/CT (jeton construit en CT puis relu en CT pour donner accès à des services).

 Des éditeurs nous ont remonté le fait que sur le masque V3, si la carte est insérée en contact, il n'était pas possible d'utiliser l'interface sans contact. Compte tenu de la configuration du masque CPS V4, il n'est pas possible, même sur la CPS V4 de travailler en simultané sur les interfaces contact et sans contact (à partir du moment où DESFire est opérationnel sur l'interface sans contact canal 0, aucun autre canal ne peut être utilisé et donc aucune autre applet sur n'importe quelle interface). Cependant il est possible :

- Si l'on dispose d'un lecteur dual et
- Qu'on ait accès aux 2 interfaces (contact et sans contact) lorsque la carte est insérée en contact

De travailler tantôt en CT puis CL …. Sans avoir à manipuler à la carte. Des exemples sont donnés an annexe (Use case JETON et exemples de pré enrôlement).

Contraintes :

- Fichier DATA sous DESFire ANS de 300 octets max (il était de 4096 octets sous CPS V3)
- Possibilité d'utiliser l'interface sans contact alors que le masque est inséré en contact, sinon l'interface « homme-machine » devra gérer avec l'opérateur le fait de retirer la carte du lecteur contact pour la mettre dans le lecteur sans contact (voir exemples en annexe)
- Les établissements / éditeurs qui ont mis en place cette solution devront la modifier pour le masque V4 avec en plus potentiellement un changement de lecteurs (lecteurs qui permettent d'utiliser l'interface sans contact alors que la carte est insérée dans le lecteur contact)

AUTRE ALTERNATIVE

Une autre solution pourrait consister à ne pas inscrire le jeton dans la carte. En effet dans la mesure où des données accessibles en sans contact permettent de retrouver la carte/titulaire, on pourrait imaginer que le jeton soit stocké sur un serveur d'accès, SSO … et lors de la lecture des identifiants sans contact, on retrouve la carte concernée, puis le ou les jetons pour cette carte ….

3.7.4. Authentification statique

Dans le cadre de l'application régalienne, un mécanisme d'authentification va être mis en place. Il s'agit d'un mécanisme identique à ce que l'on peut trouver :

- Dans le monde du Transport : PA = Passive Authentication
- Dans le monde du Bancaire : SDA = Static Data Authentication

Ce principe permet de s'assurer que les données qu'on lit sur la carte correspondent à des données issues d'une vraie carte. C'est pour cela qu'on parle d'authentification statique ou passive. Elle ne permet pas de contrer le rejeu à l'identique (« Yes Carte ») mais constitue néanmoins une garantie sur la véracité des données qu'on reçoit et que l'on va utiliser.

Le principe est simple :

- On dispose d'un certificat/clé privée ANS qui permet d'authentifier les données cartes CPx
- Le certificat est public, la clé privée n'est connue que de ING (qui produit les cartes)
- Lors de la phase de personnalisation, les fichiers SN, IDCARD, IDNAT sont signés (condensé puis signature)
- Cette signature est stockée sous un fichier SDA stocké sous l'application ANS DESFIRE.

En même temps que la signature, on stocke une référence qui pointe sur le certificat utilisé. On peut imaginer en effet, que pour des raisons de sécurité, on change la clé de signature :

- Tous les ans,
- Quand on produit plus de 1M de cartes
- …

La mise en œuvre de l'authentification passive va donc consister en :

1) Lecture des données : SN, IDCARD et IDNAT
2) Calcul condensé (algorithme de hachage dans certificat)
3) Lecture référence certificat puis récupération de ce certificat auprès de l'ANS
4) Vérification du certificat, si OK, on dispose d'une clé publique vérifiée
5) Lecture signature, vérification avec la clé publique de 4)
6) Extraction du condensé puis comparaison avec condensé calculé en 2)

3.7.5. Application DESFire

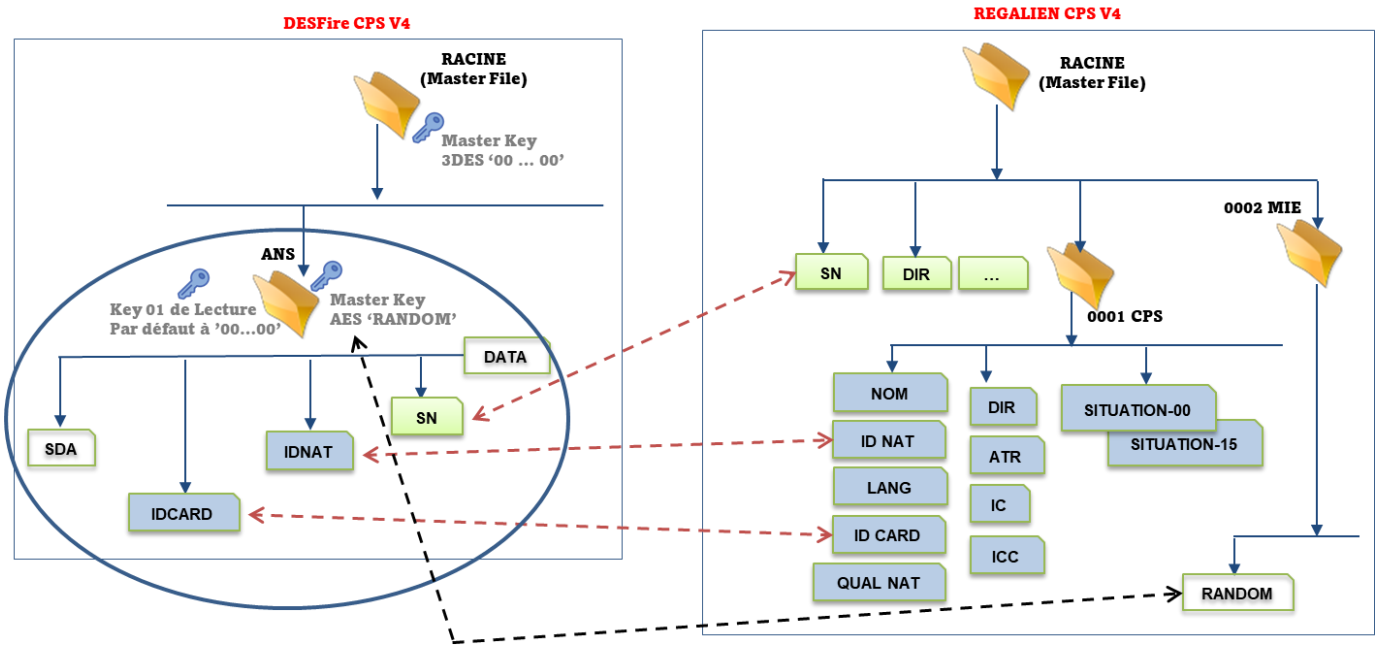
Il n'est évidemment pas question ici de décrire les fonctionnalités du DESFire, juste quelques éléments simples qui nous permettront ensuite de définir le contenu de l'application DESFire ANS.

- Le DESFire peut être assimilé à une arborescence fichiers possédant une racine, des répertoires, et des fichiers (sous la racine il ne peut y avoir que des répertoires, sous un répertoire que des fichiers)
- La racine est là par défaut (pas de création nécessaire), souvent nommée PICC dans la terminologie DESFire
- Chaque répertoire est ce qu'on appelle une application DESFire, la racine est elle-même un répertoire (qui n'a pas de parent)
- A chaque application (répertoire) est associé un ensemble de clés symétriques (au moins une, la MASTER KEY). Ces clés peuvent être de type TDES ou AES 128 bits
- Des fichiers peuvent être créés au sein d'une application, L'accès à ces fichiers peut être libre ou bien protégé par une des clés de l'application elle-même ou bien par une des clés de la racine.

Configuration application DESFire ANS :

- Une MASTER KEY de type AES 128 bits initialisée avec le fichier **RANDOM** de la partie régalienne
- Une clé de lecture de type AES 128 bits initialisée avec des '00'
- Modification configuration protégée par authentification avec la MASTER KEY ANS
- Modification MASTER KEY ANS, type et/ou valeur protégée par authentification avec la MASTER KEY ANS
- Création fichier DESFire possible sans authentification avec la MASTER KEY ANS
- Suppression fichier par authentification avec la MASTER KEY ANS
- AID et ISO DFname sur 3 octets '**414E53**' (correspond à « ANS » en ASCII).
- ISOFileID = '**A000**'

Application ANS :



Fichier RANDOM

Ce fichier, créé lors de la personnalisation, est en WRITE NEVER, READ PIN. Il est donc protégé par le PIN du porteur. La MASTER KEY de l'application ANS est initialisée avec ce RANDOM de 16 octets. Ce mécanisme permet de réaliser l'administration de l'application DESFIRE ANS sous l'autorisation du titulaire de la carte (puisque pour s'authentifier il faudra, au préalable lire ce RANDOM et donc présenter le PIN).

Eléments de personnalisation électrique

Ce fichier, créé lors de la personnalisation, est en WRITE NEVER, READ PIN. Il est donc protégé par le PIN du titulaire de la carte.

Champ	Fichier SN	Fichier IDCARD	Fichier IDNAT	Fichier SDA	Fichier DATA
File N°	'03'	'02'	'01'	'04'	'05'
ISO File ID	'A003'	'A002'	'A001'	'A004'	'A005'
File options	'00'	'00'	'00'	'00'	'00'
Access Right	'EFFF'	'EFF0'	'EFF0'	'EFFF'	'E0FF'
Size	12	7	33	300	300

Pour ce qui est des conditions d'accès :

SN et SDA

- Lecture en clair des fichiers
- Lecture LIBRE

IDCARD et IDNAT

- Lecture en clair des fichiers
- Lecture LIBRE
- Configuration pouvant être changée après authentification avec master key application
  - ⇨ **Cela veut dire qu'on peut protéger et/ou désactiver l'accès à ces données, faire du SM …**

DATA

- Lecture en clair des fichiers
- Lecture LIBRE
- Update après authentification avec master key application


ATTENTION : le DF Name '414E53' n'est opérationnel sur l'application DESFire que à partir de la version R4V4. Pour les versions précédentes R4V2 et R4V3, il faut donc réaliser un select DESFire : '90'

Dans la mesure du possible, il ne faudra pas utiliser ce mécanisme. En effet, il se peut que d'autres composants logiciel (OS, autres applications …) accèdent à la ressource matérielle et envoient des commandes sur l'interface sans contact. Dans ce cas, on ne peut plus se baser sur une sélection au travers de la première.

4. RECOMMANDATIONS

Quelques recommandations quant à l'utilisation de l'application DESFire :

	Recommandation ANS
RA01	Changer MASTER key racine : valeur et type (AES)
RA02	Changer clé de lecture ANS : valeur, on a '00...00' par défaut
RA03	Pour chaque solution (application DESFire) clé différente, privilégier AES
RA04	Chaque fois que cela est possible, privilégier commandes ISO plutôt que commandes natives
RA05	Identifiant unique : privilégier SN plutôt que UID
RA06	Afin d'être en conformité avec la CNIL, une fois les données du titulaire lues (en sans contact), réaliser l'enrôlement du titulaire puis protéger l'accès à ces données. Des exemples sont donnés en Annexe : lecture de ces données sous canal sécurisée (chiffrées) ou bien suppression de ces données.

5. ANNEXES

5.1. Fichiers

5.1.1. Fichier SN

Fichier standard de 12 octets, on conserve le même format que l'application régalienne, à savoir :

'5A 0A 80 250 00001 xxxxxxxx y F'

Champ	Description
'5A'	Tag
'0A'	Longueur champ données (10 octets)
'80'	Catégorie émetteur, 80 = secteur SANTE
'250'	Code pays, 250 = France
'00001'	Identifiant émetteur
'xx … xx'	Numéro de série composant sur 8 chiffres
'y'	Clé de Luhn
'F'	Octet de padding

5.1.2. Fichier IDCARD

Fichier standard de 7 octets, on conserve le même format que l'application régalienne, à savoir 10 chiffres codés en DCB (correspond au tag 'E3/81' du fichier IDCARD de la partie régalienne (FID 'D101'), on reprend le tag '81' :

'8105XXXXXXXXXX'

5.1.3. Fichier IDNAT

Fichier standard de 33 octets, on conserve le même format que l'application régalienne, à savoir 10 à 31 caractères alphanumériques (correspond au tag 'ED/81' du fichier INFO_PS de la partie régalienne (FID 'D104'):

'81nnXXXXXXXXX …'

5.1.4. Fichier DATA

Fichier standard de 300 octets. Son contenu est laissé à la discrétion des utilisateurs. Si une taille plus petite est utilisée, afin d'optimiser les performances lors de la lecture, il serait judicieux de structurer le fichier ainsi :

Champ	Description
Octets 0-1	Longueur sur 2 octets MSB-LSB (données utiles de 255 octets donnerait '00FF')
Octets 2-299	Données (jeton …)

5.1.5. Fichier SDA

Fichier SDA constitué :

- D'un identifiant de certificat (certificat ANS pour Authentification des données cartes CPS)
- De la signature des données suivantes :

  ✓ Contenu du fichier SN
  ✓ Contenu du fichier IDCARD
  ✓ Contenu du fichier IDNAT

Signature RSA (PSS 2048 bits ou ECC, toutes les informations sont contenues dans le certificat).
La taille de l'identifiant et celle de la signature sont stockées dans le certificat :

- Identifiant = empreinte numérique sur 20 octets (SHA1)
- Taille signature déduite de clé publique, algorithme de signature …

5.2. Exemples

5.2.1. Introduction

Tests réalisés avec :

- Lecteur dual OMNIKEY 5321
- Carte CPS V4 R4V2 ou R4V3

5.2.2. Lecture VERSION / UID

```
>>> %Run cpsV4-desfire-version.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: tmaj-desfire-version.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04810043011A05

==> GET VERSION DESFIRE step 2
APDU:  90AF000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04814603001A05

==> GET VERSION DESFIRE step 3
APDU:  90AF000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (14 bytes)
043E9AAA14199020760600002823

Hardware related informations:
 - Vendor ID (04: NXP)  = 04
 - HW type              = 81
 - HW sub-type          = 00
 - HM major version     = 43
 - HM minor version     = 4301
 - Storage size         = 1A
 - Protocole            = 05

Software related informations:
 - Vendor ID (04: NXP)  = 04
 - SW type              = 81
 - SW sub-type          = 46
```

- SW major version = 03
- SW minor version = 0300
- Storage size = 1A
- Protocole = 05

Protocole related informations:
- UID = **043E9AAA141990**
- Batch number = 207606
- Type ID = 0000
- CW of production = 28
- Year of production = 23

[Event-ENDS] No Error.
>>>

5.2.3. Changement MASTER KEY RACINE (3DES vers AES)

Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR: 3B8180018080

==> GET KEY SETTING
APDU: 9045000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (2 bytes)
0F01

==> AUTHENTICATE
APDU: 901A0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (8 bytes)
CE93CA8ADBC80115

Key value= 00000000000000000000000000000000
Key type = 2K3DES
Key number (00=MASTER) = 00
randomB_encrypted = CE93CA8ADBC80115
randomB = c76778e6f859d318

randomA = E2CD97081A35E3D8
randomB_rotate = 6778e6f859d318c7
randomAB_encrypted = 0a011e6da2c3176dc66d19b45cdb7db7

==> AUTHENTICATE (AF)
APDU: 90AF0000100a011e6da2c3176dc66d19b45cdb7db700 (22 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
DFE9049D80AD8613

[Event-INFO] Authentification réussie.

SessionKey = E2CD9708c76778e61A35E3D8f859d318
Subkey 1 = 9FC40DADC7F597D090B998F9467B0042
Subkey 2 = 3F881B5B8FEB2FA1217331F28CF60003
Iv = 00000000000000000000000000000000

Input for CRC = C4801122334455667788 99AABBCCDDEEFF0000
CRC32 = 5C66F521

=== E2CD9708c76778e61A35E3D8f859d318 ===
Input for ENCRYPT = 112233445566778899AABBCCDDEEFF00005C66F521000000
Data encrypted = 98cf496e868d6dc9ad4a4d1c4295a4a5e8277339f782043c

==> CHANGE 2K3DES to AES
APDU: 90C40000198098cf496e868d6dc9ad4a4d1c4295a4a5e8277339f782043c00 (31 bytes)
SW12: 9100 (OK)

==> GET KEY SETTING
APDU: 9045000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (2 bytes)
0F81

==> AUTHENTICATE
APDU: 90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
EF919EA3E8A1671ADA95991DA999CF0F

Key value= **11223344556677889AABBCCDDEEFF00**
Key type = **AES**
Key number (00=MASTER) = 00
randomB_encrypted = EF919EA3E8A1671ADA95991DA999CF0F
randomB = 71b67964fdf087dfe5794259bea05ef2

randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = b67964fdf087dfe5794259bea05ef271
randomAB_encrypted = 1e1f72be20d9e019d812af5fd817f592c24ee279e5887350da3c4f83bcfb6ddc

==> AUTHENTICATE (AF)
APDU: 90AF0000201e1f72be20d9e019d812af5fd817f592c24ee279e5887350da3c4f83bcfb6ddc00 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
9E99AAB1AC8C312474EDA5B69BE1DDDC

[Event-INFO] Authentification réussie.

SessionKey = 0011223371b67964CCDDEEFFbea05ef2
Subkey 1 = C1C9C815B0FEAD09909724E28EE5D1CB
Subkey 2 = 8393902B61FD5A13212E49C51DCBA311
Iv = 00000000000000000000000000000000

[Event-ENDS] No Error.
>>>

5.2.4. Lecture Fichiers

Comme indiqué dans les recommandations, on privilégie les commandes ISO :

- Les fichiers DATA et SDA sont lues en 2 fois (300 octets)
- La première commande est un SELECT ISO par FID, **le SELECT BY NAME ne sera disponible que à partir de la version R4V4**

>>> %Run cpsV4-desfire-read.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU: 00A4000002A00000 (8 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ SN file
APDU: 00B0830000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (12 bytes)
5A0A80250000001030953290F

==> ISO READ IDCARD
APDU: 00B0820000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (7 bytes)
81053100603747

==> ISO READ IDNAT
APDU: 00B0810000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (33 bytes)
81143330423032353639303532F435045543030303031000000000000000000000000

==> ISO READ SDA file - step 1
APDU: 00B08400F0 (5 bytes)
SW12: 9000 (Command successfully executed (OK)) ➔ **SDA non initialisé, on ne récupère que des '00'**
Data from card: (240 bytes)
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000

==> ISO READ SDA file - step 2
APDU: 00B084F000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (60 bytes)
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000

==> ISO READ DATA file - step 1
APDU: 00B08500F0 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (240 bytes)
0007112233445566770000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000

==> ISO READ DATA file _ step 2
APDU: 00B085F000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (60 bytes)
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000

[Event-ENDS] No Error.
>>>

5.2.5. Update Fichier

Exemple de mise à jour fichier DATA et tentative de WRITE sur les autres fichiers IDNAT, IDCARD, SN et SDA :

- Authentification est conforme à une MUTUAL AUTHENTICATE ISO (EXTERNAL AUTHENTICATE puis INTERNAL AUTHENTICATE)
- Authentification nécessaire car UPDATE fichier DATA protégée par authentification avec la MASTER key application.
- Cet exemple essaie de faire un UPDATE sur les autres fichiers après authentification, on a '6982'. L'authentification avec la MASTER KEY ANS suppose qu'on a lu le fichier RANDOM de la partie régalienne ('F4AFC6172C86C9ED45B05C72B8C61176' pour cette carte)
- La première commande est un SELECT ISO par FID, le SELECT BY NAME ne sera disponible que à partir de la version R4V4

>>> %Run cpsV4-desfire-update.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU: 00A4000002A00000 (8 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ DATA file
APDU: 00B0850010 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (16 bytes)
00000000000000000000000000000000

==> ISO UPDATE DATA file
APDU: 00D6850009000711223344556688 (14 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **Normal WRITE protégé avec Master Key APP**

==> ISO READ DATA file
APDU: 00B0850010 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (16 bytes)
00000000000000000000000000000000

==> GET CHALLENGE
APDU: 0084000010 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (16 bytes)
8DCEC2DA6174CDF65F019BC5E89FFDE9

==> EXTERNAL AUTH
APDU: 0082098020609c7a5c3e08add93b4e75f128ba5e924247d1ec13da7c8d05cbcf546046fa30 (37 bytes)
SW12: 9000 (Command successfully executed (OK))

==> INTERNAL AUTH
APDU: 008809801000112233445566778899AABBCCDDEEFF20 (22 bytes)
SW12: 9000 (Command successfully executed (OK)) **Authentification ok avec MASTER KEY APPLICATION**
Data from card: (32 bytes)
DC6C087845FE24BB83724B6884961DAC9D7F3BC6352E4E2D576DDF0F196EE2CB

==> ISO READ DATA file
APDU: 00B0850010 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (16 bytes)
00000000000000003EF4546A4BFA4C7B

==> ISO UPDATE DATA file
APDU: 00D6850009000711223344556677 (14 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ DATA file
APDU: 00B0850010 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (16 bytes)
00071122334455663CC7536B3864DFAB

==> ISO UPDATE IDCARD file
APDU: 00D68100015B (6 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **Normal WRITE NEVER**

==> ISO UPDATE IDNAT file
APDU: 00D68200015B (6 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **Normal WRITE NEVER**

==> ISO UPDATE SN file
APDU: 00D68300015B (6 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **Normal WRITE NEVER**

==> ISO UPDATE SDA file
APDU: 00D68400015B (6 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **Normal WRITE NEVER**

[Event-ENDS] No Error.
>>>

5.2.6. Désactivation des Identifiants

Exemple qui passe les fichiers IDNAT et IDCARD à « READ NEVER », ce qui consiste à désactiver l'accès aux données personnelles. Cela est possible après l'authentification avec la MASTER KEY de l'application ANS. Pour la carte utilisée, cette clé (fichier RANDOM de la partie régalienne) est '24A3BF5FC09DA45B13546A4E699F55BB'

Le SELECT BY NAME peut être utilisé sur cet exemple car il s'agit d'une carte R4V4 de test.

Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU: 00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ IDNAT file
APDU: 00B0810000 (5 bytes)
SW12: 9000 (Command successfully executed (OK)) ➔ **Par défaut, donnée accessible**
Data from card: (33 bytes)
810C383939373030363232313437000000000000000000000000000000000000

==> ISO READ IDCARD file
APDU: 00B0820000 (5 bytes)
SW12: 9000 (Command successfully executed (OK)) ➔ **Par défaut, donnée accessible**
Data from card: (7 bytes)
81053100603746

==> Change File Setting 01
APDU : 905F0000040100F0FF00 (10 bytes) ➔ **Changement de configuration protégé par authentification**
SW12: 91AE (Authentication status does not allow the requested command)

==> Change File Setting 02
APDU : 905F0000040200F0FF00 (10 bytes) ➔ **Changement de configuration protégé par authentification**
SW12: 91AE (Authentication status does not allow the requested command)

==> AUTHENTICATE
APDU: 90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
559951DCFB1228EC118E446ADDDFBB23

Key value= 24A3BF5FC09DA45B13546A4E699F55BB
Key type = AES
Key number (00=MASTER) = 00
randomB_encrypted = 559951DCFB1228EC118E446ADDDFBB23
randomB = de402c3a39205f35e3a83bb09d507289

randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = 402c3a39205f35e3a83bb09d507289de
randomAB_encrypted = 340fa964431f7e569e6a00b9ca2695e971e7775ceb584e6d8a587eff13ec78b9

==> AUTHENTICATE (AF)
APDU: 90AF000020340fa964431f7e569e6a00b9ca2695e971e7775ceb584e6d8a587eff13ec78b900 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
C732C86D49D3E2378C9CCDA9C1B289FC

[Event-INFO] Authentification réussie.

SessionKey = 00112233de402c3aCCDDEEFF9d507289
Subkey 1 = B6C732ED2F03A1DFAD898DAFCFC1D8CD
Subkey 2 = 6D8E65DA5E0743BF5B131B5F9F83B11D
Iv = 00000000000000000000000000000000
crc = AD399C02
=== 00112233de402c3aCCDDEEFF9d507289 ===

==> Change File Setting 01
APDU: 905F00001101e12e71080a4f6d732164e2be99cd212900 (23 bytes)
SW12 : 9100 (OK) ➔ **Changement de configuration (READ NEVER)**
Data from card: (8 bytes)
FA8FDBBCCB92690D

crc = 43962910
=== 00112233de402c3aCCDDEEFF9d507289 ===

==> Change File Setting 02
APDU: 905F0000110244a2c4846d005d60d3de6c5d93ded4da00 (23 bytes)
SW12: 9100 (OK) ➔ **Changement de configuration (READ NEVER)**
Data from card: (8 bytes)
D0B82404A381AFE9

[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU: 00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ IDNAT file
APDU : 00B0810000 (5 bytes) ➔ **Lecture impossible**
SW12: 6982 (Security condition not satisfied)

==> ISO READ IDCARD file
APDU: 00B0820000 (5 bytes) ➔ **Lecture impossible**
SW12: 6982 (Security condition not satisfied)

[Event-ENDS] No Error.
>>>

5.2.7. Activation des Identifiants

Cet exemple correspond au retour arrière par rapport à l'exemple précédent. La même carte est utilisée et l'on configure à nouveau les fichiers liés aux identifiants afin de pouvoir les lire.

>>> %Run cpsV4-desfire-activate.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO]
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU: 00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ IDNAT file
APDU: 00B0810000 (5 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **Lecture impossible**

==> ISO READ IDCARD file
APDU: 00B0820000 (5 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **Lecture impossible**

==> Change File Setting 01
APDU: 905F0000040100F0EF00 (10 bytes)
SW12: 91AE (Authentication status does not allow the requested command)

==> Change File Setting 02
APDU: 905F0000040200F0EF00 (10 bytes)
SW12: 91AE (Authentication status does not allow the requested command)

==> AUTHENTICATE
APDU: 90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
F0E637AD685ECDB389B70023C67F9E1C

Key value= 24A3BF5FC09DA45B13546A4E699F55BB
Key type = AES
Key number (00=MASTER) = 00
randomB_encrypted = F0E637AD685ECDB389B70023C67F9E1C
randomB = bfe9cc770d97f8dbe565dcad771a9217

randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = e9cc770d97f8dbe565dcad771a9217bf
randomAB_encrypted = 569f2b1f1383d4ea7b1ce4e95ba2aab9c9ec7398ca132073c7eb897836577359

==> AUTHENTICATE (AF)
APDU: 90AF000020569f2b1f1383d4ea7b1ce4e95ba2aab9c9ec7398ca132073c7eb89783657735900 (38 bytes)

SW12: 9100 (OK)
Data from card: (16 bytes)
ABD5816B53A8B145D873EE895E5A9061

[Event-INFO] Authentification réussie.

SessionKey = 00112233bfe9cc77CCDDEEFF771a9217
Subkey 1 = 5789E57A3902068C1F312DB62EB393C9
Subkey 2 = AF13CAF472040D183E625B6C5D672792
Iv = 00000000000000000000000000000000
crc = C9292B1F
=== 00112233bfe9cc77CCDDEEFF771a9217 ===

==> Change File Setting 01
APDU: 905F0000110122b097e55969da6854d2bf1f5fc959f400 (23 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
D3F599F4B91EE75D

[Event-INFO] CMAC ok
crc = 27869E0D
=== 00112233bfe9cc77CCDDEEFF771a9217 ===

==> Change File Setting 02
APDU: 905F00001102ed9a48da7822c9eda4ee706ea859e19300 (23 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
531755045FB5CF81

[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU: 00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ IDNAT file
APDU: 00B0810000 (5 bytes)
SW12: 9000 (Command successfully executed (OK)) ➔ **Lecture possible**
Data from card: (33 bytes)
810C383939373030363232313437000000000000000000000000000000000000

==> ISO READ IDCARD file
APDU: 00B0820000 (5 bytes)
SW12: 9000 (Command successfully executed (OK)) ➔ **Lecture possible**
Data from card: (7 bytes)
81053100603746

[Event-ENDS] No Error.
>>>

5.2.8. Protection des Identifiants

Toujours dans le souci de protéger l'accès aux identifiants, cet exemple montre comment protéger ces données. La lecture est alors possible mais est transmise au travers d'un canal sécurisé (Chiffrée, MAC). La clé de lecture est la clé 01 de l'application ANS. Par défaut, cette clé est initialisée à '00 … 00', à charge des utilisateurs de modifier cette valeur.

```
>>> %Run cpsV4-desfire-protect.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU:  00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ IDNAT file
APDU:  90BD000007010000021000000 (13 bytes)
SW12: 9100 (OK)
Data from card: (33 bytes)
810C383939373030363232313437000000000000000000000000000000000000000

==> ISO READ IDCARD file
APDU:  90BD000007020000007000000 (13 bytes)
SW12: 9100 (OK)
Data from card: (7 bytes)
81053100603746

==> Change File Setting 01
APDU:  905F0000040103F01F00 (10 bytes)
SW12: 91AE (Authentication status does not allow the requested command)

==> Change File Setting 02
APDU:  905F0000040203F01F00 (10 bytes)
SW12: 91AE (Authentication status does not allow the requested command)

==> AUTHENTICATE
APDU:  90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
8FBCF7F0333D63434629224DD981A500

Key value= 24A3BF5FC09DA45B13546A4E699F55BB
Key type = AES
Key number (00=MASTER) = 00
randomB_encrypted = 8FBCF7F0333D63434629224DD981A500
randomB = f8cf7e1325dbe65bbb04de474023c2f1
randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = cf7e1325dbe65bbb04de474023c2f1f8
```

randomAB_encrypted = 41ed74d29f3ac58b378b6324c87fdaf07a7026ae9f245e975a923fe222148074

==> AUTHENTICATE (AF)
APDU: 90AF00002041ed74d29f3ac58b378b6324c87fdaf07a7026ae9f245e975a923fe22214807400 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
5E3BFC02373651026F9A14166B7B29C6

[Event-INFO] Authentification réussie.

SessionKey = 00112233f8cf7e13CCDDEEFF4023c2f1
Subkey 1 = 6CFD1EE7996937CB5F044EDE67510CB1
Subkey 2 = D9FA3DCF32D26F96BE089DBCCEA21962
Iv = 00000000000000000000000000000000
crc = 8C65D0A0
=== 00112233f8cf7e13CCDDEEFF4023c2f1 ===

==> Change File Setting 01
APDU: 905F000011017ac3f03abf679fa21feef45f834528d500 (23 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes) ➔ **protection fichier par clé de lecture (01)**
ADA5341E84C6FBB2

[Event-INFO] CMAC ok
crc = 62CA65B2
=== 00112233f8cf7e13CCDDEEFF4023c2f1 ===

==> Change File Setting 02
APDU: 905F0000110249c329a86cd0dae7fccdd6d7f0190af900 (23 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes) ➔ **protection fichier par clé de lecture (01)**
8490C355626A7A40

[Event-INFO] CMAC ok

[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

==> ISO SELECT AID ANS
APDU: 00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ IDNAT file
APDU: 90BD0000070100000021000000 (13 bytes)
SW12: 91AE (Authentication status does not allow the requested command) ➔ **lecture impossible**

==> ISO READ IDCARD file
APDU: 90BD0000070200000007000000 (13 bytes)
SW12: 91AE (Authentication status does not allow the requested command) ➔ **lecture impossible**

==> ISO READ IDCARD file
APDU: 00B0820000 (5 bytes)
SW12: 6982 (Security condition not satisfied) ➔ **lecture impossible**

==> AUTHENTICATE
APDU: 90AA0000010100 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
98EAEA766CB475B6BCBABDF4A4AA1054

Key value= 00000000000000000000000000000000
Key type = AES
Key number (00=MASTER) = 01
randomB_encrypted = 98EAEA766CB475B6BCBABDF4A4AA1054
randomB = 170e2e800bb5f609130cc7fd34c9603f
randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = 0e2e800bb5f609130cc7fd34c9603f17
randomAB_encrypted = dde03810aaa54228620ae0cbd5b3098017a383ae8800e8d182575da362c3ef58

==> AUTHENTICATE (AF)
APDU: 90AF000020dde03810aaa54228620ae0cbd5b3098017a383ae8800e8d182575da362c3ef5800 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
04CFDA1B41E09E70397B6D2CCC2F2CFC

[Event-INFO] Authentification réussie. ➔ **avec clé de lecture (01)**

SessionKey = 00112233170e2e80CCDDEEFF34c9603f
Subkey 1 = 7F20050F34322D8DAC5E645CDD8700EE
Subkey 2 = FE400A1E68645B1B58BCC8B9BB0E01DC
Iv = 00000000000000000000000000000000

==> ISO READ IDNAT file
APDU: 90BD000007010000021000000 (13 bytes)
SW12: 9100 (OK) ➔ **lecture possible mais chiffrée**
Data from card: (48 bytes)
25D1D02C37A0B5710B2266FDEFAF9EECD88D89F022FB5A07D83317675E4CEC7CD4F9005ACD9321CE54E
2A602E593A0A0

=== 00112233170e2e80CCDDEEFF34c9603f ===
Decrypted RESPONSE = 810c383939373030363232313437000000000000000000000000000000000000000000

==> ISO READ IDCARD file
APDU: 90BD000007020000007000000 (13 bytes)
SW12: 9100 (OK) ➔ **lecture possible mais chiffrée**
Data from card: (16 bytes)
A86E4BBDFE44888A50315D6CF6F45F98

=== 00112233170e2e80CCDDEEFF34c9603f ===
Decrypted RESPONSE = 81053100603746

[Event-ENDS] No Error.
>>>

5.2.9. Création/Suppression application DESFIRE

Exemple de création/suppression d'application :

- Création de 2 applications : TEST 1 (AID '111111') et TEST 2 (AID '222222') possible sans authentification
- Suppression de la première application après authentification avec la MASTER KEY PICC
- Suppression de la seconde application après authentification avec la MASTER KEY application

```
>>> %Run cpsV4-desfire-delete-app.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: cpsv4-desfire-delete-app.py On reader: OMNIKEY CardMan 5x21-CL 0
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8580018073C821100E

==> GET FREE MEMORY
APDU:  906E000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (3 bytes)
C01C00

==> AUTHENTICATE (1A)
APDU:  901A0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (8 bytes)
8161AC66E5D0543A

Master key = 00000000000000000000000000000000
Key type = 2K3DES
randomB_encrypted = 8161AC66E5D0543A
randomB = 5810fabd3f7a64e8
randomA = E2CD97081A35E3D8
randomB_rotate = 10fabd3f7a64e858
randomAB_encrypted = 33d84d155fa32da591a5683b0cb20270

==> AUTHENTICATE (AF)
APDU:  90AF00001033d84d155fa32da591a5683b0cb2027000 (22 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
A37277F765AB9C14

[Event-INFO] Authentification réussie.

SessionKey = E2CD97085810fabd1A35E3D83f7a64e8

Create application TEST 1 with default master AES = '00 ... 00'
==> Create AID TEST 1
APDU:  90CA00000A111111E3A2A00111111100 (16 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
9BFAAC7024C081DF
```

Create application TEST 2 with AES master key = ['00', 'AES', '112233445566778899AABBCCDDEEFF00', '00']
==> Create AID TEST 2
APDU: 90CA00000A222222E3A2A00222222200 (16 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
9BA34369212D2AE0

Input for CRC = C4801122334455667788 99AABBCCDDEEFF0000
CRC32 = 5C66F521

=== E2CD97085810fabd1A35E3D83f7a64e8 ===
Input for ENCRYPT = 112233445566778899AABBCCDDEEFF00005C66F521000000
Data encrypted = 599d14197ee32e3fd24547369465852d2bad3ff696dbdf52

==> SELECT AID TEST 1
APDU: 905A00000311111100 (9 bytes)
SW12: 9100 (OK)

==> SELECT AID TEST 2
APDU: 905A00000322222200 (9 bytes)
SW12: 9100 (OK)

==> GET FREE MEMORY
APDU: 906E000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (3 bytes)
001C00

====> 1) Try to delete without authentication

[Event-INFO]Power-on
[Event-REC]State:The card has been reset and specific communication protocols have been established.
[Event-REC]Protocol:Use of T=1 protocol
[Event-REC]ATR:3B8580018073C821100E

==> DELETE AID ANS
APDU: 90DA00000311111100 (9 bytes)
SW12: 91AE (Authentication status does not allow the requested command)

==> DELETE AID ANS
APDU: 90DA00000322222200 (9 bytes)
SW12: 91AE (Authentication status does not allow the requested command)

====> 2) Try to delete after PICC MASTER KEY authentication

==> AUTHENTICATE (1A)
APDU: 901A0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (8 bytes)
B26A96E9D3D49668

Master key = 00000000000000000000000000000000
Key type = 2K3DES
randomB_encrypted = B26A96E9D3D49668
randomB = 06a9ea1f2ab29ffd

randomA = E2CD97081A35E3D8
randomB_rotate = a9ea1f2ab29ffd06
randomAB_encrypted = 066237c446905954b1c380701cff0318

==> AUTHENTICATE (AF)
APDU: 90AF000010066237c446905954b1c380701cff031800 (22 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
BABCB4C83905C623

[Event-INFO] Authentification réussie.

SessionKey = E2CD970806a9ea1f1A35E3D82ab29ffd

==> DELETE AID TEST 2
APDU: 90DA00000322222200 (9 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
06D360539EE06E19

====> 3) Try to delete after APP MASTER KEY authentication

[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8580018073C821100E

==> SELECT AID TEST 1
APDU: 905A00000311111100 (9 bytes)
SW12: 9100 (OK)

==> AUTHENTICATE (1A)
APDU: 90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
C62FF5370DF1E7870F4E3CD19708E132

Master key = 00000000000000000000000000000000
Key type = AES
randomB_encrypted = C62FF5370DF1E7870F4E3CD19708E132
randomB = 1496e4a706771d441fd5396378cad0c3
randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = 96e4a706771d441fd5396378cad0c314
randomAB_encrypted = cc9a4d20088b011b10fcebd89fa996de0577a36490e7b605fbee6dd219c5631c

==> AUTHENTICATE (AF)
APDU: 90AF000020cc9a4d20088b011b10fcebd89fa996de0577a36490e7b605fbee6dd219c5631c00 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
5E1F8E13BB113E58C258038070451A6A

[Event-INFO] Authentification réussie.

SessionKey = 001122331496e4a7CCDDEEFF78cad0c3

==> DELETE AID ANS
APDU: 90DA00000311111100 (9 bytes)
SW12: 9100 (OK)

==> SELECT AID TEST 1
APDU: 905A00000311111100 (9 bytes)
SW12: 91A0 (Requested AID not present on PICC)

==> SELECT AID TEST 2
APDU: 905A00000322222200 (9 bytes)
SW12: 91A0 (Requested AID not present on PICC)

[Event-ENDS] No Error.
>>>

5.2.10. Use case jeton CT/CL – exemple 1

Cet exemple montre l'utilisation d'un jeton généré en contact (application régalienne) puis sauvé sur le volet DESFire afin de pouvoir être récupéré en sans contact ensuite. On utilise la clé d'authentification, signature avec algorithme CKM_RSA_PKCS. Les trois étapes sont les suivantes :

- Etape 1 – CONTACT : création du jeton (signature du message), lecture de la clé DESFIRE (fichier RANDOM) nécessaire pour l'étape 2 et lecture clé publique (nécessaire pour l'étape 3). Cette étape est réalisée au travers de la Cryptolib (PKCS#11)
- Etape 2 – SAN CONTACT : Sauvegarde du jeton dans le ficher DATA.
- Etape 3 – SANS CONTACT : Lecture du jeton, contrôle avec la clé publique

STEP 1

```
>>> %Run cpsv4-jeton-step1.py

Infos token slot 0:
==============
firmwareVersion: 0.0
flags: CKF_RNG, CKF_LOGIN_REQUIRED, CKF_USER_PIN_INITIALIZED, CKF_TOKEN_INITIALIZED
hardwareVersion: 0.0
label: CPS4v1-3100640931
manufacturerID: ANS
model: IAS ECCNone

DESFire random:
==============
abb32a63ffeba58d706df301afcba72e

Jeton (signature):
==============
9dc2b556bcdf12d41e20a38f794c696ff0ccece14c0fcd22ebca8791597d30bcad31bba7ddeba0382cc8f855a819a6df
4d9307e40c81537786da1e8f6c2867d01418b6cc1c97c6320b77edf83ab81efedc79eeebb0eec9ed47db8d28ac6850
b11711a54e45c9d793aeee283d9afa3c2cdbda4248187f48d5081c0ae9eb5c68f9e1483d6b5be40eaf1711a339eee6
dc72fe9e01d7cbeeeb76476e2d64314d7935610a8ffd46c465ff92b31c3b30cbb8c15027b8c698d5e14b8aef66a9dcf3
3ae5d7d43514e6508ec32652543c53c7b696f84e42d27e7f18206a6382ed3e5add34ba4dbb450397fa297bddade24
611309598a0eae7e50d0ad82dd4022a817b8d65

PUBLIC key:
==========
Exponent = 010001
Modulus =
b62ea1e69ff25cac065ca4ecad0a3a74a7055bc8d67db1516193e86a7315b45ea582b75a13e977dd37a35960474ed
cae6ec6c586564fe2d093a49c9552a043ea44979d5788c605050e27c3bfa352ebea8ca5cf99d32c06fd4d885e1dbb1
a9417b6bb2583a2c79389ab7b9ba2c763fae76b54be1d00e8391309ee64cee2b04d06caeef650b208da360614cbb0
80a09ffaa4ccfdffc3eccbecf26ccd27ee645bd70cb73a7d32b738fcd786d4a0c82d55971e2b7323dcc2aa8113fb026b8
a208ba69184ef009d3a97a53111e68074cfe80fc7b108ef57f1942e282739d7267d3fc6c2609f7725aeda3124dc2741
876ee11410cff50f2406ce3ec0236e2ed11f8c35

===> PKCS#11 session closed
[Event-ENDS] No Error.
>>>
```

STEP 2

>>> %Run cpsv4-jeton-step2.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR : 3B8180018080

==> ISO SELECT FID ANS
APDU : 00A4000002A00000 (8 bytes)
SW12 : 9000 (Command successfully executed (OK)

==> AUTHENTICATE
APDU : 90AA0000010000 (7 bytes)
SW12 : 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
EB592E3861CAC28142BAF2C8174EA519

Key value= abb32a63ffeba58d706df301afcba72e
Key type = AES / Key number (00=MASTER) = 00
randomB_encrypted = EB592E3861CAC28142BAF2C8174EA519
randomB = 8a801bca7aa21e8c8babc61d042a7cae
randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = 801bca7aa21e8c8babc61d042a7cae8a
randomAB_encrypted = 2eee3590346ae7f2a5f6f7687eedc68d7f90a1803ff5206ca6af4e2bbf7b5fae

==> AUTHENTICATE (AF)
APDU : 90AF0000202eee3590346ae7f2a5f6f7687eedc68d7f90a1803ff5206ca6af4e2bbf7b5fae00 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
6C79DCB5A4DB004677784DE77B814DF0

[Event-INFO] Authentification réussie.
SessionKey = 001122338a801bcaCCDDEEFF042a7cae

==> ISO UPDATE DATA file - step 1
APDU:
00D685008101029dc2b556bcdf12d41e20a38f794c696ff0ccece14c0fcd22ebca8791597d30bcad31bba7ddeba038
2cc8f855a819a6df4d9307e40c81537786da1e8f6c2867d01418b6cc1c97c6320b77edf83ab81efedc79eeebb0eec9e
d47db8d28ac6850b11711a54e45c9d793aeee283d9afa3c2cdbda4248187f48d5081c0ae9eb5c68 (134 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO UPDATE DATA file - step 2
APDU:
00D6858181f9e1483d6b5be40eaf1711a339eee6dc72fe9e01d7cbeeeb76476e2d64314d7935610a8ffd46c465ff92b
31c3b30cbb8c15027b8c698d5e14b8aef66a9dcf33ae5d7d43514e6508ec32652543c53c7b696f84e42d27e7f18206
a6382ed3e5add34ba4dbb450397fa297bddade24611309598a0eae7e50d0ad82dd4022a817b8d65 (134 bytes)
SW12: 9000 (Command successfully executed (OK))

==========
Jeton généré en CT (2 + 256 octets) et sauvegarde en CL (fichier DATA)
==========

[Event-ENDS] No Error.

STEP 3

```
>>> %Run cpsv4-jeton-step3.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}
Use of T=1 protocol

======================================================================
En sans contact, lecture du jeton et déchiffrement
======================================================================

[Event-INFO]
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR: 3B8180018080

==> ISO SELECT AID ANS
APDU:  00A4000002A00000 (8 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ DATA file / LNG
APDU:  00B0850002 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (2 bytes)
0102

==> ISO READ DATA file / TOKEN 1
APDU:  00B0850280 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (128 bytes)
9DC2B556BCDF12D41E20A38F794C696FF0CCECE14C0FCD22EBCA8791597D30BCAD31BBA7DDEBA0382C
C8F855A819A6DF4D9307E40C81537786DA1E8F6C2867D01418B6CC1C97C6320B77EDF83AB81EFEDC79EE
EBB0EEC9ED47DB8D28AC6850B11711A54E45C9D793AEEE283D9AFA3C2CDBDA4248187F48D5081C0AE9
EB5C68F9

==> ISO READ DATA file / TOKEN 2
APDU:  00B0858280 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (128 bytes)
E1483D6B5BE40EAF1711A339EEE6DC72FE9E01D7CBEEEB76476E2D64314D7935610A8FFD46C465FF92B3
1C3B30CBB8C15027B8C698D5E14B8AEF66A9DCF33AE5D7D43514E6508EC32652543C53C7B696F84E42D2
7E7F18206A6382ED3E5ADD34BA4DBB450397FA297BDDADE24611309598A0EAE7E50D0AD82DD4022A817B
8D65

Message initial = Use case jeton CT/CL ... je suis un jeton pour carte CPS N°1234567 valable de 24/09/2025 09:00
a 24/09/2025 17:00
Jeton déchiffré = Use case jeton CT/CL via PKCS11 ... je suis un jeton pour carte CPS Num 1234567 valable de
24/09/2025 09:00 a 24/09/2025 17:00

[Event-ENDS] No Error.
>>>
```

5.2.11. Use case jeton CT/CL – exemple 2

Cet exemple est identique au précédent si ce n'est qu'on utilise le déchiffrement en lieu et place de la signature. En conséquence, pour l'étape 1, il n'est pas possible de passer par la Cryptolib car le déchiffrement, au travers de la Cryptolib, est conforme au standard PKCSv1.5. Du coup les données à chiffer doivent respecter un certain padding, ce qui n'est pas notre cas puisqu'en fait on utilise l'opération de déchiffrement pour « chiffrer » un message initial.

- On utilise la clé d'authentification, déchiffrement
- Le lecteur utilisé est dual et toutes les opérations ont été réalisées avec la carte insérée en contact

```
>>> %Run cpsv4-jeton-ctcl.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Script: cpsv4-jeton-ctcl.py On reader: OMNIKEY CardMan 5x21 0
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=0 protocol
[Event-REC] ATR :3BDC18FF000012250064800004010090000

Sélection de l'APPLICATION Régalienne
=====================================
APDU:  00A4040C088025000001FF0100 (13 bytes)
SW12: 9000 (Command successfully executed (OK))

selection DF authentication 0001/0102
=====================================
APDU:  00A408000400010102 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

[Event-INFO] Read AUTH public key

APDU:  00A4000002902000 (8 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (21 bytes)
6F13800300010D8203010000830290208603000303

APDU:  00B00000E7 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (231 bytes)
3082010902820100AD91F662C5C509EE7B6DC7D4893EFD3F2BE5C8E110B93EE481BDDD3A8E4459BB0B98
57BF6193BCC5D0195C6B0743DEE2E70A72BBC14228A4180E7E8C74B31B891B5513568B3FB35E4B2FB1767
604CC0C2C8ED00DD9E7620A54FDB8444F5090D6124B59D32F6760689ED4F355D65383E96CBD45AF41BDF
5C8DC7A3090DF9A8CECEECE1D9E12D4E267255C136F24162D0BF72F0A83B51969DC31033AAB29F701339
52A34A5C17610CDE6764C278F4B3E4816FA49D72AA00D2A73B14663B1F5D8E18E3C0B9F31463B2A2F48E9
4C07A8E0CB2A3020EED00A4EB608B20C2AB08EF4

APDU:  00B000E726 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (38 bytes)
1BECCECEAAE1F5CA20A6F6C1D8DEF5ECC2C459AC3B5C8FE08ADE011B334F5377DF020301000
```

Authentification PORTEUR
=========================
APDU: 002000010431323334 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

MSE SET : Déchiffrement avec clé RSA
=====================================
Tag 81 - octet 0 = '20' = refence clé à utiliser
APDU: 002241B803830120 (8 bytes)
SW12: 9000 (Command successfully executed (OK))

Dechiffrement
=============
APDU:
102A80868000725573652063617365206a65746f6e2043542f434c202e2e2e206a65207375697320756e206a6574
6f6e20706f757220636172746520435053204ec2b0313233343536372076616c61626c652064652032342f30392f32
3032352030393a303020612032342f30392f323032352031373a3030FFFFFFFFFFFFFFFFFFFFFFFF (133 bytes)
SW12: 9000 (Command successfully executed (OK))

APDU:
002A808680FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF (133 bytes)
SW12: 6100 (61XX: Command successfully executed; ‘XX’ bytes of data are available and can be requested using GET RESPONSE)

APDU: 00C0000000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (256 bytes)
48F155E8E3F58C9A3A1BAF8115D6FCD056E4804E195D998DACC247DDE0FF66E36174759C07BDC5B03705
70D99FE570DD553CB465AC6D2CF1626A9E3FAD51F6F73FC15B32A5189C908860296E75997C6D84AE74896
94DCCF88AD35EE2FBAED0491253F5F671BBCD7186D49AEB9B9D2CFE6AD52572B4749F54B3A17B41037E2
E26A0C0DC6B1AA1FBE99ACD5FE73CC671E9DEF3A445C73E98A2851DAC3CA3AF4012EE25C1D284876CC
7E989FFDA5DED4388B1339CC8A009197006EC431B5C6E33125D0193754E8686C5CD5DBDE4D79FE01219C
61455D541C524D81E2034E0E03C622A1558867F6D1A0B031281252E1E8C393F14769944161E54AF5DD0D87
E084EC5

==
Disconnect CT interface and connect to CL
==

[Event-INFO]
[Event-INFO]Power-on
[Event-REC]State:The card has been reset and specific communication protocols have been established.
[Event-REC]Protocol:Use of T=1 protocol
[Event-REC]ATR:3B8180018080

APDU: 00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

APDU: 90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
4F333FC6A557CAFA511D68838BA77025

Key value= 0134A44F0376CDB97A472E541DAD2E46
Key type = AES
Key number (00=MASTER) = 00
randomB_encrypted = 4F333FC6A557CAFA511D68838BA77025
randomB = 047cee18faf0cb824f59eb8a73eadcc3
randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = 7cee18faf0cb824f59eb8a73eadcc304
randomAB_encrypted = 8c39560781ffa3201e69610826b9055bb92f16beff947276efaebe14663820be

APDU: 90AF0000208c39560781ffa3201e69610826b9055bb92f16beff947276efaebe14663820be00 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
0DCA6658AEA86285E03640E9A2862799

[Event-INFO] Authentification réussie.

SessionKey = 00112233047cee18CCDDEEFF73eadcc3

APDU:
00D6850081010048F155E8E3F58C9A3A1BAF8115D6FCD056E4804E195D998DACC247DDE0FF66E36174759C07BDC5B0370570D99FE570DD553CB465AC6D2CF1626A9E3FAD51F6F73FC15B32A5189C908860296E75997C6D84AE7489694DCCF88AD35EE2FBAED0491253F5F671BBCD7186D49AEB9B9D2CFE6AD52572B4749F54B3A17B41037E2E (134 bytes)
SW12: 9000 (Command successfully executed (OK))

APDU:
00D685818126A0C0DC6B1AA1FBE99ACD5FE73CC671E9DEF3A445C73E98A2851DAC3CA3AF4012EE25C1D284876CC7E989FFDA5DED4388B1339CC8A009197006EC431B5C6E33125D0193754E8686C5CD5DBDE4D79FE01219C61455D541C524D81E2034E0E03C622A1558867F6D1A0B031281252E1E8C393F14769944161E54AF5DD0D87E084EC5 (134 bytes)
SW12: 9000 (Command successfully executed (OK))

==========
Jeton généré en CT (2 + 256 octets) et sauvegarde en CL dans fichier DATA
==========

==========
En sans contact, lecture du jeton et déchiffrement avec clé publique
==========

[Event-INFO]
[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR:3B8180018080

APDU: 00A4040C03414E5300 (9 bytes)
SW12: 9000 (Command successfully executed (OK))

APDU: 00B0850002 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (2 bytes)
0100

APDU: 00B0850280 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (128 bytes)
48F155E8E3F58C9A3A1BAF8115D6FCD056E4804E195D998DACC247DDE0FF66E36174759C07BDC5B03705
70D99FE570DD553CB465AC6D2CF1626A9E3FAD51F6F73FC15B32A5189C908860296E75997C6D84AE74896
94DCCF88AD35EE2FBAED0491253F5F671BBCD7186D49AEB9B9D2CFE6AD52572B4749F54B3A17B41037E2
E26

APDU: 00B0858280 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (128 bytes)
A0C0DC6B1AA1FBE99ACD5FE73CC671E9DEF3A445C73E98A2851DAC3CA3AF4012EE25C1D284876CC7E9
89FFDA5DED4388B1339CC8A009197006EC431B5C6E33125D0193754E8686C5CD5DBDE4D79FE01219C614
55D541C524D81E2034E0E03C622A1558867F6D1A0B031281252E1E8C393F14769944161E54AF5DD0D87E08
4EC5

Message initial =
Use case jeton CT/CL ... je suis un jeton pour carte CPS N°1234567 valable de 24/09/2025 09:00 a 24/09/2025 17:00

Jeton déchiffré =
Use case jeton CT/CL ... je suis un jeton pour carte CPS N°1234567 valable de 24/09/2025 09:00 a 24/09/2025 17:00

[Event-ENDS] No Error.
>>>

5.2.12. Exemple d'enrôlement CL

Cet exemple montre un enrôlement qui consiste en sans contact à :

- Lire toutes les données, ce qui permettra d'enregistrer ensuite ces données au sein d'un serveur contrôle d'accès
- Supprimer ensuite l'application DESFire ANS, plus aucune donnée permettant d'identifier le titulaire n'est alors présente au sein du volet DESFire

```
>>> %Run cpsV4-desfire-enrol.py
Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}
[Event-INFO] Script: readcps.py On reader: OMNIKEY CardMan 5x21-CL 0

[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR: 3B8180018080

==> ISO SELECT AID ANS
APDU:  00A4000002A00000 (8 bytes)
SW12: 9000 (Command successfully executed (OK))

==> ISO READ SN file
APDU:  00B0830000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (12 bytes)
5A0A80250000001035000063F

==> ISO READ IDCARD
APDU:  00B0820000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (7 bytes)
31006409310000

==> ISO READ IDNAT
APDU:  00B0810000 (5 bytes)
SW12: 9000 (Command successfully executed (OK))
Data from card: (33 bytes)
383939373030363232323837000000000000000000000000000000000000000000000

==> GET VERSION DESFIRE
APDU:  9060000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04810043011A05

==> GET VERSION DESFIRE step 2
APDU:  90AF000000 (5 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (7 bytes)
04814603001A05
```

==> GET VERSION DESFIRE step 3 (UID)
APDU: 90AF000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (14 bytes)
04252FE2041D9021025000001524

==> ISO SELECT AID ANS
APDU: 00A4040007D276000085010000 (13 bytes)
SW12: 9000 (Command successfully executed (OK))

==> AUTHENTICATE
APDU: 901A0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (8 bytes)
8CE83B16BA9ABDCE

Key value= 00000000000000000000000000000000
Key type = 2K3DES
Key number (00=MASTER) = 00
randomB_encrypted = 8CE83B16BA9ABDCE
randomB = 13ea39d4b594190c
randomA = E2CD97081A35E3D8
randomB_rotate = ea39d4b594190c13
randomAB_encrypted = f16460f56e66ccabc4e43837a30083e2

==> AUTHENTICATE (AF)
APDU: 90AF000010f16460f56e66ccabc4e43837a30083e200 (22 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
6FA5E564629B7BF7

[Event-INFO] Authentification réussie.

Session Key = E2CD970813ea39d41A35E3D8b594190c
Subkey 1 = 2EF1A7A078CBEC60F38B70BF653C2780
Subkey 2 = 5DE34F40F197D8C1E716E17ECA784F00
Iv = 00000000000000000000000000000000

==> DELETE AID ANS
APDU: 90DA000003534E4100 (9 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
7CD3EFBCB27E3E97

==> ISO SELECT AID ANS
APDU: 00A4000002A00000 (8 bytes)
SW12: 6A82 (File not found) ➔ **Application ANS supprimée**

[Event-ENDS] No Error.
>>>

5.2.13. Personnalisation application DESFire ANS

Exemple de création d'application ANS :

- Création d'une application
- Création de fichier DATA : SN, IDCARD …
- Changement des conditions d'accès à ces fichiers

Readers: {"response": 0, "data": ["OMNIKEY CardMan 5x21 0", "OMNIKEY CardMan 5x21-CL 0"]}

[Event-INFO] Power-on
[Event-REC] State:The card has been reset and specific communication protocols have been established.
[Event-REC] Protocol:Use of T=1 protocol
[Event-REC] ATR: 3B8180018080

==> GET FREE MEMORY
APDU: 906E000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (3 bytes)
002000

==> AUTHENTICATE
APDU: 90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
A866A30C4E07092E05DE4FDA10089943

Key value= 112233445566778899AABBCCDDEEFF00
Key type = AES
Key number (00=MASTER) = 00
randomB_encrypted = A866A30C4E07092E05DE4FDA10089943
randomB = b3a9ae89c43a4d9afbed57c4aceb73ce
randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = a9ae89c43a4d9afbed57c4aceb73ceb3
randomAB_encrypted = 91932c1d85950dda9e958d786b4dc25b61974e13149bf7797a31b33cb099f892

==> AUTHENTICATE (AF)
APDU: 90AF00002091932c1d85950dda9e958d786b4dc25b61974e13149bf7797a31b33cb099f89200 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
D5D901ED55F8604CE265115608553AB2

[Event-INFO] Authentification réussie / MASTER KEY RACINE

SessionKey = 00112233b3a9ae89CCDDEEFFaceb73ce
Subkey 1 = 032AD148CF83D8A5CF75FEFA66EF4287
Subkey 2 = 0655A2919F07B14B9EEBFDF4CDDE850E
Iv = 00000000000000000000000000000000

==> Create AID '414E53'
APDU: 90CA00000A414E53E3A2A000414E5300 (16 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
9D3E447204986F7B

==> SELECT AID ANS
APDU: 905A000003414E5300 (9 bytes)
SW12: 9100 (OK)

==> AUTHENTICATE
APDU: 90AA0000010000 (7 bytes)
SW12: 91AF (Additional data frame is expected to be sent)
Data from card: (16 bytes)
195C36615B2A820AD0661F1DDD5A2E87

Key value= 00000000000000000000000000000000
Key type = AES
Key number (00=MASTER) = 00
randomB_encrypted = 195C36615B2A820AD0661F1DDD5A2E87
randomB = d735edd2e697bd76e2814e2c066ab969

randomA = 00112233445566778899AABBCCDDEEFF
randomB_rotate = 35edd2e697bd76e2814e2c066ab969d7
randomAB_encrypted = d66469a5fb1741df47c56819afa70f0fe8af34545df00df76c706b0e13990f12

==> AUTHENTICATE (AF)
APDU: 90AF000020d66469a5fb1741df47c56819afa70f0fe8af34545df00df76c706b0e13990f1200 (38 bytes)
SW12: 9100 (OK)
Data from card: (16 bytes)
F492EEA1370E36F42668EECC45B312E9

[Event-INFO] Authentification réussie MASTER KEY APPLICATION (Exemple avec valeur à 00..00)

SessionKey = 00112233d735edd2CCDDEEFF066ab969
Subkey 1 = 9633A6F3719DFEA8C48D92CABA9E05BB
Subkey 2 = 2C674DE6E33BFD51891B2595753C0BF1
Iv = 00000000000000000000000000000000

==> Create File 03 / SN
APDU: 90CD0000090303A000FEEE0C000000 (15 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
A9262312C3026CDF

[Event-INFO] CMAC ok

==> Update File
APDU: 903D000013030000000C00005A0A80250000010309533308F00 (25 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
9D19C8CF724ED796

[Event-INFO] CMAC ok

==> Change File Setting 03 / READ ALWAYS – WRITE NEVER
APDU: 905F0000040300FFEF00 (10 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
473305D84846ED96

[Event-INFO] CMAC ok

==> Create File 04 / SDA
APDU: 90CD0000090404A000FEEE10010000 (15 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
47A6D51A9877ECB0

[Event-INFO] CMAC ok

==> Change File Setting 04 / READ ALWAYS – WRITE NEVER
APDU: 905F0000040400FFEF00 (10 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
6BF91871CAF1F744

[Event-INFO] CMAC ok

==> Create File 05 / DATA / READ DATA – WRITE NEVER après AUTH Master key application ANS
APDU: 90CD0000090505A000FFE02C010000 (15 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
2D89DE3D99E5183D

[Event-INFO] CMAC ok

==> Create File 02 / IDCARD
APDU: 90CD0000090202A000FEEE07000000 (15 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
3657DC0344A0986B

[Event-INFO] CMAC ok

==> Update File
APDU: 903D00000E020000000700008105310060375200 (20 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
B703ED0D32E04576

[Event-INFO] CMAC ok

==> Change File Setting 02 / READ ALWAYS – WRITE NEVER – CHANGE ACCESS après AUTH MASTER KEY application ANS
APDU: 905F0000040200F0EF00 (10 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
96D87B23DD47E6F2

[Event-INFO] CMAC ok

==> Create File 01 / IDNAT
APDU: 90CD0000090101A000FEEE21000000 (15 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
F9EC3F42786CCC90

[Event-INFO] CMAC ok

==> Update File
APDU: 903D00001D010000001600008114333042303235363932312F43504554303030303100 (35 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
11857DF9E8C67B8B

[Event-INFO] CMAC ok

==> Change File Setting 01 / READ ALWAYS – WRITE NEVER – CHANGE ACCESS après AUTH MASTER KEY application ANS
APDU: 905F0000040100F0EF00 (10 bytes)
SW12: 9100 (OK)
Data from card: (8 bytes)
26F2B0149F82DBB2

[Event-INFO] CMAC ok

==> GET FREE MEMORY
APDU: 906E000000 (5 bytes)
SW12: 9100 (OK)
Data from card: (11 bytes)
401C00B22671245C376199

[Event-ENDS] No Error.
>>>