Support de la

Cryptolib CPS v5

Impacts de la **migration** **CPS2Ter** vers **CPS3**

asipsanté
AGENCE DES SYSTÈMES D'INFORMATION PARTAGÉS DE SANTÉ

V1.2.10 du 09/04/2015

Support de la Cryptolib CPS v5 - Impacts de la migration CPS2Ter vers CPS3

« ASIP Santé / PUSC / PSCE »

Version 1.2.10 du 09/04/2015

Historique du document			
Version	Date	Auteur	Commentaires
1.2.6	10/09/2014	ASIP	
1.2.7	20/10/2014	ASIP	Conseils CPTAB Problème 8.3.3 depuis le 24/06/2014
1.2.8	12/11/2014	ASIP	KEY_ID Sans-contact Remarque sur la gestion du cache par les différentes versions de la Cryptolib CPS
1.2.9	03/03/2015	ASIP	Mise à jour Glossaire Précisions alternative pour l'accès au DAM
1.2.10	09/04/2015	ASIP	Mise à jour Schéma saisie de code porteur

1 Références

Documents de référence				
N°	**Version**	**Date**	**Auteur**	**Document**
DETECT_CRY	1.0.0	05/03/2014	ASIP Santé	Note technique Implémentation de la détection de la Cryptolib ASIP-PUSC-PSCE_NP_Guide-implementation-Detection-Cryptolib-CPS_20140305_v1.0.0.pdf
MANUEL_PROG	1.5.0	16/10/2013	ASIP Santé	Cryptolib CPS v5 Manuel de Programmation ASIP-PUSC-PSCE_MP_Cryptolib-CPS-v5-Manuel-de-programmation_20131016_v1.5.0.pdf
ARRACH_CPS	1.0.3	17/10/2013	ASIP Santé	Guide d'implémentation Détection de l'arrachage de la carte CPS « PSCE » ASIP-PUSC-PSCE_NP_Guide-implementation-detection-arrachage-CPS_20131017_v1.0.3.pdf
API_CPS			ASIP Santé	APIs Des Services Carte du Professionnel de Santé win32504_0.zip\CW320504\doc\cpmpracp.doc
DONNEES_METIER	1.0.2		ASIP Santé	Les données métier de la CPS3 Volets CPS2ter et IAS ASIP_CPS3_Données-métier_v1.0.2.pdf
PKCS11_EXT[1]	1.0.1		ASIP Santé	CryptoLib CPS3 Spécifications externes du module PKCS#11 SpécificationsExternes_PKCS11_CryptoLib_CPS3_v1.0.1.pdf
MANUEL_INST_UTIL	5.0.9		ASIP Santé	Manuel d'installation et d'utilisation de la Cryptolib CPS ASIP-PUSC-PSCE_Manuel-Installation-utilisation-Cryptolib-CPS_20140912_v5.0.9.pdf

Tableau 1 : Documents de référence

[1] Document distribué sur le site http://integrateurs-cps.asipsante.fr et nécessitant un compte « intégrateur »

2 Résumé

Ce document reprend les éléments présentés dans **[MANUEL_INST_UTIL]** en se focalisant sur les impacts de migration de la Cryptolib CPS v4 vers la Cryptolib CPS v5.

La Cryptolib CPS v5 assure une compatibilité ascendante avec la Cryptolib CPS v4 :

1. en copiant sur le disque dur, lors de l'installation, les librairies de la Cryptolib CPS v4
2. en enregistrant des composants iso-fonctionnels avec ceux de la Cryptolib CPS v4 (CSP par exemple)

Dans un premier temps, l'installation d'une Cryptolib CPS v5 est donc **transparente** pour une application s'appuyant sur la Cryptolib CPS v4.

L'installeur de la Cryptolib CPS v5 déploie par ailleurs des composants logiciels en version 5 qui apportent **4 évolutions importantes** :

1. Une meilleure compartimentation du code porteur
2. Un meilleur support des architectures Microsoft Windows
3. Un meilleur support du standard PKCS#11
4. Le support du standard IAS-ECC

Ce guide identifie les développements qu'il est nécessaire d'effectuer afin de passer d'une utilisation du volet CPS2Ter à une utilisation du volet CPS3 et de **tirer pleinement parti de ces améliorations**.

Cette **phase de migration** applicative peut donc se faire dans **un second temps**.

3 Sommaire

4 Glossaire

Sigle	Signification
AW DMP	Access Web DMP
API	Application Programming Interface
CAPI	Cryptographic Application Programming Interface
CCM	CAPI Certificate Manager
CPS ou CPx	Carte de Professionnel de Santé ou Carte de la famille CPS (CPS, CPE, CPA, …)
CSP	Cryptographic Service Provider
DLL	Dynamic Link Library (bibliothèque à liens dynamiques)
DMP	Dossier Médical Personnel
ES	Etablissement de Santé
FSE	Feuille de Soins Electronique
FSV	Facturation SESAME Vitale
GALSS	Gestionnaire des Accès aux Lecteurs Santé Social
GIE	Groupement d'Intérêt Economique
IAS	Identification-Authentification-Signature : norme française pour les cartes à puces
ODI	Outil de Diagnostic et d'Installation du poste PS (http://www.outil-diagnostic.asipsante.fr)
PC/SC	Personal Computer/Smart Card
PKCS	Public-key Cryptography Standards
PKI	Public Key Infrastructure
PS	Professionnel de Santé
PSS	Protocol Santé Social
RGS	Référentiel Général de Sécurité

Tableau 2 : Glossaire

5 Liste des entreprises citées

Le présent document cite les produits des entreprises ou organismes suivants:

Nom	Site Web	Lien avec la Cryptolib CPS
Apple	www.apple.com	Mac OS X
Debian	www.debian.org	Linux, .deb
Fedora	fedoraproject.org	Linux, .rpm
Google	www.google.com	Google Chrome
GIE SESAM-Vitale	www.sesam-vitale.fr	PSS, GALSS, API de lecture Vitale
GIXEL	www.gixel.fr	Standard IAS-ECC
Microsoft	www.microsoft.com	Windows, CSP, Internet Explorer, C#, .NET, TSE
Mozilla	www.mozilla.org	Mozilla Firefox
OASIS	www.oasis-open.org	Responsable des évolutions du Standard PKCS#11 depuis sa version 2.3
PC/SC Lite	ludovic.rousseau.free.fr	PC/SC sous Linux
PC/SC Workgroup	www.pcscworkgroup.com	Responsable du standard PC/SC visant l'intégration de la carte à puce et des lecteurs de cartes dans les systèmes informatiques
Redhat	www.redhat.com	Linux, .rpm
RSA Security Inc.	www.rsa.com	PKCS, RSA

Tableau 3 : Entreprises citées

6 Avertissements

Sur le nécessaire strict respect des procédures décrites dans le document L'attention de l'utilisateur est attirée sur l'importance de respecter strictement les procédures décrites dans le présent document. Toutes les procédures qui y sont décrites ont été préalablement testées par l'ASIP Santé. Elles doivent permettre à l'utilisateur d'évaluer les efforts de migration vers l'utilisation la Cryptolib CPS v5 sur son poste de travail ou tout autre dispositif informatique. En cas de non-respect de ces procédures et des conditions normales d'utilisation de la Cryptolib CPS v5, sa mise en œuvre est susceptible d'engendrer des dysfonctionnements dans l'environnement de travail de l'utilisateur. En cas de dysfonctionnement, quel qu'il soit, l'ASIP Santé prêtera dans la mesure du possible assistance à l'utilisateur, qui ne pourra rechercher sa responsabilité en cas de non-respect des procédures décrites dans le présent manuel.
Sur les liens externes Le présent document contient des liens vers des sites Internet. Ces liens ne visent qu'à informer l'utilisateur. Ces sites Web ne sont pas gérés par l'ASIP Santé et l'ASIP Santé n'exerce sur eux aucun contrôle : leur mention ne saurait engager l'ASIP Santé quant à leur contenu. L'utilisation des sites tiers mentionnés relève de la seule responsabilité du lecteur ou de l'utilisateur des produits documentés.
Sur les copies d'écran et les lignes de commande Les lignes de commandes données ci-après le sont à titre indicatif. Elles documentent des cas « passants » qui peuvent différer d'un système à l'autre. Les copies d'écran présentées dans ce document sont données à titre illustratif. Les pages ou écrans réellement affichés peuvent être différents, notamment en raison de montées de version ou de configurations d'environnements différentes.
Citations L'ASIP Santé est contrainte de citer le nom de certaines entreprises recensées au tableau n°5 afin d'apporter toute l'aide nécessaire aux utilisateurs de la Cryptolib CPS v5 dans son installation et son utilisation. Les entreprises citées peuvent prendre contact avec l'ASIP Santé à l'adresse email editeurs@asipsante.fr pour toute demande en lien avec la citation les concernant. Les entreprises non citées dans ce manuel et ayant une activité en lien avec la Cryptolib CPS v5 peuvent également se faire connaître auprès de l'ASIP Santé en la contactant à la même adresse.
Contact Toute question en rapport avec le contenu du présent document doit être adressée à l'adresse suivante: editeurs@asipsante.fr

Tableau 4 : Avertissements

7 Principales fonctionnalités de la Cryptolib CPS v5

[MANUEL_INST_UTIL] décrit exhaustivement l'architecture, l'installation et l'utilisation de la Cryptolib CPS v5 sur les trois systèmes Microsoft, Linux et Mac OS X.

Le présent document suppose que les principales notions introduites par la Cryptolib CPS v5 sont acquises, notamment après lecture de **[MANUEL_INST_UTIL]**.

Ce chapitre en reprend les éléments permettant d'identifier les impacts de migration de la Cryptolib CPS v4 vers la Cryptolib CPS v5.

7.1 Compatibilité ascendante complète avec la Cryptolib CPS v4

Le premier élément à bien comprendre concernant la Cryptolib CPS v5 est qu'elle assure une compatibilité ascendante avec la Cryptolib CPS v4 :

3. en copiant sur le disque, lors de l'installation, les librairies de la Cryptolib CPS v4 :
    1. API CPS de la Cryptolib CPS v4
    2. PKCS#11 de la Cryptolib CPS v4
    3. CSP de la Cryptolib CPS v4
4. sous Microsoft Windows : en enregistrant le CSP de la Cryptolib CPS v5 au niveau système
    a. Le CSP de la Cryptolib CPS v5 est iso-fonctionnel avec celui de la Cryptolib CPS v4

De fait, l' « installeur Cryptolib CPS v5 » est composé des éléments suivants :

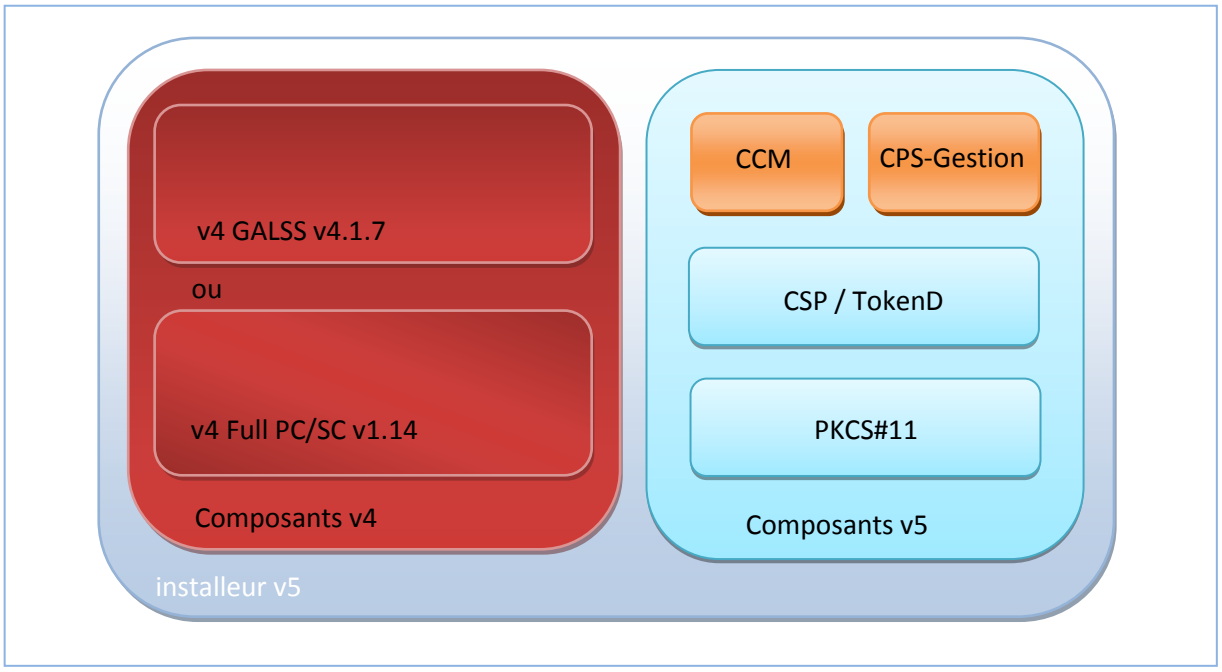


Figure 1 : description de l'installeur Cryptolib CPS v5

Dans les différents cas listés ci-dessous, l'installation d'une Cryptolib CPS v5 est transparente pour une application s'appuyant sur la Cryptolib CPS v4 :

Architecture de l'application	**Description de l'impact du passage de la Cryptolib CPS v4 à la v5**
l'application utilise le PKCS#11 de la Cryptolib CPS v4	aucun impact après installation du package Cryptolib CPS v5 puisque le PKCS#11 de la Cryptolib CPS v4 est copié
l'application utilise l'API CPS de la Cryptolib CPS v4	aucun impact après installation du package Cryptolib CPS v5 puisque l'API CPS de la Cryptolib CPS v4 est copiée
l'application adresse en dur le CSP de la Cryptolib CPS v4	aucun impact après installation du package Cryptolib CPS v5 puisque le CSP de la Cryptolib CPS v4 est copié
Sous Microsoft Windows, l'application utilise l'interface CAPI du système	aucun impact après installation du package Cryptolib CPS v5 puisque :
	- c'est le CSP de la Cryptolib CPS v5 qui est sollicité
	- et qu'il est iso-fonctionnel avec le CSP de la Cryptolib CPS v4

Tableau 5 : Impact du passage de la Cryptolib CPS v4 à la v5

7.2 Cryptolib CPS v5 : un meilleur suivi des standards

La Cryptolib CPS v5 introduit 4 évolutions importantes :

5. Une meilleure compartimentation du code porteur
    a. La Cryptolib CPS v5 suit les dernières recommandations de sécurité en matière de saisie du code porteur
6. Un meilleur support des architectures Microsoft Windows
    a. Support du 64b
    b. Améliorations du CSP
        i. Notamment en termes de performances
7. Un meilleur support du standard PKCS#11
    a. Support plus standard des opérations cryptographiques
    b. Calcul de signature avec un dernier tour de hash calculé par la carte
    c. Utilisation plus standard des attributs des objets PKCS#11
        i. Qui permet notamment de « déprécier » l'API CPS
8. Le support du standard IAS-ECC
    a. Avec comme perspective la signature qualifiée, l'interopérabilité avec les systèmes d'information des autres pays européens…

Ces évolutions peuvent avoir des impacts :

1. sur les applications utilisant l'API CPS
    a. à « T0 », comme expliqué plus haut, l'installeur de la Cryptolib CPS v5 installe l'API CPS
        i. il n'y a aucun impact
    b. il est conseillé de remplacer les appels à cette API par des appels équivalents à l'API PKCS#11
        i. voir ci-après pour le détail de la migration appel par appel
2. sur les applications utilisant déjà le PKCS#11 de la Cryptolib CPS v4
    a. avec une utilisation peu standard des appels PKCS#11
3. sur les applications ne gérant pas parfaitement la saisie du code porteur
    a. en particulier sur les applications existantes n'ayant pas cherché à rationaliser leurs filières d'accès aux cartes, que ces cartes soient CPx ou Vitale

Ces points sont repris dans le détail ci-après.

8 Préconisations d'architectures logicielles pour les LPS

8.1 Gestion du code porteur

8.1.1 Architecture logicielle problématique

La nouvelle gestion du code porteur peut avoir des effets de bord sur les applications mettant en œuvre une architecture hétérogène pour ses accès carte.

Un exemple d'application impactée :

1. L'application utilise l'interface CAPI pour certaines opérations
    a. OPE_TYPE_1
    b. avec la Cryptolib CPS v5, c'est le CSP v5 qui va être sollicité
2. L'application utilise pour d'autres opérations soit le PKCS#11 de la Cryptolib CPS v4 soit l'API CPS
    a. OPE_TYPE_2

Dans ce cas, le CSP v5 ne permettant pas le partage de l'état de présentation du code porteur, le passage de la Cryptolib CPS v4 vers la Cryptolib CPS v5 n'est pas transparent :

1. L'utilisateur lance une OPE_TYPE_1 : le code porteur va lui être demandé
2. L'utilisateur lance une OPE_TYPE_2 : le code porteur va de nouveau lui être demandé
    a. ce qui n'était pas le cas avec la Cryptolib CPS v4 puisque le CSP v4 permettait le partage de l'état de présentation du code porteur

Cependant, une application ayant ces caractéristiques présente un défaut d'architecture.

8.1.2 Algorithme de présentation du code porteur au niveau PKCS#11

L'ASIP Santé documente et partage pour information et avis un algorithme de gestion de la présentation du code porteur au niveau PKCS#11 :

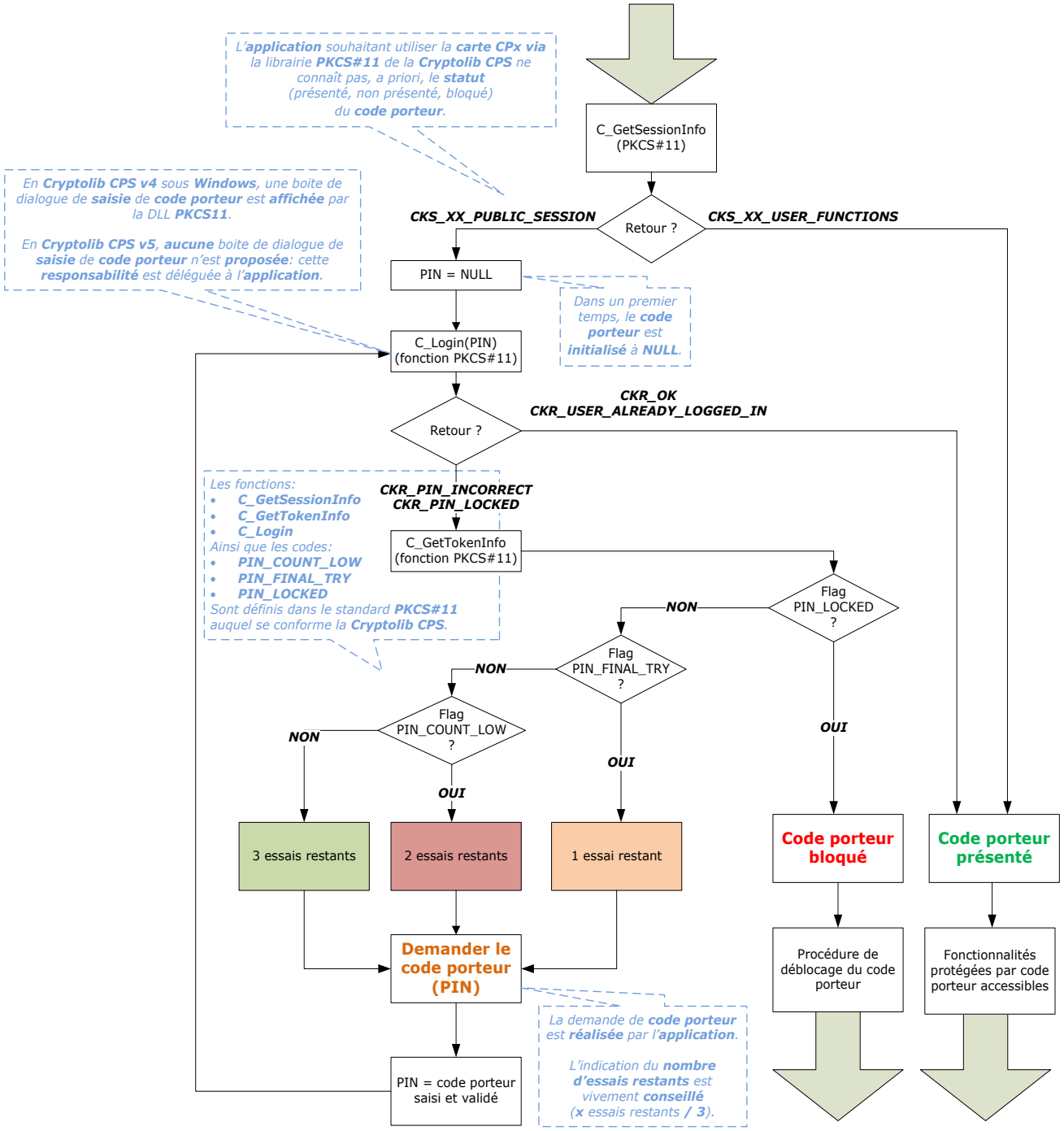


Figure 2 : Algorithme de présentation du code porteur au niveau PKCS#11

L'utilisation de cet algorithme, toutes versions de Cryptolib CPS confondues et tous niveaux d'intégration confondus (PKCS#11 et système i.e. CSP et TokenD), tend à minimiser les demandes de code porteur effectuées auprès de l'utilisateur.

8.1.3 Factorisation des demandes de codes porteur

A partir de la Cryptolib CPS v5, les demandes de code porteur se font processus par processus.

Au sein d'un même processus, modulo la mise en œuvre d'une véritable stratégie de demande du code porteur basée sur les informations de ce document :

- Rationalisation des filières d'accès carte (cf. plus haut)
- Détection du statut de présentation du code porteur (cf. plus haut)

il est possible de réduire au minimum les demandes de code porteur.

Il est difficilement compréhensible qu'un même logiciel demande par lui-même et pour son propre fonctionnement plusieurs fois le code porteur CPS. Il est encore moins compréhensible que les demandes de saisie de code porteur se fassent via des champs de saisie différents (manque de cohérence, défaut d'ergonomie de l'application).

8.2 Niveau et homogénéité de l'intégration avec la carte

Les chapitres précédents font apparaître un point important : les applications doivent se préoccuper du niveau et du degré d'homogénéité de leur intégration avec la carte.

4 règles se dégagent :

#	Recommandation	Détails
1	L'intégration doit se faire préférentiellement au niveau du système	cf. **[MANUEL_INST_UTIL]** : • CSP pour Microsoft • PKCS#11 pour Linux • TokenD pour Mac OS X
2	A défaut: l'intégration doit se faire au niveau PKCS#11	logiciel cross-plateforme par exemple

3	Dans le cas particulier d'une « intégration experte » :
	• Intégration au niveau choisi en conséquence
	  ○ de besoins clairement identifiés et exprimés
	  ○ et des compétences expertes identifiées et disponibles
		cf. **[MANUEL_INST_UTIL]**, par exemple : intégration au niveau PC/SC par des entreprises ayant une expertise avérée en cartes / IAS / PC/SC / Cryptographie

4	L'intégration doit se faire à un seul et même niveau	Pas de mélange des niveaux

Tableau 6 : Règles d'intégration avec la carte

Les applications existantes ne respectant pas ces 4 règles présentent un défaut d'architecture.

Parallèlement, les middlewares cartes ne permettant pas un tel niveau d'intégration présentent un grave défaut de conception qui induit des coûts de conception et de maintenance très importants pour les éditeurs:

- nécessité de monter en compétence les équipes techniques sur des technologies de niche peu performantes, mal maintenues, mal maîtrisées et mal déployées
- manque de compétences disponibles sur le marché
- « insularité » de la réflexion, des corrections, des perspectives d'évolution qui freine l'innovation et l'efficience à long terme
- interopérabilité difficile (bridges)
- frein à l'apparition de nouveaux acteurs hors secteur pouvant potentiellement apporter des nouvelles idées, des nouveaux concepts ou des nouvelles méthod(ologi)es

Cette préconisation que l'on peut résumer sous le terme « respect des standards » est reprise par l'ASIP Santé dans ses préconisations d'architecture logicielle. L'ASIP Santé applique, autant que faire se peut, sa propre règle à ses propres projets.

8.3 Analyse de différents logiciels dans leur partie « accès carte »

8.3.1 Exemple 1: étude du cas « application multi-processus »

Une application multi-processus utilisant la Cryptolib CPS pour accéder à la carte CPx et organisée tel qu'illustré ci-après est impactée par le passage de la Cryptolib CPS v4 vers les composants Cryptolib CPS v5 (« double saisie » de code porteur probable) :

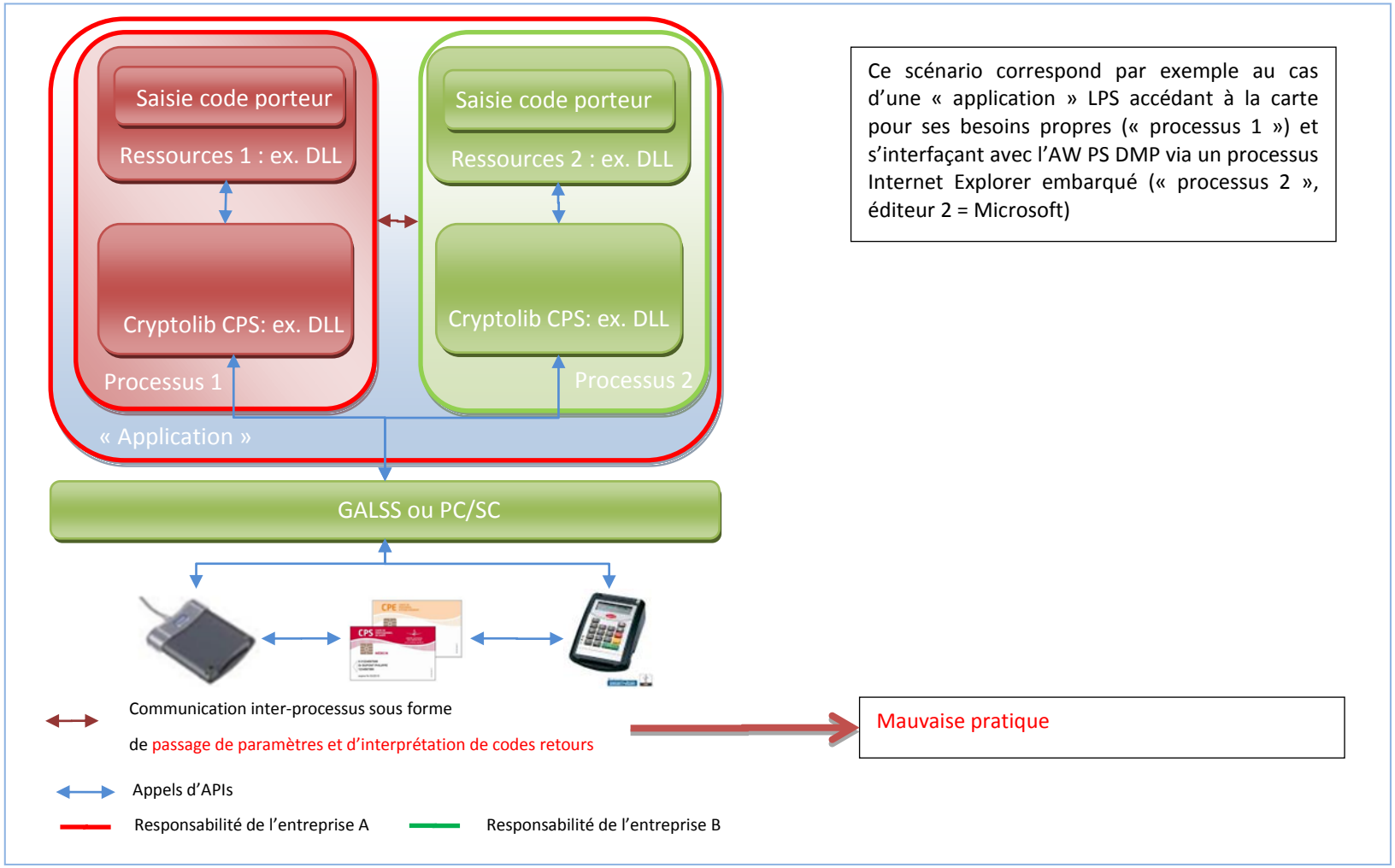


Figure 3 : Application multi-processus utilisant la Cryptolib CPS pour accéder à la carte CPx

3 manières de régler le problème :

8.3.1.1 Solution 1- Communication inter-processus « ad-hoc »

« ad_hoc » i.e. spécifique au contexte particulier de « application », pour les connexions cartes

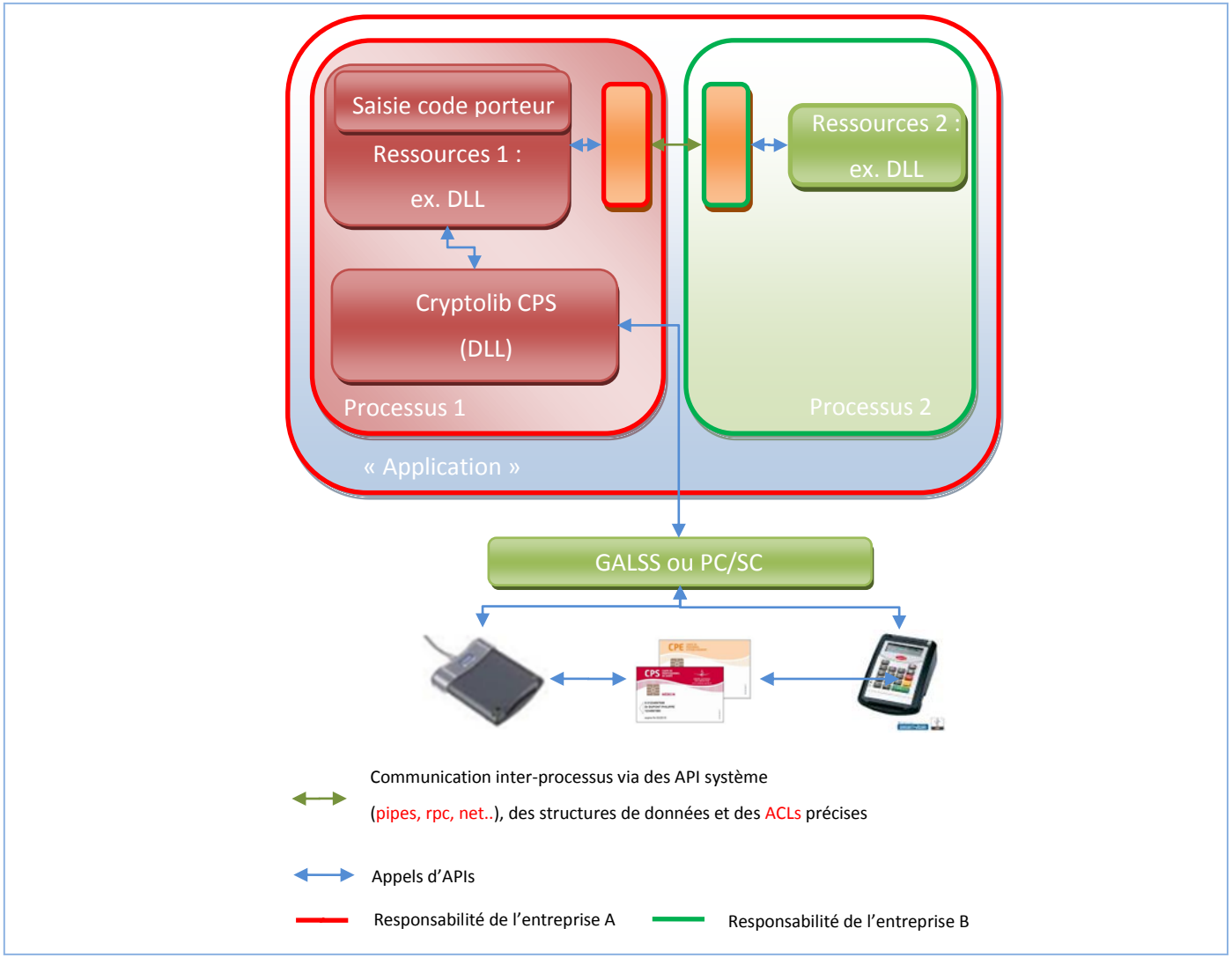
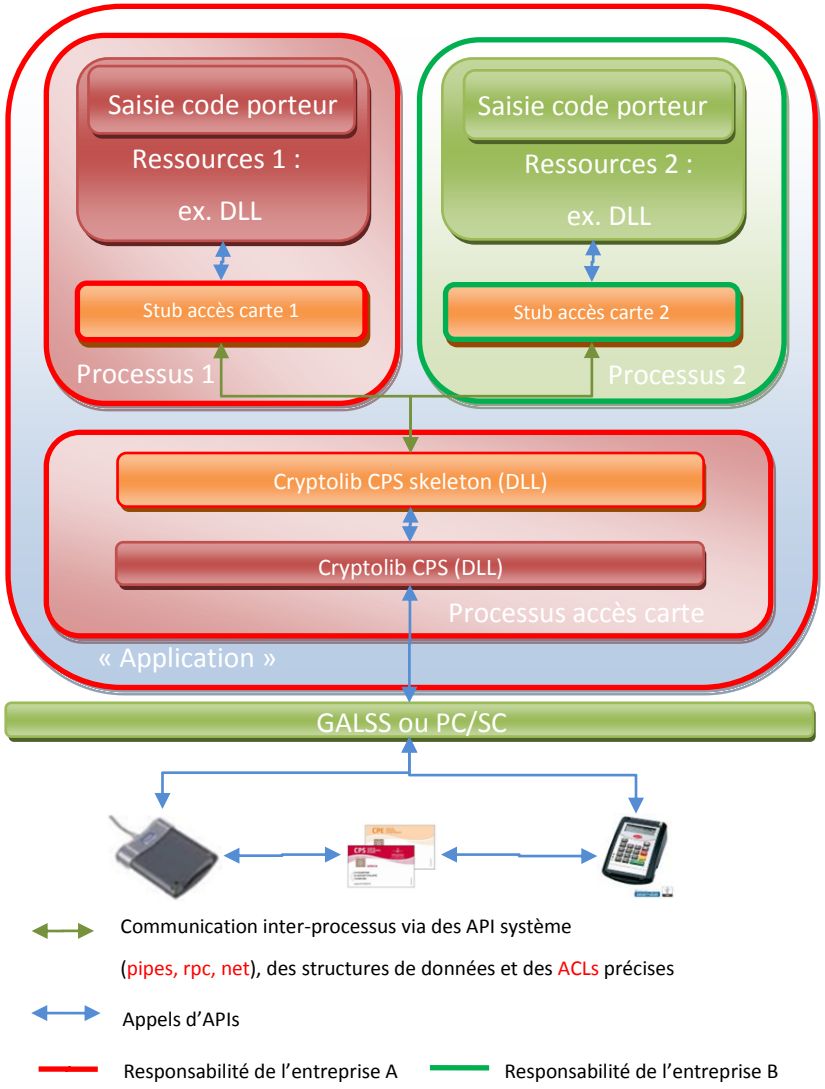


Figure 4 : Application multi-processus : mise en place d'une communication inter-processus ad-hoc

8.3.1.2 Solution 2- Communication inter-processus via un bus d'accès cartes



Stub accès carte 1 et 2 peuvent être des DLLs PKCS#11 (peu d'impact dans le code des processus 1 et 2 si ceux-ci utilisent déjà PKCS#11) agissant comme client RPC du « processus accès carte »

« Cryptolib CPS skeleton » peut prendre la forme d'un {wrapper PKCS#11 ; serveur RPC} et être lui-même une DLL PKCS#11…

Figure 5 : Application multi-processus : mise en place d'une communication inter-processus via un bus d'accès carte

8.3.1.3 Solution 3- Intégration logicielle totale

L'intégration souhaitée par A pour son produit des produits de B est formalisée sous la forme d'une API exposée par B et utilisée par A. B expose l'intégralité de ses fonctionnalités via des APIs logicielles documentées.

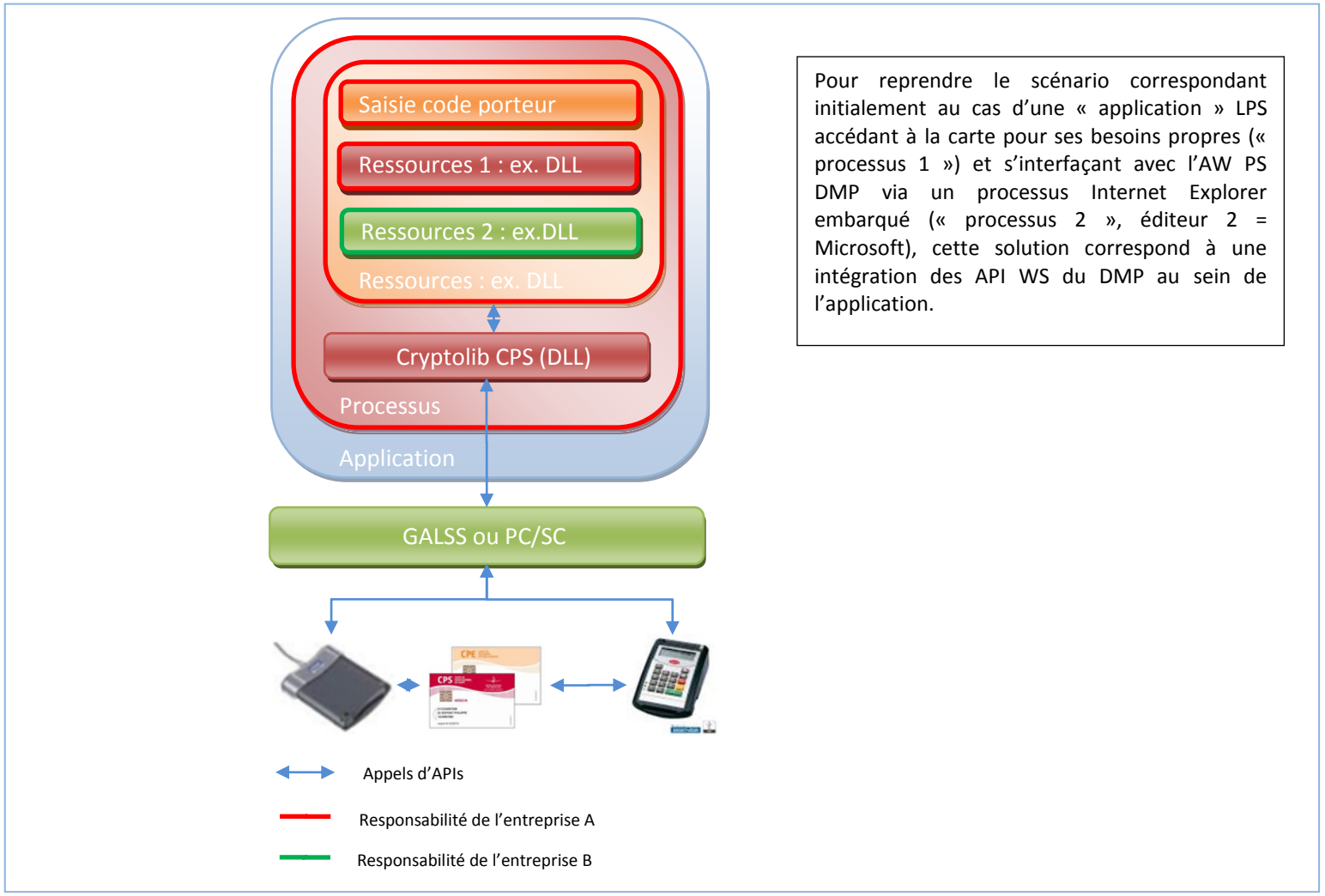


Figure 6 : Application « intégrée » utilisant la Cryptolib CPS pour accéder à la carte CPx

8.3.2 Exemple 2: étude d'un exemple en Java

8.3.2.1 Code d'accès à la CPx

Le code suivant utilise Java et PKCS#11 pour récupérer les 2 certificats carte et les identifier afin de faire ultérieurement soit une opération de signature soit une opération d'authentification.

#	Code	Commentaire
0	[...]	
1	StringBuilder cardConfig = new StringBuilder();	
2	cardConfig.append("name = CPS\n");	Nous sommes en Java donc potentiellement cross-plateforme : l'utilisation de PKCS#11 est une bonne idée, d'autant plus que c'est un standard bien suivi et bien supporté par le monde Java
3	cardConfig.append("library = " + module);	
4	InputStream is = new ByteArrayInputStream(cardConfig.toString().getBytes());	
5	log.info("Loading Sun PKCS11 security provider...");	
6	Provider securityProvider = new sun.security.pkcs11.SunPKCS11(is);	
7	Security.*addProvider*(securityProvider);	Java permet ici d'adapter directement les objets PKCS#11 dans sa propre architecture JCE/JCA, ce qui est une bonne chose
8	[...]	
9	// Loading client KeyStore from card	
10	log.debug("Loading CPS KeyStore...");	
11	keystore = KeyStore.*getInstance*("PKCS11");	
12	keystore.load(null, cardPassword == null ? null : cardPassword);	
13	log.debug("Looking for alias");	
14	// Looking for the private key into the keystore, looking for an	
15	// alias containing 'signature'	
16	if (keystore != null) {	
17	Enumeration<String> e = keystore.aliases();	
18	while (e.hasMoreElements()) {	
19	String alias = e.nextElement();	
20	log.debug("Alias : " + alias);	
21	if (keystore.isKeyEntry(alias) && alias.contains("signature")) {	L'usage du certificat est ici récupéré à partir de l'alias PKCS#11 : - ce qui rend ce code plus sensible au changement d'alias - et donc, dans ce cas, au changement Volet CPS2Ter / Volet IAS - ce qui n'est pas conforme aux bonnes pratiques cryptographiques -> utiliser plutôt l' « usage » du certificat : - getKeyUsage()[OFFSET_NON_REPUDIATION] - plus contraignant mais plus générique
22	// Get a private key handle:	
23	log.debug("Get private key using alias : " + alias);	
24	signaturePrivateKey = (PrivateKey) keystore.getKey(alias, cardPassword);	
25	signatureCertificate = (X509Certificate) keystore.getCertificate(alias);	
26	signatureCertificationChain = keystore.getCertificateChain(alias);	
27	} else if (initAuthentificationCertificate	

#	Code	Commentaire
28	&& keystore.isKeyEntry(alias)	
29	&& alias.contains("authentification")) {	
30	authentificationCertificateChain = (X509Certificate[]) keystore	
31	.getCertificateChain(alias);	
32	// Get the private key	
33	log.debug("Get private key using alias : " + alias);	
34	authentificationPrivateKey = (PrivateKey) keystore.getKey(alias,	
35	cardPassword);	
36	}	
37	}	
38	}	
39	[...]	

Tableau 7 : Exemple 1 - Java

8.3.2.2 Conclusion

- ce code n'est pas impacté à « T0 » par la migration Cryptolib CPS v4 vers Cryptolib CPS v5
  - car le package Cryptolib CPS v5 déploie les composants Cryptolib CPS v4
- ce code est impacté au moment de passer à l'exploitation du volet IAS de la CPS3 car
  - de fait, il ne fonctionne plus du tout
  - les objets PKCS#11 correspondant au volet IAS ont changé (cf. ci-après)
  - Ce qui n'a pas d'impact si les préconisations PKCS#11 sont bien suivies
    - ce qui n'est pas le cas ici

8.3.3 Exemple 3: étude d'un exemple en C# (1)

8.3.3.1 Code d'accès aux certificats CPx

Le code suivant utilise C# pour récupérer les 2 certificats carte et les identifier afin de faire ultérieurement soit une opération de signature soit une opération d'authentification.

#	Code	Commentaire
0	`[...]`	
1	`Namespace Pkcs11 {`	On parle ici de PKCS#11 mais on intègre en fait au niveau CSP au travers du RSACryptoServiceProvider .NET! - Intégration au niveau CSP / .NET est une bonne idée sous Windows - Le nom PKCS#11 induit de la confusion et désoriente au moment de maintenir le code, ou d'évaluer l'impact d'une migration par exemple (migration de CSP dans ce cas et non de PKCS#11)
2	`  class Pkcs11 {`	
3	`[...]`	
4	`    public Pkcs11() {`	
5	`      x509Store = new X509Store(StoreName.My, StoreLocation.CurrentUser);`	
6	`      x509Store.Open(OpenFlags.ReadWrite);`	
7	`      foreach (X509Certificate2 certificate in x509Store.Certificates) {`	
8	`        if (certificate.Issuer.` `  Equals("CN=TEST CLASSE-1, OU=TEST PROFESSIONNEL, O=TEST, C=FR")) {`	Chaînes en dur : - mauvaise pratique (« magic string ») - Sensible au changement d'IGC - Mettre en configuration
9	`          if(authentificationCertificate==null) {`	
10	`            authentificationCertificate = certificate;`	
11	`          } else if (signatureCertificate == null) {`	
12	`            signatureCertificate = certificate;`	

#	Code	Commentaire
13	`if (Int32.Parse(signatureCertificate.SerialNumber,`	
14	`System.Globalization.NumberStyles.HexNumber) >`	Détection du type de certificat sur la base de la comparaison de numéro de série : - Complètement « propriétaire » ASIP Santé - C'est correct (cf. ci-après) mais déconseillé ! - Pourrait changer : de fait, cela à changer depuis le 24/06/2014 : la logique depuis est exactement l'inverse, rendant ce code non fonctionnel et non conforme aux référentiels de sécurité puisqu'on peut dès lors signer des challenges d'authentification avec la clé de signature. - Contraire aux bonnes pratiques Crypto : - Préférer l'utilisation de l' « usage » - Qui plus est l'usage est exposé par les APIs .NET csp.KeyNumber = (int) KeyNumber.Signature; par exemple (cf. [MANUEL_INST_UTIL])
15	`Int32.Parse(authentificationCertificate.SerialNumber,`	
16	`System.Globalization.NumberStyles.HexNumber)) {`	
17	`signatureCertificate = authentificationCertificate;`	
18	`authentificationCertificate = certificate;`	
19	`}`	
20	`}`	
21	`} /*end if certificate.Issuer */`	
22	`} /*end foreach*/`	
23	`} /*end constructor*/`	
24	`} /*end class*/`	
25	`} /*end namespace*/`	
26	`[...]`	

Tableau 8 : Exemple 3 – C#

8.3.3.2 Conclusion

- ce code n'est pas impacté à « T0 » par la migration Cryptolib CPS v4 vers Cryptolib CPS v5
  - car le package Cryptolib CPS v5 déploie les composants Cryptolib CPS v4
- ce code n'est pas impacté au moment de passer à l'exploitation du volet IAS de la CPS3 car
  - le CSP v5 est iso-fonctionnel avec le CSP v4
    - encore faut-il savoir que seul le niveau CSP est pertinent ici, et pas le niveau PKCS#11
- ce code ne s'inscrit pas du tout dans les bonnes pratiques cryptographiques (et dans les bonnes pratiques du génie logiciel tout court) et pose problème depuis le 24/06/2014, date à laquelle les cartes CPx contiennent des certificats ordonnés différemment.
- ce code doit être remplacé par les appels à l'API Cryptographique pertinente du .NET Framework

8.3.4 Exemple 4: étude d'un exemple en C# (2)

8.3.4.1 Code d'accès aux certificats CPx

Le code suivant utilise C# pour :

- récupérer les 2 certificats carte CPx en itérant dans le magasin de certificats Microsoft
- les identifier afin de faire ultérieurement soit une opération de signature soit une opération d'authentification.

Ce code fait implicitement 2 hypothèses :

1. Il table sur le fait qu'il n'y a que 2 certificats dans le magasin (les 2 certificats attachés à la carte CPx) ce qui n'est généralement pas le cas
   a. L'utilisateur peut avoir des bi-clés logiciels en plus des certificats CPx dans son magasin personnel
   b. Si le CCM est arrêté, le service de propagation Microsoft n'efface pas les certificats lors du retrait carte : tous les certificats de toutes les cartes vues par la machine sont présents en magasin
2. Il table cependant sur le fait que les éléments sont stockés ordonnés dans le magasin

#	Code	Commentaire
0	`[...]`	
1	`if (certificats.Count() == 2) {`	A NE PAS FAIRE : le magasin personnel de certificats peut contenir d'autres certificats que ceux associés à la CPS !
2	`  cpsCertificates = new CpsCertificates();`	
3	`  cpsCertificates.SignatureCertificate = certificats.ElementAt(0);`	A NE PAS FAIRE : personne ne garantit l'ordre des éléments en magasins ou dans le tableau mémoire
4	`  cpsCertificates.AuthenticationCertificate = certificats.ElementAt(1);`	
5	`}`	
6	`[...]`	

Tableau 9 : Exemple 4 – C#

Ce type de code n'est pas pérenne.

8.3.5 Exemple 5: étude d'un exemple en C++

8.3.5.1 Code d'accès à la CPx

Le code suivant utilise C++ et PKCS#11 pour récupérer les 2 certificats carte et les identifier afin de faire ultérieurement soit une opération de signature soit une opération d'authentification.

#	Code	Commentaire
0	[...]	Il s'agit plutôt d'un code cross-plateforme: l'intégration au niveau PKCS#11 est justifiée
1	CK_RV rc = CKR_OK;	
2	CK_OBJECT_CLASS objClass = CKO_CERTIFICATE; /* classe à rechercher */	
3	CK_OBJECT_HANDLE hCert;	
4	CK_BBOOL bFalse = CK_FALSE; /* rechercher public CKA_PRIVATE->CK_FALSE */	
5	CK_BBOOL bTrue = CK_TRUE; /* rechercher est token CKA_TOKEN->CK_TRUE */	
6	CK_ULONG ulMaxObjectCount = 1; /* Nombre maximum d'objets à récupérer */	
7	char label[] = "Certificat de Signature CPS"; /* Label de l'objet */	
8		
9	CK_ATTRIBUTE searchTemplate[]= { /* Template de recherche */	
10	{CKA_CLASS, &objClass, sizeof(objClass)},	
11	{CKA_TOKEN, &bTrue, sizeof(bTrue)},	Un vecteur de recherche d'objets est initialisé sur le label « Certificat de Signature CPS »
12	{CKA_PRIVATE, &bFalse, sizeof(bFalse)},	
13	{CKA_LABEL, label, strlen(label)}	
14	};	
15		
16	CK_ATTRIBUTE templateAttr[] = { /* Template de récupération de donnée */	
17	{CKA_VALUE, NULL_PTR, 0}	
18	};	
19		
20	CK_ULONG searchTemplateSize = sizeof(searchTemplate)/sizeof(CK_ATTRIBUTE);	
21	CK_ULONG templateAttrSize = sizeof(templateAttr)/sizeof(CK_ATTRIBUTE);	
22		
23	rc = (*pFunctionList->C_FindObjectsInit)(hSession, searchTemplate, searchTemplateSize);	Une recherche PKCS#11 est initialisée avec le vecteur
24		Il est aussi possible d'initialiser un template de recherche sur tous les objets avec la CKA_CLASS à CKO_CERTIFICATE, ulMaxObjectCount = 2 et d'itérer sur le keyUsage du certificat pour trouver celui qui correspond à notre usage.
25	hCert = CK_INVALID_HANDLE;	
26	rc = (*pFunctionList->C_FindObjects)(hSession, &hCert, ulMaxObjectCount, &ulMaxObjectCount);	
27	[...]	

Tableau 10 : Exemple 5 – C++

8.3.5.2 Conclusion

Ce code, qui provient du Manuel de Programmation dans sa partie « lecture de certificat », est uniquement valable sur le PKCS#11 de la Cryptolib CPS v5, du fait de l'utilisation « en dur » du label "Certificat de Signature CPS".

Pour rappel, pour la lecture, le Manuel de Programmation utilise le label pour sélectionner un objet de classe CKA_CLASS=CKO_CERTIFICATE puis passe au code de lecture du certificat proprement dit. Idéalement, il vaut mieux initialiser un template de recherche sur les objets CKA_CLASS=CKO_CERTIFICATE, itérer et parser les réponses et ressortir que le CKA_CLASS=CKO_CERTIFICATE avec le bon key usage. Ce qui amène 3 considérations :

1. des considérations de performances
2. des considérations de dépendances (besoin d'OpenSSL pour parser le certificat par ex.)
3. à cet endroit du manuel, le but recherché est de présenter la lecture de la valeur du certificat, pas sa sélection.

Une remarque afférente importante : un code utilisant l'objet de classe CKA_CLASS=CKO_CERTIFICATE ainsi récupéré ne pourra pas par la suite déclencher des opérations cryptographiques sur la clé correspondant à ce certificat. Il ne pourra donc pas en faire mauvais usage cryptographique, ce dont on cherche généralement à se prémunir.

Pour les opérations cryptographiques, le Manuel de Programmation utilise un template de recherche PKCS#11 avant de déclencher l'opération sur la clé privée.
Le code récupère un objet de classe CKA_CLASS=CKO_PRIVATE_KEY. Les références importantes sont alors: &cka_id_keyAuth ou &cka_id_keySign.
Les préconisations génériques du PKCS#11 pour discriminer une clé de signature d'une clé d'authentification sont d'examiner l'attribut CKA_DECRYPT (à False sur la clé privée de signature et CKA_DECRYPT à True sur la clé privée d'authentification), ce qui ne manque pas de poser un problème quand il y a plusieurs clés d'authentification (ce qui n'est pas le cas avec le PKCS#11 de la Cryptolib CPS).

Noter que PKCS#11 et CAPI ou TokenD ne fonctionnent pas sur ce même mode : là où il y a 3 objets en PKCS#11 (clé publique, clé privée et certificat), il n'y en a qu'un seul en CAPI (le certificat) et CAPI fait seul en interne le lien entre les 3 objets sous-jacents.

Le PKCS#11 de la Cryptolib CPS v5 assure un lien entre ces 3 objets via l'attribut CKA_ID (cf. Gestion des identifiants et des labels PKCS#11) conformément aux recommandations OASIS pour PKCS#11.

De fait, entre CAPI ou TokenD et Pkcs11, les niveaux sémantiques sont différents :

1. CAPI ou TokenD
    a. signature et authentification doivent être considérées au niveau fonctionnel, avec tous les aspects juridique que cela sous-tend
2. PKCS#11
    a. le niveau est plus technique, il n'y a pas vraiment d'authentification ou de signature, il n'y a que de la signature.
    b. Au niveau PKCS#11, il est possible de demander une opération de signature (C_Sign et CKA_SIGN à true sur les 2 clés d'authentification et de signature).

La recherche d'objets dans les magasins cryptographiques est un réel souci.
Ce souci est commun à toutes les technologies (Keystore Java, Token PKCS#11, Magasins de certificats Microsoft Windows…).
Il rappelle les problématiques génériques de recherche dans des bases de données (d'ailleurs le pattern DAO avec un finder puis un findByPrimaryKey s'applique très bien).
Il s'agit de bien isoler cette recherche, de bien y respecter les bonnes pratiques de développement (paramètres en configuration) et de bien en maîtriser les effets (dossier de conception).

8.3.6 Exemple 6: cas de multiplication des filières d'accès carte

8.3.6.1 Architecture LPS considérée

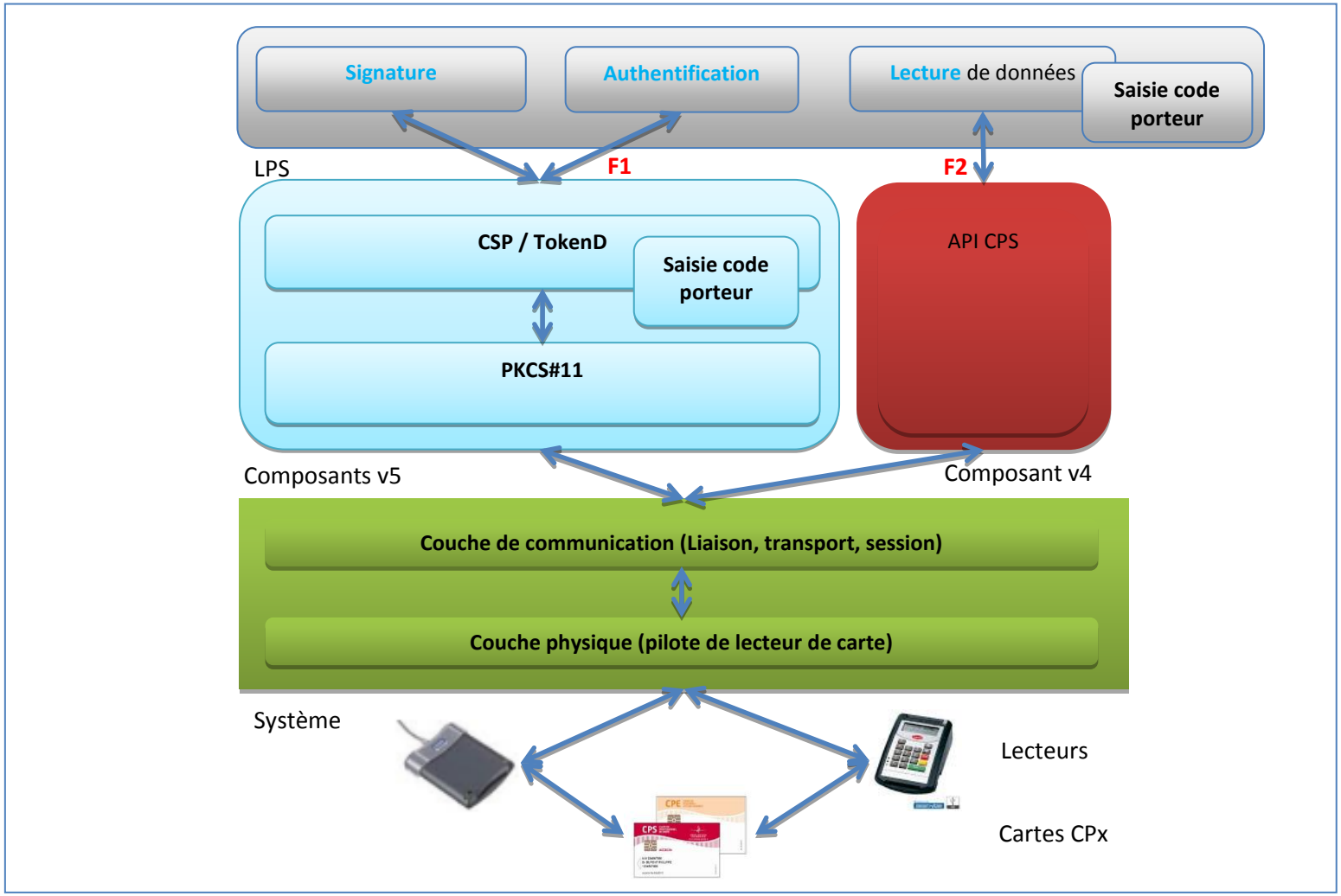


Figure 7 : Cas de multiplication des filières d'accès carte CPx

➔ L'utilisation parallèle de F1 et F2 entraine potentiellement une double saisie de code porteur.

8.3.6.2 Code d'accès à la carte : étude d'un exemple en C#

Cette situation est illustrée ci-après avec un code de LPS DMP-Compatible.

#	Code	Commentaire
0	[...]	
1	private void SaisirCodeCPS_Execute(object parameter) {	Le logiciel demande une saisie de code porteur CPS pour autoriser l'accès aux dossiers patients
2	string codePin = (parameter as PasswordBox).Password;	
3	ResultatSaisiePIN resultatSaisiePIN = cpsServices.SaisirPIN(NumSession, codePin);	
4	[...]	
5	}	
6	[...]	
7	public ResultatSaisiePIN SaisirPIN(int numSession, string codePIN) {	Cette saisie de code porteur passe par l'API CPS
8	IntPtr etatCartePtr = new IntPtr();	
9	StringBuilder codeSB = new StringBuilder(codePIN);	
10	CPS_StatutService cps_StatutService = new CPS_StatutService();	
11	CPSRetourAPI retourCPS = ParseRetour(CPS_PresCodePorteur(new IntPtr(numSession), codeSB, ref cps_StatutService, ref etatCartePtr));	
12	return new ResultatSaisiePIN(retourCPS, numSession,	
13	(CPS_StatutService)cps_StatutService, etatCartePtr.ToInt32());	
14	}	
15	[...]	
16	[DllImport("cpsw32", EntryPoint = "CPS_PresCodePorteur", SetLastError =	
17	true)] private static extern UInt16 CPS_PresCodePorteur(IntPtr numSession,	
18	StringBuilder pCodePorteur, ref CPS_StatutService pstatutService, ref IntPtr pEtatCarte);	
19	[...]	
20	public string EnvoyerDocumentsVersDmp(ProfessionnelSanteDMP professionnel,	Parallèlement, le logiciel utilise les API DMP, qui ont besoin de la carte CPx, via le CSP
21	List<DocumentDMP> documents, string applicationInstanceID) {	
22	X509Certificate2 x509Certificate2 = smartCard.DonneCPSCertificates().SignatureCertificate;	
23	[...]	
24	List<X509Certificate2> certis;	
25	if (certis.Count() == 2) {	
26	cpsCertificates = new CpsCertificates();	
27	cpsCertificates.SignatureCertificate = certis.ElementAt(0);	
28	cpsCertificates.AuthenticationCertificate = certis.ElementAt(1);	
29	} else { [...] }	

Tableau 11 : Exemple 6 – Multiplication des filières d'accès carte

Cette situation s'accompagne généralement de l'affichage de plusieurs fenêtres de saisie du code porteur à l'utilisateur (problème de cohérence et d'ergonomie). Pour la connexion sur l'application, par exemple, on verrait apparaître :

Figure 8 : Fenêtre de saisie du code porteur CPx par le logiciel lui-même

Tandis que pour la communication avec le DMP / l'accès via le CSP à la carte, la boite de dialogue suivante apparaît :

Figure 9 : Fenêtre de saisie du code porteur CPx via le CSP Cryptolib CPS v5 ASIP santé

8.3.6.3 Conclusion

- ce code n'est pas impacté à « T0 » par la migration Cryptolib CPS v4 vers Cryptolib CPS v5 car le package Cryptolib CPS v5 déploie les composants Cryptolib CPS v4
- ce code est impacté au moment de passer à l'exploitation du volet IAS de la CPS3
    - 2 demandes de code porteur apparaissent au lieu d'une seule
        - Du fait de l'utilisation des 2 filières API CPS et CSP
            - Qui sont indépendantes l'une de l'autre
        - Quel que soit l'ordre d'appel
            - Composant (CSP ou PKCS#11) de la Cryptolib CPS v5 puis API CPS
            - API CPS puis composant v5
    - Ce code est particulièrement impacté avec les lecteurs PC/SC
        - Avec les lecteurs PSS, si un composant v5 est utilisé avant l'API CPS, l'API CPS est avisée par le GALSS que le code porteur a été présenté
    - La première soumission de code porteur peut se faire par PKCS#11 ou par CSP directement
    - Ces changements peuvent se faire selon une trajectoire de migration vers le volet IAS
        - Changements qui peuvent d'autant mieux s'appréhender que la Cryptolib CPS v5 est disponible pour intégration depuis 2 ans sur le site de l'ASIP Santé
        - En prévoyant une phase de développement visant à rationaliser les accès carte CPx

Le problème se règle de 2 façons :

1- Utiliser les composants v5 avant l'API CPS
    2. Avec les lecteurs PSS, le code porteur ne sera pas redemandé
    3. Avec les lecteurs PC/SC : L'état carte « code porteur présenté » exposé par l'API CPS peut être ignoré
        a. Gestion de l'erreur de lecture de ressource protégée par code porteur dévolue à l'application

2- En suivant les préconisations de l'ASIP Santé ci-après

8.3.6.4 Un autre cas problématique : l'accès direct à la liaison lecteur (PC/SC ou GALSS):

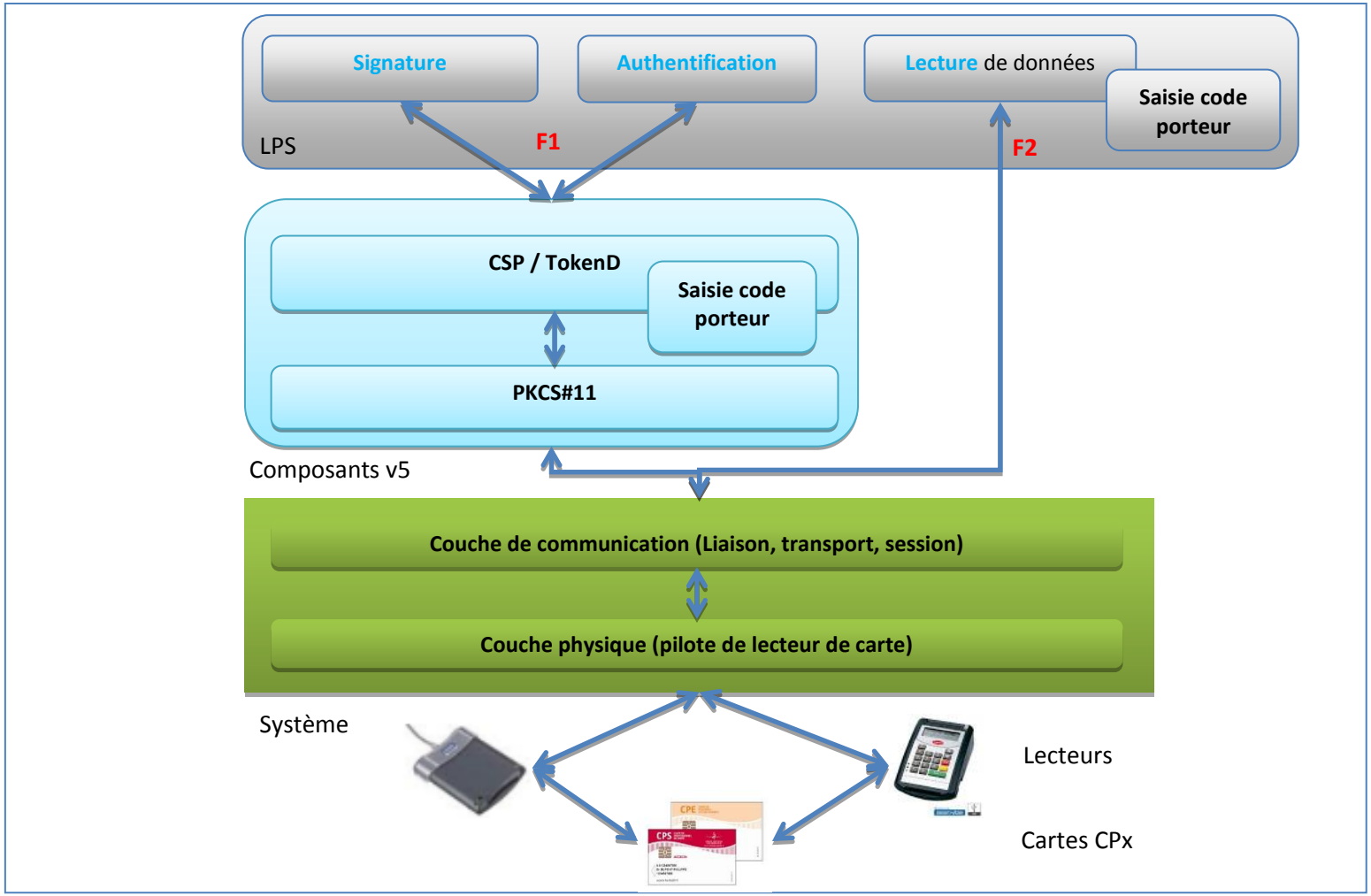


Figure 10 : Cas de multiplication des filières d'accès carte CPx (cas de l'API de Facturation SESAME Vitale 1.40)

➔ Dans ce dernier cas, l'utilisation parallèle de F1 et F2 entraine aussi potentiellement une double saisie de code porteur.

Le problème se règle de 2 façons :

1- En imposant le code porteur via F1 avant de passer par F2, qui présuppose de son côté que le code porteur a déjà été préalablement soumis et donc ne le redemande pas
2- En suivant les préconisations de l'ASIP Santé ci-après

8.3.6.5 Un autre cas problématique : l'accès en parallèle aux composants CSP (de la Cryptolib CPS v5) / PKCS#11 de la Cryptolib CPS v4

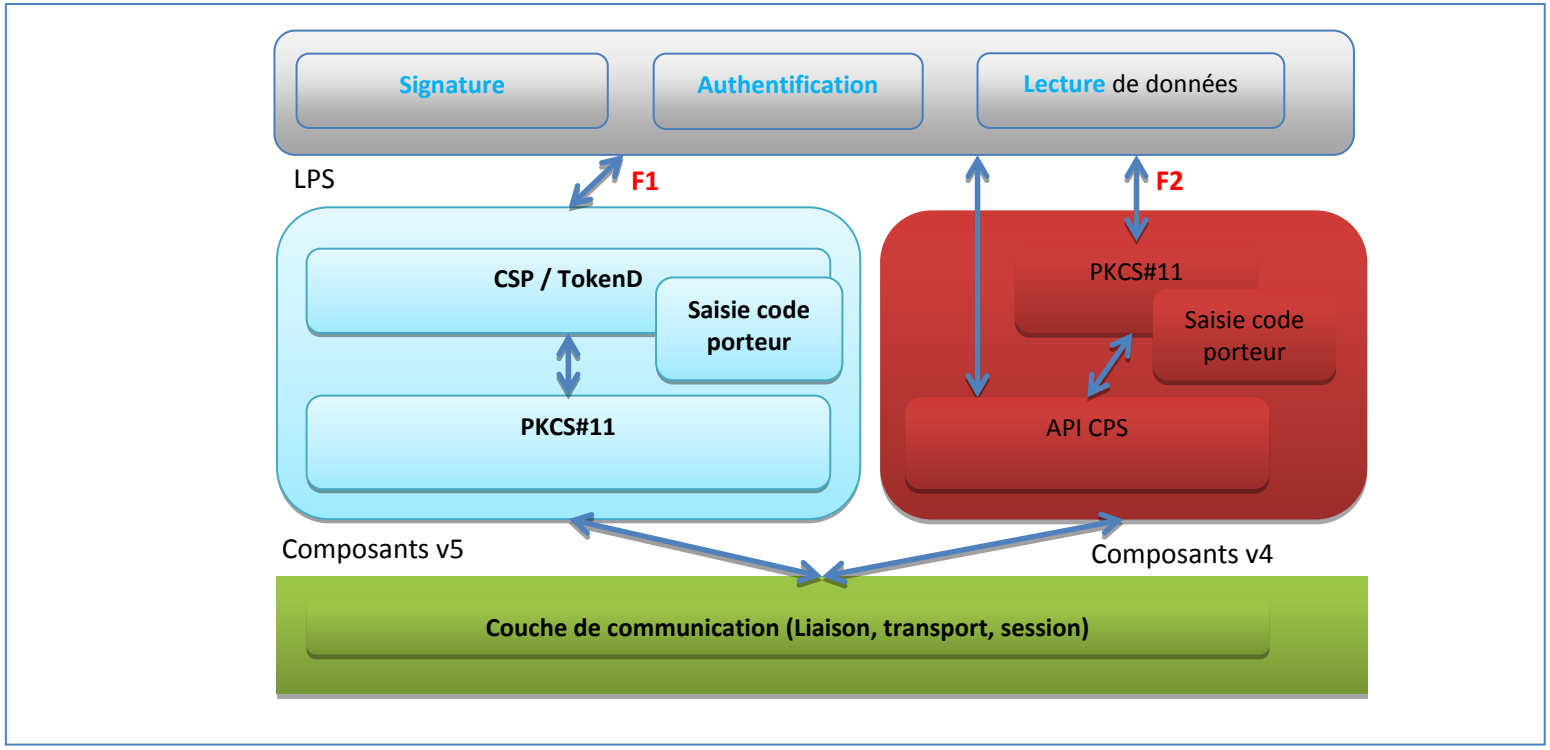


Figure 11 : Cas de multiplication des filières d'accès carte CPx

Le problème se règle d'une seule façon :

1- En suivant les préconisations de l'ASIP Santé ci-après

8.4 Illustration de la préconisation d'architecture LPS

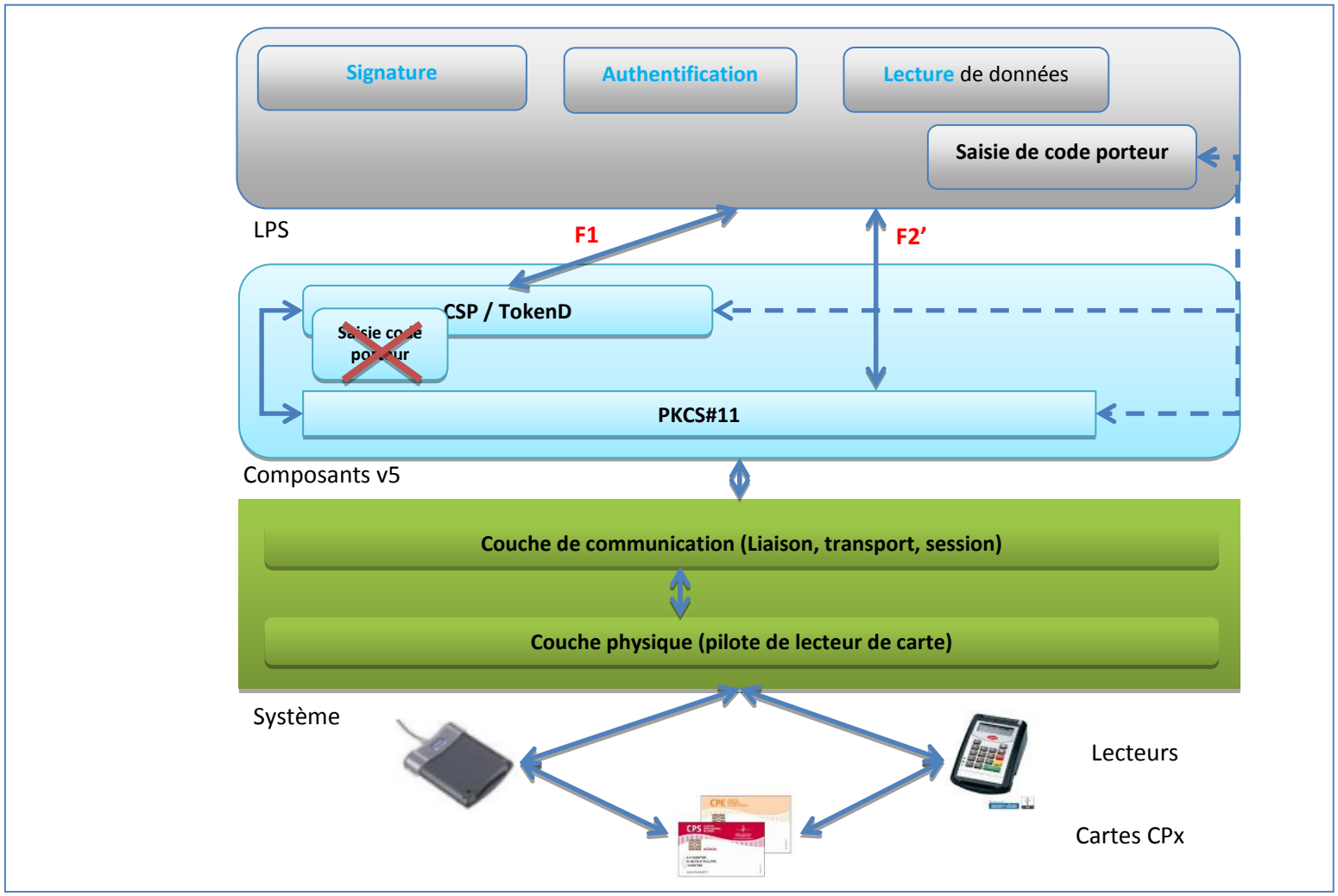


Figure 12 : Préconisation d'accès à la carte CPx

L'API CPS n'est plus maintenue par l'ASIP Santé. L'application est responsable de la saisie du code porteur (sans cache).
L'utilisation parallèle de F1 et F2' n'entraine pas de double saisie de code porteur, le module commun PKCS#11 gérant une session avec code porteur présenté.

9 Stratégie de migration de la Cryptolib CPS v4 vers la Cryptolib CPS v5

9.1 Rappels

La stratégie de l'ASIP Santé est de ne plus distribuer que des interfaces « standards » (PKCS#11 et CSP ou TokenD) et d'abandonner la distribution et le support de l'API CPS.

Les LPS exploitant la carte CPx ont tout intérêt à bien s' « architecturer » afin de bien isoler la partie « accès carte CPx » en leur sein.

Les éditeurs ont besoin d'anticiper l'effort que représente pour leur(s) application le fait de « migrer » vers du « standard ».

L'équivalent de l'API CPS dans les LPS peut:

- être réalisé à l'aide de la nouvelle implémentation du PKCS#11 de la Cryptolib CPS v5 (voir tableau de correspondance en fin de document)
- prendre la forme d'un package de classes dédiées isolé et chargé dynamiquement (typiquement une DLL)
- prendre la forme d'un package de classes dédiées embarqué et donc utilisé « statiquement »
- est du ressort et de la responsabilité de l'éditeur

9.2 Stratégie nominale

En apportant les composants de la Cryptolib CPS v4 utilisant le volet CPS2ter de la carte CPS3 et les nouveaux composants utilisant le volet IAS de la carte CPS3, le package actuel de la Cryptolib CPS v5 permet aux éditeurs et aux intégrateurs de mettre en place une stratégie progressive de migration de l'utilisation du volet 2Ter vers l'utilisation du volet IAS de la CPS3.

En particulier, un jalon « T0 » peut être prévu :

- les nouvelles applications
    - peuvent utiliser directement les nouveaux composants v5 correspondant au volet IAS de la CPS3
- les anciennes applications
    - peuvent déployer la nouvelle Cryptolib CPS v5
        - tout en utilisant les anciens composants v4
    - dans une phase de « compatibilité ascendante » sans besoin de portage

Par ailleurs, les composants v4 sont destinés à être abandonnés : les applications doivent programmer leur migration vers les nouveaux composants de la Cryptolib CPS v5 (PKCS#11 v5 ou CSP / TokenD v5).

Les applications – en particulier les applications existantes – doivent donc prévoir un jalon « T1 » à partir duquel

- elles auront abandonné l'utilisation des composants v4, et donc l'utilisation systématique du volet CPS2Ter
- elles basculeront vers l'utilisation systématique des interfaces système CSP / TokenD ou PKCS#11 de la Cryptolib CPS v5

Ce document permet d'évaluer les tâches à réaliser pour y parvenir.

9.3 Exemple 1 : Chemin de migration v4 / v5 pour une application utilisant l'API CPS

9.3.1 Schéma de principe

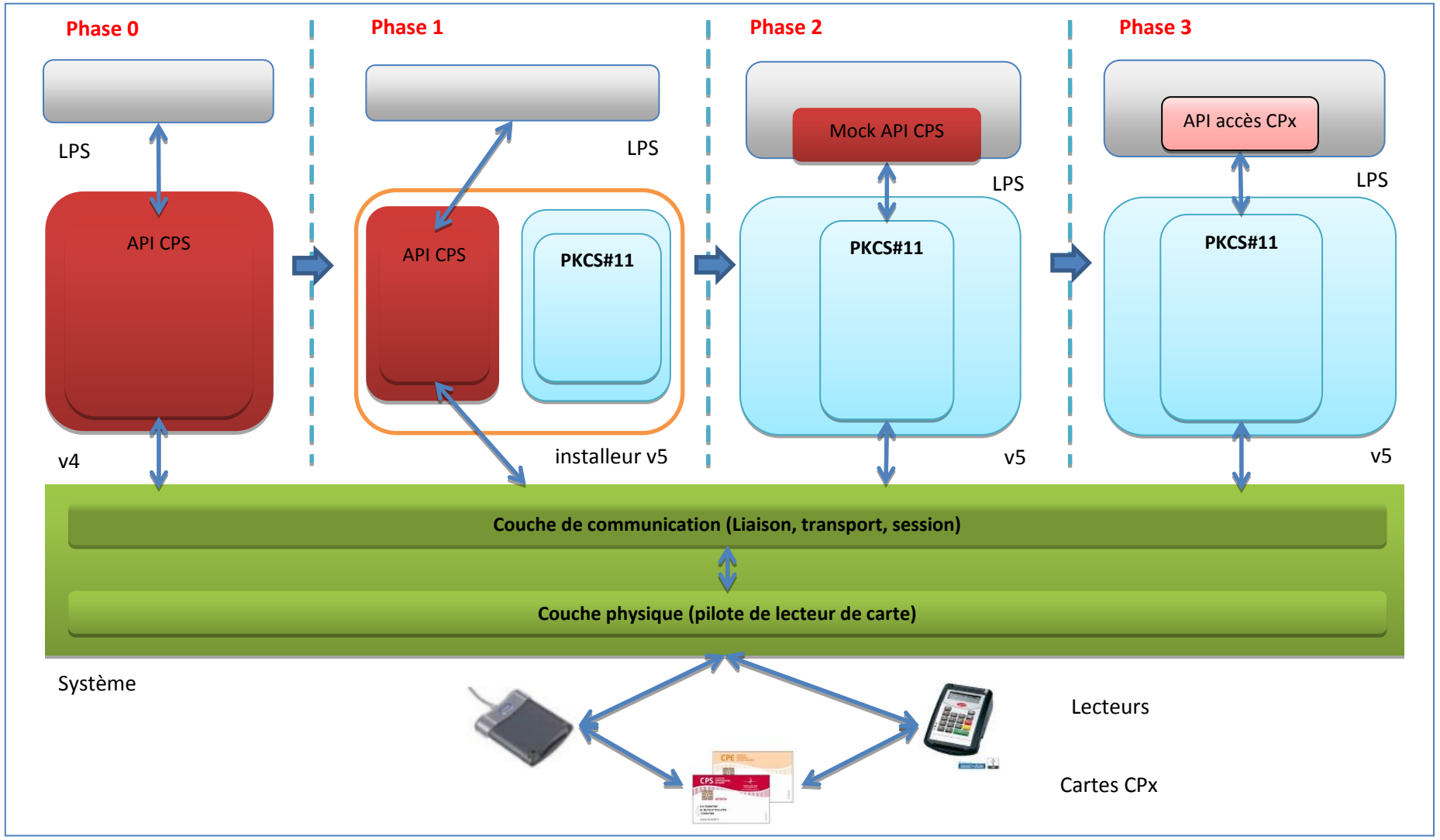


Figure 13 : Chemin de migration v4 / v5 pour une application utilisant l'API CPS

9.3.2 Description

Phase	Description	Développements	Coûts de transition depuis la phase précédente
0	LPS dans sa version courante (**v0**), supportant la Cryptolib CPS **v4** et accédant à la carte CPx via l'API CPS	N/A	N/A
1	LPS dans sa version **v1**, supportant la Cryptolib CPS **v5** et accédant à la carte CPx via l'API CPS, i.e. via la partie Cryptolib CPS **v4** apportée par l'installeur Cryptolib CPS **v5**	N/A	Tests Q&A avec la Cryptolib CPS v5 Mise à jour des documents utilisateurs Formation du Support Déploiement de la Cryptolib CPS v5 à la place de la Cryptolib CPS v4
2	(optionnel : peut-être substituée directement par la phase 3) LPS dans sa version **v2**, supportant la Cryptolib CPS **v5** et accédant à la carte CPx via une API CPS reconstituée au-dessus de l'API PKCS#11 sur la base : - de l'identification du sous ensemble d'appels effectivement utilisés (pas nécessaire de recoder toutes les méthodes de l'API CPS) - du tableau établissant la relation 1-1 entre API CPS et PKCS#11 de la Cryptolib CPS v5 fourni en annexe	Développement d'un équivalent à l'API CPS au-dessus du PKCS#11 de la Cryptolib CPS v5 pour limiter les changements d'appels disséminés dans l'application. Cet équivalent peut prendre la forme d'une DLL indépendante ou d'une arborescence de classes dédiées. Dans tous les cas, le pattern adéquat est de type (connection) factory. Cette phase, en reprenant les signatures de l'API CPS, permet à l'application de ne se concentrer que sur la carte sans toucher au reste du code (en particulier, cas de LPS où les structures de l'API CPS sont reprises « telles quelles » dans le reste de l'application). Cette phase permet aussi de se placer directement dans les bonnes pratiques de rationalisation des filières d'accès carte CPx : les problèmes de double saisie de code porteur sont résolus.	Identification des appels vers l'API CPS effectivement utilisés par le LPS Identification des équivalents PKCS#11 à l'aide du tableau en annexe Développements Tests Q&A avec la Cryptolib CPS v5 Déploiement
3	(optionnel : dépend du degré d'aboutissement de la phase précédente) LPS dans sa version **v3**, supportant la Cryptolib CPS **v5** et accédant à la carte CPx via l'API PKCS#11	Développement d'une API d'accès à la carte CPx en relation avec le besoin réel de l'application en termes d'accès carte CPx	Définition de l'API Développements Tests Q&A avec la Cryptolib CPS v5 Déploiement

Tableau 12 : Chemin de migration v4 / v5 pour une application utilisant l'API CPS

9.3.3 Coûts

Hypothèses chiffrage phase 2
Manuels d'installation et d'utilisation de la Cryptolib CPS v5 et guide de programmation PKCS#11 lus et assimilés
Expression de besoins et spécifications relatifs à l'accès carte CPx (ergonomie, fonctionnalités…) disponibles
Tests unitaires de développements relatifs à l'accès carte CPx via l'API CPS disponibles
Développements dédiés au portage (pas d'autre développement, correction de bogue ou évolution en parallèle)

Tableau 13 : Hypothèses chiffrage phase 2

Coûts associés à la phase 2		
Description	**Chiffrage unitaire**	**Consolidé**
Nombre de fonctions API CPS utilisées sur les 28 fonctions exposées par l'API CPS	15	
Montée en compétence sur le PKCS#11	5 jours	5 jours
Effort de portage vers le PKCS#11 pour les 3 premières fonctions de l'API CPS	4 jours de portage / fonction	12 jours
Effort de portage vers le PKCS#11 pour les fonctions supplémentaires de l'API CPS	0,5 jour par fonctions supplémentaires	(15-3)*0.5 = 6 jours
	Total	env. 25 jours de développements

Tableau 14 : Coûts migration v4 / v5 pour une application utilisant l'API CPS (phase 2)

Les coûts associés à la phase 3 sont quant à eux directement liés à la conception du LPS qui fait l'objet de la migration :

- Complétude des expressions de besoins, spécifications et dossiers de conception
    - Accès carte CPx mais plus globalement comportement du LPS dans sa globalité
- Passage par la phase 2
    - ou non
- Réutilisation des structures de l'API CPS dans le code métier
    - ou non
- Volonté de rework de la partie accès carte CPx une fois le mock d'API CPS réalisé
    - ou non

9.4 Exemple 2 : Chemin de migration v4 / v5 pour une application utilisant le PKCS#11

9.4.1 Schéma de principe

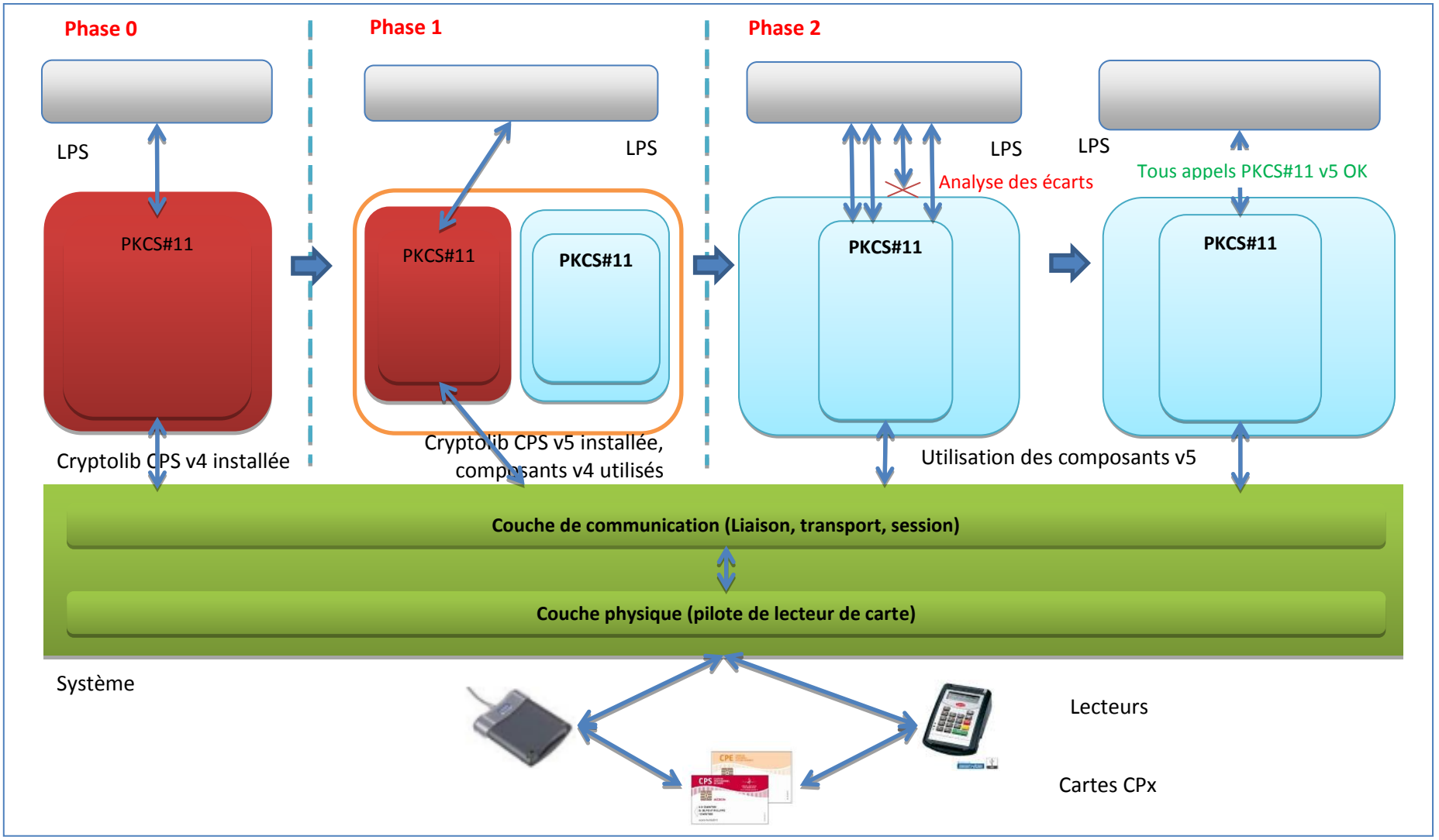


Figure 14 : Chemin de migration v4 / v5 pour une application utilisant le PKCS#11

9.4.2 Description

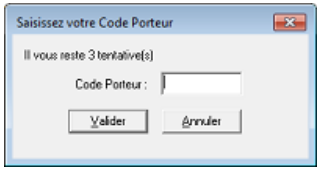
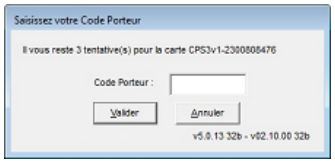
Phase	Description	Développements	Couts de transition depuis phase précédente
0	LPS dans sa version courante (**v0**), supportant la Cryptolib CPS **v4** et accédant à la carte CPx via l'API PKCS#11	N/A	N/A
1	LPS dans sa version **v1**, supportant la Cryptolib CPS **v5** et accédant à la carte CPx via le PKCS#11 de la v4, i.e. via la partie Cryptolib CPS **v4** apportée par l'installeur Cryptolib CPS **v5**	N/A	Tests Q&A avec la Cryptolib CPS v5 Mises à jour des documents utilisateurs Formation du Support Déploiement de la Cryptolib CPS v5 à la place de la Cryptolib CPS v4
2	LPS dans sa version **v2**, supportant la Cryptolib CPS **v5** et accédant à la carte CPx via le PKCS#11 de la **v5**	Adaptation des développements PKCS#11 à la v5 après identification des objets PKCS#11 utilisés et de leurs équivalents en **v5**.	Identification des objets PKCS#11 effectivement utilisés par le LPS Identification des équivalents PKCS#11 à l'aide de ce guide Développements Tests Q&A avec la Cryptolib CPS v5 Déploiement

Tableau 15 : Chemin de migration v4 / v5 pour une application utilisant l'API CPS

10Aperçu général des différences entre Cryptolib CPS v4 et v5

Le tableau ci-dessous décrit les différences de gestion des fonctionnalités entre les Cryptolib CPS v4 (CPS2ter) et Cryptolib CPS v5 (CPS3, volet IAS) et les impacts techniques de ces évolutions. Ce tableau concerne les **applications** qui **migrent explicitement** vers les **composants Cryptolib CPS v5** contenus dans l'installeur 5.0+ après être passé par la phase de compatibilité (voir chapitre 7 plus haut):

#	Module	OS	Fonctionnalité	En Cryptolib v4 (CPS2ter)	En Cryptolib v5 (CPS3)	Motif de l'évolution	Impact technique estimé	Détails Cf. §
1	PKCS#11		Changement de nomenclature des librairies	Noms des librairies préfixés par cps ou libcps	Noms des librairies préfixés par cps3 ou libcps3	Copie des composants v4 et v5 par l'installeur Cryptolib CPS v5		11.1.1
2	PKCS#11		Nouvelle librairie 64bit	Une unique DLL PKCS#11 32bit est fournie	Deux DLLs au lieu d'une : 64b et 32b	Support du 64b		11.1.2
3	PKCS#11		Gestion des identifiants et des labels PKCS#11	PKCS#11 permet de discriminer efficacement les différents objets cryptographiques, en particulier les clés de signature et les clés d'authentification	Changement de l'algorithme de récupération des bi-clés	Recommandations PKCS#11		11.2

#	Module	OS	Fonctionnalité	En Cryptolib v4 (CPS2ter)	En Cryptolib v5 (CPS3)	Motif de l'évolution	Impact technique estimé	Détails Cf. §
4	PKCS#11		Identification de la carte insérée	L'utilisation de la DLL PKCS#11 fournie permet de mettre en œuvre un mécanisme de détection de la présence et de l'arrachage de la carte CPS efficace	Changement de l'algorithme de détection de la présence de la carte	Recommandations PKCS#11		11.3
5	PKCS#11		Saisie du code porteur pour chaque application	Le code porteur est entré une seule fois	Le code porteur est entré 1 fois par processus	Recommandations de sécurité		11.4.1
6	PKCS#11		Boîte de saisie du code porteur hors DLL PKCS#11	La librairie PKCS#11 gère la saisie du code porteur	L'application utilisant la librairie PKCS#11 est responsable de la saisie du code porteur	Recommandations PKCS#11		11.4.2
7	PKCS#11 CSP		Changement d'aspect de la fenêtre de saisie du code porteur sous Windows			Amélioration de l'ergonomie et du support		11.4.3

#	Module	OS	Fonctionnalité	En Cryptolib v4 (CPS2ter)	En Cryptolib v5 (CPS3)	Motif de l'évolution	Impact technique estimé	Détails Cf. §
8	PKCS#11		Calcul de signature	La librairie PKCS#11 supporte la signature de hash pré-calculé	La librairie PKCS#11 calcule elle-même une partie du hash, la carte finit le calcul et signe	Recommandations RGS		11.5
9	PKCS#11		Changement de comportement de C_WaitForSlotEvent	Le mode bloquant est supporté	Le mode bloquant n'est plus supporté suite à des problèmes de performances détectés avec Firefox	Améliorations des performances		11.6
10	PKCS#11		L'API CPS est dépréciée au profit de l'interface PKCS#11			Standardisation		11.7
11	CSP		Propagation des certificats		Sous Windows 7, le service de propagation de certificat alimente le magasin en parallèle du CCM	Recommandation architecture Microsoft		12.1

#	Module	OS	Fonctionnalité	En Cryptolib v4 (CPS2ter)	En Cryptolib v5 (CPS3)	Motif de l'évolution	Impact technique estimé	Détails Cf. §
12	Général		Système 64bit avec Cryptolib CPS 64bit	Les systèmes 32bit sont nativement supportés. La Cryptolib CPS v4 s'installe sur les systèmes 64bit. Certaines fonctionnalités offertes par le système 64bit ne sont toutefois pas exploitables.	Les systèmes 64bit sont nativement supportés	Support du 64b		13.1
13	Général		Installations Full PC/SC et GALSS factorisées	Deux installeurs distincts sont disponibles : un pour la filière GALSS, un autre pour la filière Full PC/SC	Les deux filières PC/SC et GALSS sont factorisées	Simplification du processus d'installation		13.2
14	Général		Possibilité de personnaliser les installations		L'installeur peut prendre des paramètres en entrée	Amélioration du déploiement, notamment dans les SI de santé		13.3

#	Module	OS	Fonctionnalité	En Cryptolib v4 (CPS2ter)	En Cryptolib v5 (CPS3)	Motif de l'évolution	Impact technique estimé	Détails Cf. §
15	Général		Configuration de la Cryptolib CPS	La Cryptolib CPS v4 se configure via un fichier unique	La Cryptolib CPS v5 se configure via la base de registre, des variables d'environnement, des fichiers de configuration	Recommandations éditeurs OS		13.4
16	Général		Nouvelle version de CPS-Gestion	CPS-Gestion v5.xx trace les appels à l'API CPS	Le nouveau CPS-Gestion v6.xx trace les appels PKCS#11 L'ancienne version est toujours installée pour compatibilité sous la dénomination « 2ter »	L'API CPS est dépréciée au profit de l'interface PKCS#11 (cf. point 10 ci-dessus)		13.5
17	Général		Changement des menus Démarrer sous Windows	Menu Démarrer / Programmes / **Cryptolib CPS**	Menu Démarrer/ Programmes / **Santé Social / CPS**	Cohérence du poste de travail		13.6
18	Général		Changement de nom du package RPM Linux	cryptolib	Cryptolib	Cohérence des noms		13.7

#	Module	OS	Fonctionnalité	En Cryptolib v4 (CPS2ter)	En Cryptolib v5 (CPS3)	Motif de l'évolution	Impact technique estimé	Détails Cf. §
19	Tokend		Changement de nom du Tokend	GIP-CPS.tokend	CPS3.tokend	Cohérence des noms		13.7
20	Général		Gestion du cache des fichiers carte	Le fichier ccert.bin sert de cache de certificats	La Cryptolib CPS v5 utilise un cache de fichiers	Amélioration des performances		14
21	Général		Changement des données métiers	Le volet CPS2ter est utilisé.	La Cryptolib CPS v5 utilise le volet IAS.	Amélioration de l'intégration de la carte CPS3 dans les processus Santé Social		15.1

Tableau 16 : Résumé des impacts migration Cryptolib CPS v5

11Description des modifications PKCS#11

Cette section s'adresse aux applications utilisant la carte CPx depuis la couche PKCS#11.

Elle concerne par exemple les applications qui font appel à une applet Java qui utilise PKCS#11 pour parler à la carte CPx.

Elle concerne plus généralement tout éditeur ayant choisi PKCS#11 pour interfacer son application avec la carte CPS et désireux de migrer du volet CPS2Ter vers le volet IAS de la CPS3.

11.1 Chargement des DLL PKCS#11

11.1.1 Changement de nomenclature des librairies

Impact 1

Les noms des librairies PKCS#11 de la Cryptolib CPS v5 ont changé

Les applications accédant à la carte via PKCS#11 doivent charger la librairie PKCS#11 CPS3 adéquate.

Solution

Changement de l'algorithme de chargement de la librairie PKCS#11

Pour être compatible avec la Cryptolib CPS v5, l'algorithme de chargement de la librairie PKCS#11 doit être modifié de façon à pouvoir gérer l'ensemble des versions des Cryptolib CPS susceptibles d'être présentes sur le poste.

Cf. **DETECT_CRY**

Applications impactées

Applications accédant à la carte via PKCS#11 (Ex. : applications utilisant Java / IAIK).

Tableau 17 : Impact : Chargement des DLL PKCS#11

11.1.2 Nouvelle librairie 64bit

Impact 2		

La Cryptolib CPS v5 est disponible en 32b et 64 b sous Windows

La Cryptolib CPS v4 était disponible en 32b uniquement.

Solution

Changement de l'algorithme de chargement de la DLL PKCS#11

Pour être compatible avec la Cryptolib CPS v5, l'algorithme de chargement de la DLL PKCS#11 doit être modifié de façon à pouvoir gérer l'ensemble des versions des Cryptolib CPS susceptibles d'être présentes sur le poste.

Les applications 64bit peuvent utiliser la nouvelle DLL :

- %WINDIR%\System32\cps3_pkcs11_w64.dll

La librairie 32bit se nomme désormais cps3_pkcs11_w32.dll :

- %WINDIR%\System32\cps3_pkcs11_w32.dll sur OS 32bit
- %WINDIR%\SysWOW64\cps3_pkcs11_w32.dll sur OS 64bit

Cf. **MANUEL_PROGMANUEL_PROG**

Cf. **DETECT_CRY**

Applications impactées

Applications accédant à la carte via PKCS#11 (Ex. : applications utilisant Java / IAIK).

Tableau 18 : Impact : Chargement des DLL PKCS#11

11.2 Gestion des identifiants et des labels PKCS#11

Impact 3a		

Les identifiants et les labels PKCS#11 ont changé

Certaines applications utilisent des identifiants des clés ou des labels d'objets PKCS#11 codés en dur. Ces identifiants peuvent être spécifiques à la version Cryptolib CPS v4.

1- La manière de calculer les identifiants différant selon les versions de Cryptolib CPS, ces applications ne pourront pas signer des documents si la Cryptolib CPS v5 est installée.

2- Les identifiants de certificats de signature et d'authentification ont changé

   a. Les chaines de caractères « (authentification) » ou « (signature) » sont remplacées par « Certificat d'Authentification CPS » ou « Certificat de Signature CPS »

Solution

Déterminer l'ID à partir de la clé publique ou du certificat et du label

A chaque objet PKCS#11 est associé un attribut CKA_ID.
Cette propriété permet d'identifier l'objet avec lequel on veut effectuer une opération.
Cette implémentation est conforme aux recommandations OASIS pour PKCS#11 ("CKA_ID [is] mandatory for all objects, with all related objects (e.g. private key and associated certificate) being linked via a common CKA_ID, and to make CKA_LABEL mandatory for at least private-key objects").

Cf. **MANUEL_PROGMANUEL_PROG**

Applications impactées

Applications accédant à la carte via PKCS#11 (Ex. : applications utilisant Java / IAIK)

Tableau 19 : Impact : Changement d'identifiants et de labels PKCS#11

Cryptolib CPS	Carte	Key / Certificat Signature ID	Key / Certificat Authentification ID	Key / Certificat Authentification Technique ID
v4	*CPS2ter*[2]	*01000000*	*04000000*	*N/A*
	CPS3	01000000	04000000	N/A
v5	*CPS2ter*	*1217*	*1216*	*N/A*
	CPS3	E828BD080F8025000001FF001001	E828BD080F8025000001FF001002	E828BD080F8025000001FF001003

Tableau 20 : Liste des identifiants de clé

[2] Les informations concernant la carte CPS2ter sont fournies à titre indicatif : depuis mars 2014, toutes les cartes en exploitation sont des CPS3.

Impact 3b

La manière de calculer les identifiants de clé a changé

A chaque objet PKCS#11 est associé un attribut CKA_ID.
Cette propriété permet d'identifier l'objet avec lequel on veut effectuer une opération.
Certaines applications utilisent des identifiants des clés codés en dur, spécifiques à la version Cryptolib CPS v4.
La manière de calculer les identifiants différant selon les versions de Cryptolib CPS, ces applications ne pourront pas signer des documents si la Cryptolib CPS v5 est installée.

Solution

Recherche exhaustive

Pour chaque clé, trois identifiants sont possibles, il suffit de tester ces trois valeurs.

Cf. **MANUEL_PROGMANUEL_PROG**

Applications impactées

Applications accédant à la carte via PKCS#11 (Ex. : applications utilisant Java / IAIK)

Tableau 21 : Impact : Changement Identifiants de clés

11.3 Identification de la carte insérée

Impact 4 

Changement de comportement de la fonction C_GetTokenInfo

Les applications vérifient périodiquement que la carte utilisée lors de l'authentification SSL est toujours présente dans le lecteur.
Pour cela, elles comparent le SerialNumber issu de la structure TOKEN_INFO avec une variable tlsCardSerial obtenue à partir de l'extension privée gipCardID du certificat d'authentification.

Lorsqu'on utilise une carte CPS3 avec la Cryptolib CPS v5, la variable SerialNumber remontée par la fonction C_GetTokenInfo ne correspond plus à l'identifiant logique de la carte mais au numéro de série de la carte.

Solution

Déterminer l'identifiant logique de la carte à partir du label

Il faut obtenir l'identifiant logique de la carte à partir du label et non plus en utilisant le paramètre SerialNumber.

Cf. **MANUEL_PROGMANUEL_PROG**

Applications impactées

Applications utilisant la fonction C_GetTokenInfo du PKCS#11 pour gérer la présence de la carte.

Tableau 22 : Impact : Changement comportement de la fonction C_GetTokenInfo

Cryptolib CPS	Carte	SerialNumber (TOKEN_INFO)	Label (TOKEN_INFO)
v4	*CPS2ter*[3]	*Identifiant logique de la carte*	*CPS-IdCarteLog*
	CPS3	Identifiant logique de la carte	CPS-IdCarteLog
v5	*CPS2ter*	*Identifiant logique de la carte*	*CPS2ter-IdCarteLog*
	CPS3	Numéro de série de la carte	CPS3v1-IdCarteLog

Tableau 23 : Gestion de la session

[3] Les informations concernant la carte CPS2ter sont fournies à titre indicatif : depuis mars 2014, toutes les cartes en exploitation sont des CPS3.

11.4 Gestion du code porteur

11.4.1 Saisie du code porteur pour chaque application

Impact 5		

Changement de la gestion du code porteur

Le code porteur est demandé à chaque rechargement de la librairie PKCS#11.

Lorsqu'une application lance plusieurs processus (« fork »), plusieurs saisies de code porteur peuvent être demandées.

Solution

Revoir l'architecture de la solution

Revue d'architecture logicielle pour remplacer les sous-processus par autre chose.

Cf. **MANUEL_PROGMANUEL_PROG**

Cf. **8 Préconisations d'architectures logicielles pour les LPS**

Applications impactées

Applications accédant à la carte via PKCS#11 (Ex. : applications utilisant Java / IAIK)

Tableau 24 : Impact : Changement de la gestion du code porteur

11.4.1.1 Cas particulier des navigateurs Web

La gestion des processus est différente d'un navigateur à l'autre et nous ramène au chapitre 7 (suivi des standards) et au chapitre 8 (architecture).

Ces gestions différentes sont autant de sources possibles de comportements différents vis-à-vis de la saisie du code porteur.

Le cas général peut être décrit en première approximation par :

- « 1 tab = 1 processus »
- sauf sous Mozilla Firefox où un processus unique est généralement constaté.

Ces gestions des processus entrainent des comportements SSL différents et donc des comportements de demande de saisie de code porteur différents :

- 1 nouveau processus peut entrainer une renégociation SSL complète
- 1 nouveau tab n'entrainera pas forcément une renégociation SSL
- On parle pour la renégociation SSL de « false start » ou de « full renégociation »
    - le terme « renégociation SSL » est souvent mal utilisé :
        - Les navigateurs et les serveurs font tout pour ne pas avoir à négocier et à renégocier un canal SSL
        - Dans les cas où l'un ou l'autre sont forcés de renégocier, tous les navigateurs ont implémenté des « optimisations »
            - La plus connue étant le « false start » de Chrome
            - Dans ces cas, les cache de clé de session (symétrique) sont intensivement utilisés coté client et coté serveur
        - Du coup, on parle de « Full renegociation » pour désigner la reprise complète d'une négociation sans utilisation de cache ou d'optimisation
    - Ces mécanismes sont autant de sources possibles de comportements différents vis-à-vis de la saisie du code porteur

11.4.1.1.1 Cas Mozilla Firefox sous Windows

Mozilla Firefox sous Windows utilise un processus global pour gérer tous les tabs :

1 tab :

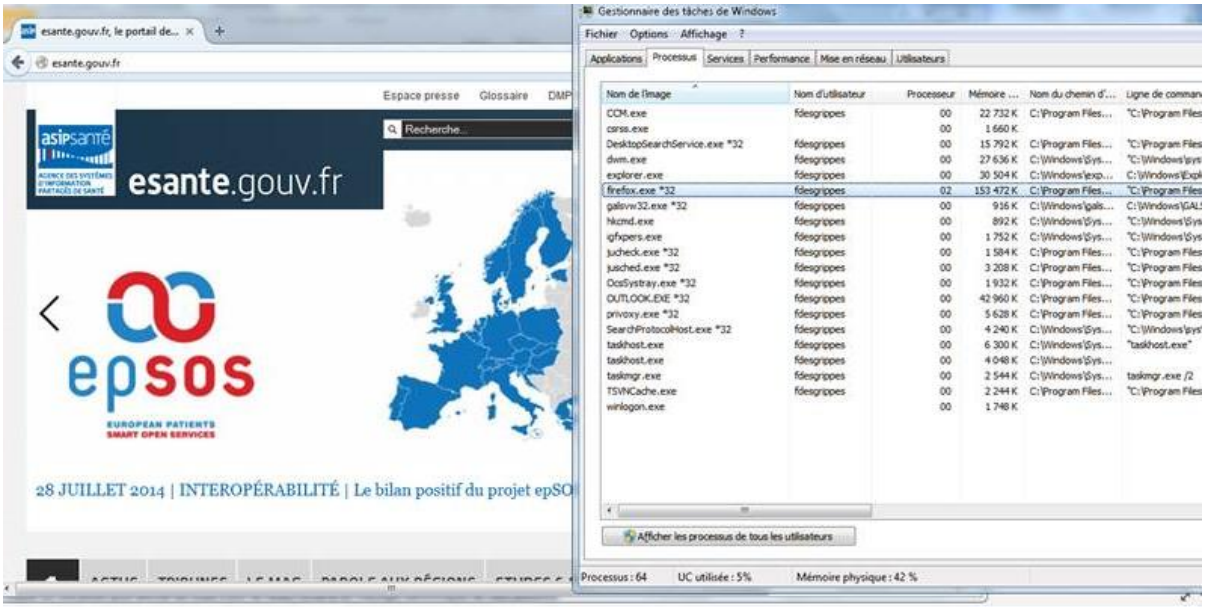


Figure 15 : Mozilla Firefox : 1 tab et l'analyse de processus afférente

2 tab :

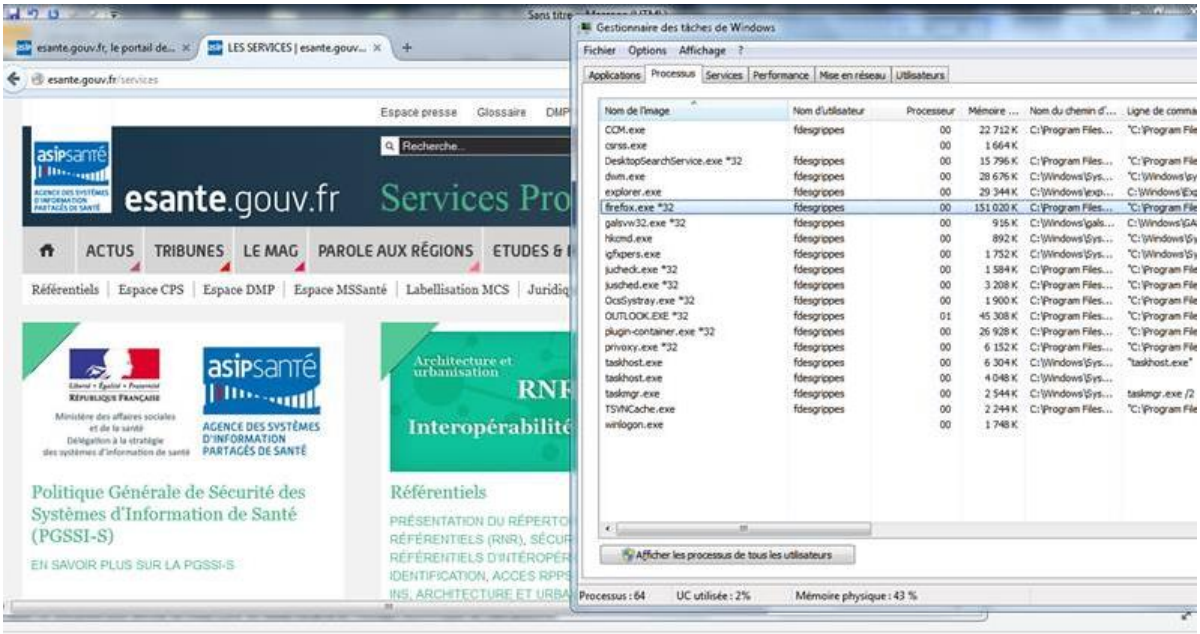


Figure 16: Mozilla Firefox : 2 tab et l'analyse de processus afférente

Mais ce n'est pas le cas général.

11.4.1.1.2 Cas Microsoft Internet Explorer

Pour Internet Explorer, les comportements changent en fonction

- Des options de sécu activées (Protected mode, Enhanced Protected mode, Enhanced Protected mode en 64b…) et de la plateforme (x86, x64, win7, win8, win8.1)
- Du nombre de tabs
    - La courbe (tab / process) implémentée par IE n'est pas linéaire
        - Il y a donc des différences entre une situation avec peu de tab et avec beaucoup de tab sous IE
            - avec la Cryptolib v5
            - du point de vue des renégociations SSL

La gestion des processus implémentée par Internet Explorer est complètement différente.

Les explications sont là (doc de référence: http://blogs.msdn.com/b/ieinternals/archive/2012/03/23/understanding-ie10-enhanced-protected-mode-network-security-addons-cookies-metro-desktop.aspx)

1 processus « père » : « le manager process » qui s'exécute toujours dans l'archi du système :

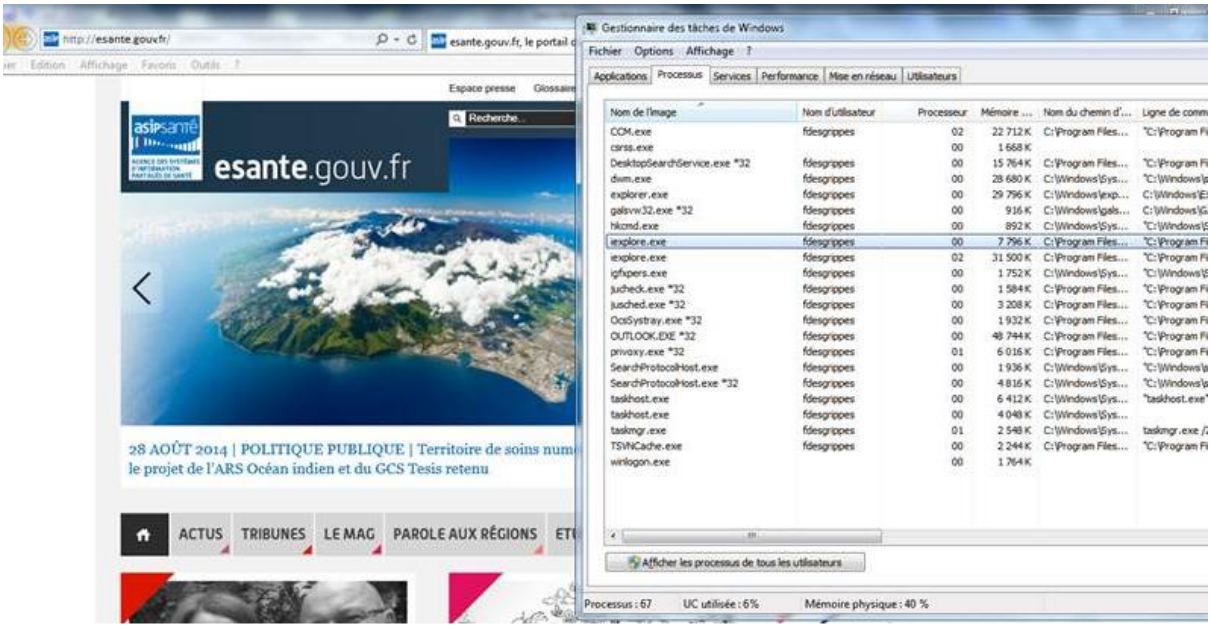


Figure 17 : Microsoft Internet Explorer: processus « manager process » dans l'architecture du système

Des processus fils : les « process manager » qui s'exécutent par défaut en 32 bit.

⇨ Dans un premier temps : 1 processus par « process manager » supplémentaire

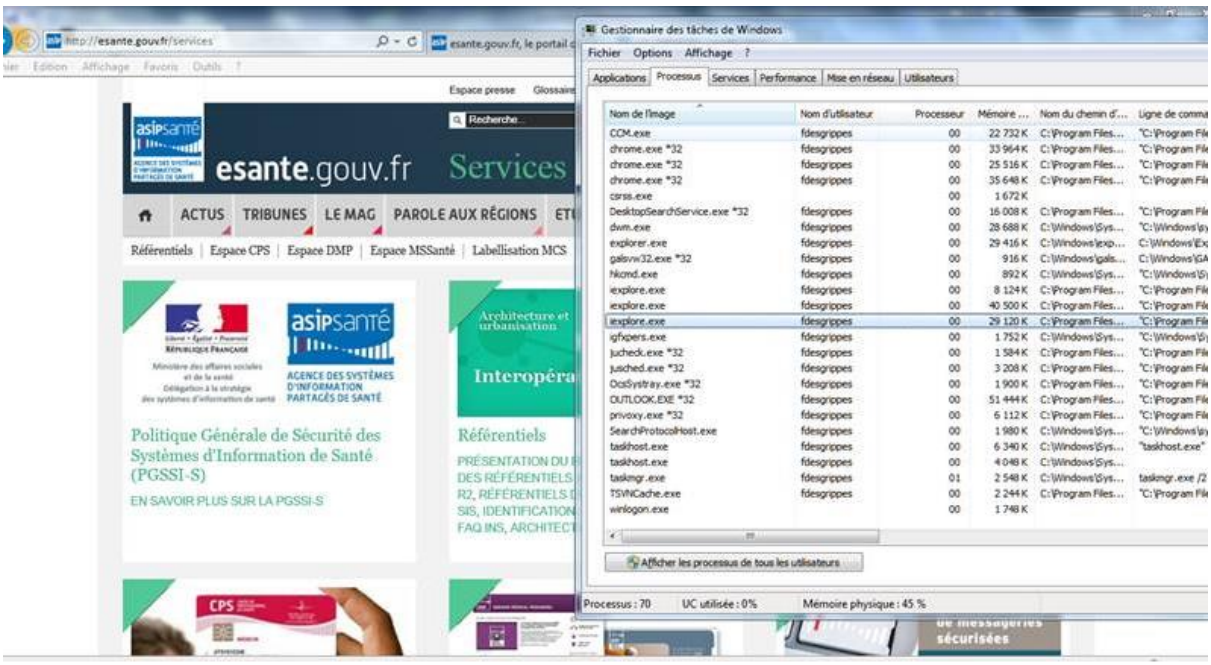
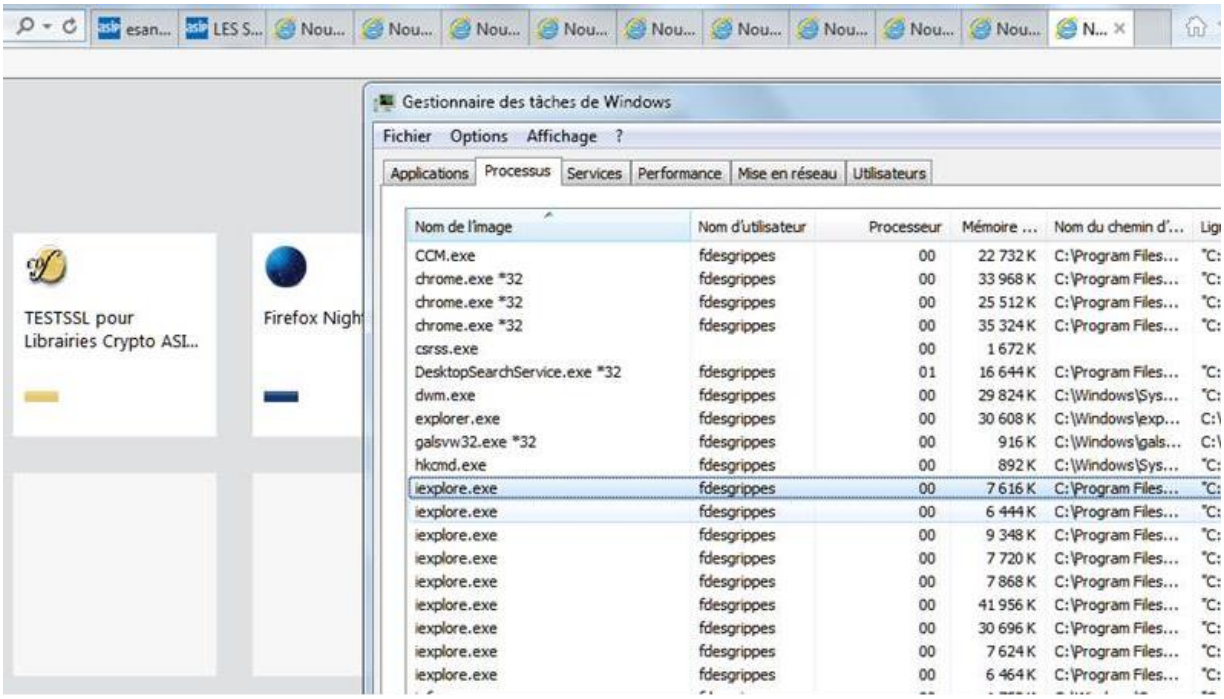


Figure 18 : Microsoft Internet Explorer: 1 tab et l'analyse de processus « processus manager »

Mais, Microsoft implémente une courbe de mise à l'échelle, ici pour 12 tab, on obtient 1 + 9 processus :



Nom de l'image	Nom d'utilisateur	Processeur	Mémoire ...	Nom du chemin d'...
CCM.exe	fdesgrippes	00	22 732 K	C:\Program Files...
chrome.exe *32	fdesgrippes	00	33 968 K	C:\Program Files...
chrome.exe *32	fdesgrippes	00	25 512 K	C:\Program Files...
chrome.exe *32	fdesgrippes	00	35 324 K	C:\Program Files...
csrss.exe		00	1 672 K	
DesktopSearchService.exe *32	fdesgrippes	01	16 644 K	C:\Program Files...
dwm.exe	fdesgrippes	00	29 824 K	C:\Windows\Sys...
explorer.exe	fdesgrippes	00	30 608 K	C:\Windows\exp...
galsvw32.exe *32	fdesgrippes	00	916 K	C:\Windows\gals...
hkcmd.exe	fdesgrippes	00	892 K	C:\Windows\Sys...
iexplore.exe	fdesgrippes	00	7 616 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	6 444 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	9 348 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	7 720 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	7 868 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	41 956 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	30 696 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	7 624 K	C:\Program Files...
iexplore.exe	fdesgrippes	00	6 464 K	C:\Program Files...

Figure 19 : Microsoft Internet Explorer: 12 tab et l'analyse de processus « processus manager »

Sous Windows / IE, la relation « IE / tab / processes » est paramétrée par la clé:

HKCU\Software\Microsoft\Internet Explorer\Main

TabProcGrowth

Cf. “Opening a New Tab may launch a New Process with Internet Explorer 8.0”

http://blogs.msdn.com/b/askie/archive/2009/03/09/opening-a-new-tab-may-launch-a-new-process-with-internet-explorer-8-0.aspx)

11.4.1.1.3 Cas Google Chrome sous Microsoft Windows

1 processus père + 1 tab par processus :

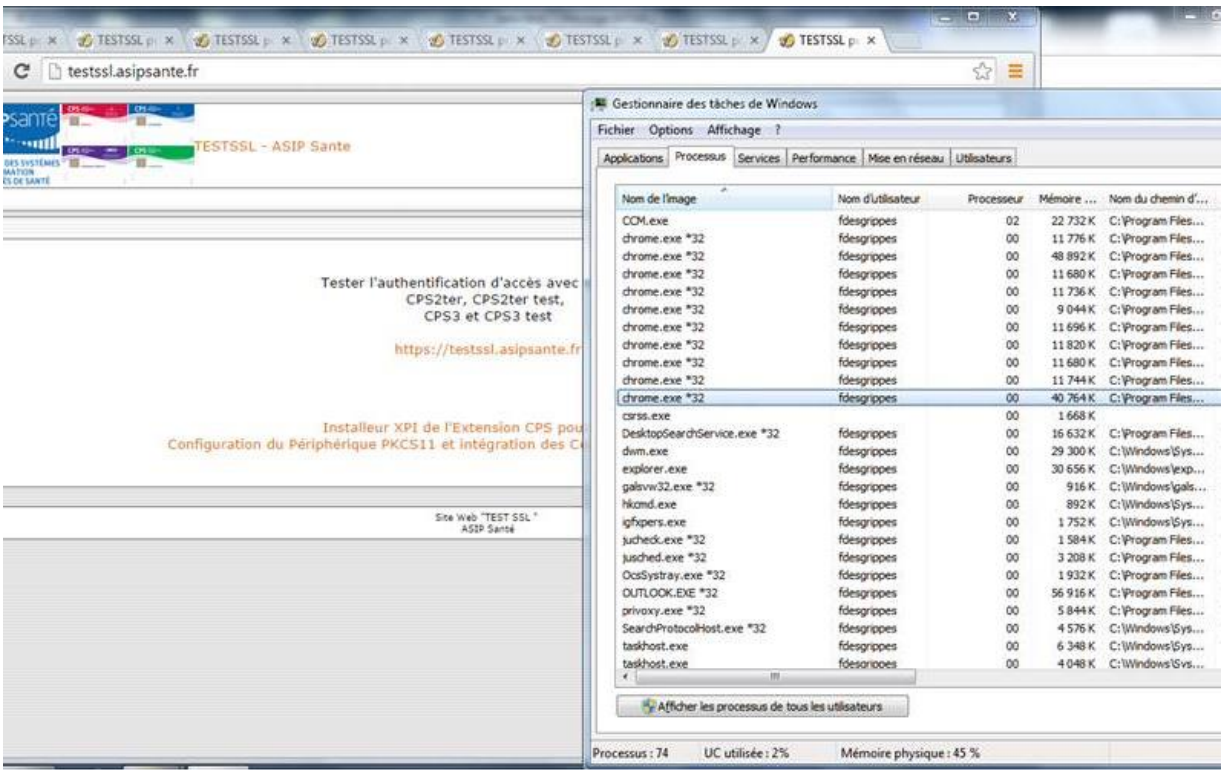


Figure 20 : Google Chrome: 1 tab et l'analyse de processus afférente : 10 tab = 10 processus

11.4.1.2 Cas particulier des navigateurs Web avec plugin Java

Dans le cas où une application utilise Internet Explorer pour l'authentification par carte par exemple et une applet Java pour l'utilisation des fonctions de la carte par la suite, 2 processus (1 pour Internet Explorer modulo ce qui vient d'être expliqué dans le paragraphe précédent, 1 autre pour la VM java) accèdent à la carte : 1 double saisie de code porteur est probable.

11.4.2 Boîte de saisie du code porteur hors DLL PKCS#11

Impact 6		

Boîte de saisie du code porteur hors DLL PKCS#11

Jusqu'aux Cryptolib CPS v4, dans les environnements Windows la boîte de dialogue de saisie du code porteur était embarquée dans la librairie PKCS#11.

Cette boîte de dialogue a été retirée de la DLL PKCS#11 et déportée dans le CSP.

Les applications qui utilisent la DLL PKCS#11 ne peuvent donc plus compter sur la boîte de demande de code porteur en passant « NULL » à la fonction C_Login.

Remarque : sous Mac OS X et Linux, il n'était déjà pas possible de passer « NULL » au C_Login avec la Cryptolib CPS v4.

Solution

Revoir l'architecture de la solution

Pour savoir si le code porteur est actif pour l'application en cours d'exécution, il faut obtenir les informations de la session via la fonction C_GetSessionInfo et vérifier le champ state de la structure CK_SESSION_INFO.

Cf. **MANUEL_PROG**

Cf. **8.1.2 Algorithme de présentation du code porteur au niveau PKCS#11**

Applications impactées

Applications accédant à la carte via PKCS#11 (Ex. : applications utilisant Java / IAIK)

Tableau 25 : Impact : Changement de la gestion du code porteur

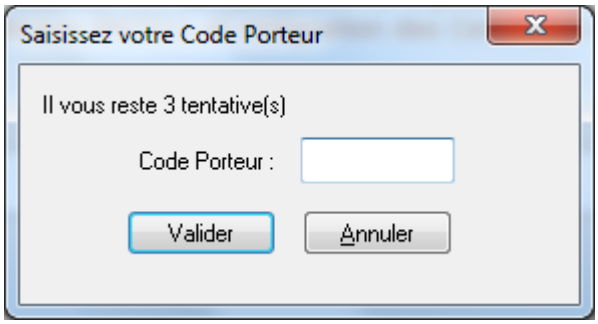
11.4.3 Changement d'aspect de la fenêtre de saisie du code porteur sous Windows

Impact 7		

Des informations ont été ajoutées à la fenêtre de saisie du code porteur sous Windows

Solution

Cryptolib CPS v4 :



Cryptolib CPS v5 :

La nouvelle boîte de dialogue de saisie du code porteur reprend les améliorations apportées par la Cryptolib CPS v4 Full PC/SC et ajoute l'affichage de la version de la Cryptolib CPS installée et du CSP instancié.

Si le mode EPM est activé, la mention « EPM » apparaît.

Applications impactées

Applications faisant appel au CSP.

Tableau 26 : Impact : Changement de fenêtre de saisie du code porteur sous Windows

11.5 Calcul de signature

Impact 8		

Changement dans les mécanismes de signature

Pour signer un document, la Cryptolib CPS v4 acceptait le scénario suivant :

- Calcul de SHA1 du document hors Cryptolib CPS
- Soumission du SHA1 obtenu pour signature à la couche PKCS#11 via CKM_RSA_PKCS

Ce scénario n'est plus possible avec la Cryptolib CPS v5 : il faut utiliser CKM_SHA1_RSA_PKCS.

Solution

Revoir l'implémentation de la solution

Appeler la couche PKCS#11 avec CKM_SHA1_RSA_PKCS.

Cf. **MANUEL_PROGMANUEL_PROG**

Applications impactées

Applications accédant à la carte via PKCS#11 pour signer des documents.

Tableau 27 : Impact : Changement dans les mécanismes de signature

11.6 Changement de comportement de C_WaitForSlotEvent

Impact 9		

Changement de comportement de C_WaitForSlotEvent

L'implémentation de la fonction C_WaitForSlotEvent dans les librairies PKCS#11 de la Cryptolib CPS v5 a dû être modifiée du fait de problèmes de performances rencontrés avec le navigateur Firefox.

Solution

Revoir l'implémentation de la solution

En v5, C_WaitForSlotEvent retourne CKR_FUNCTION_NOT_SUPPORTED en mode "bloquant".

Cf. **ARRACH_CPS**

Applications impactées

Applications accédant à la carte via PKCS#11.

Tableau 28 : Impact : Changement dans les mécanismes de signature

11.7 L'API CPS est dépréciée au profit de l'interface PKCS#11

Impact 10		

L'API CPS est dépréciée au profit de l'interface PKCS#11

Solution

Revoir l'implémentation de la solution

Cf. stratégie de migration v4 vers v5 plus haut, rationaliser les filières d'accès cartes, se référer au tableau de correspondance API CPS / PKCS#11 plus bas.

Cf**. 9.3 Exemple 1 : Chemin de migration v4 / v5 pour une application utilisant l'API CPS**

Cf. **16 Migration de l'API CPS vers l'API PKCS#11 de la Cryptolib CPS v5**

Applications impactées

Applications accédant à la carte via l'API CPS.

Tableau 29 : Impact : Dépréciation de l'API CPS au profit de l'interface PKCS#11

12Description des modifications CSP

12.1 Propagation des certificats

Impact 11		

Propagation des certificats

Sous Windows 7, la propagation des certificats CPS vers le magasin Windows est activée par défaut.

En conséquence :

- le CCM n'est pas indispensable pour l'alimentation du magasin
- le CCM reste indispensable si l'élimination des certificats est requise en cas de retrait de la carte
- 2 processus (svhost.exe et CCM.exe) alimentent le même magasin depuis la même carte en parallèle

Rappel : le CCM n'est pas utilisé par Mozilla Firefox (alimentation du magasin de certificats Firefox via le périphérique de sécurité et le PKCS#11 associé).

Solution

Vérifier adéquation de la solution avec cette implémentation

Applications impactées

Toutes les applications.

Tableau 30 : Impact : Propagation des certificats

13Changement d'installeurs

13.1 Système 64bit avec Cryptolib CPS 64bit

Impact 12

Nouvelles DLL 64bit pour systèmes Windows 64bit

L'installeur 64bit installe les DLL 64bit et 32bit.

Les outils CCM et CPS-Gestion sont portés en 64bit.

Solution

Revoir l'installation et le déploiement de la solution

Vérifier l'implémentation de la solution

Cf. **MANUEL_INST_UTIL**

Il est vivement **conseillé** d'utiliser l'installeur 64bit sur un système 64bit.

L'installeur 64bit installant les DLL de la Cryptolib CPS v5 en versions 64bit et 32bit, **les deux CSP 32bit et 64bit sont présents et déclarés dans le système**.

Sur un système 64bit de type Win7, la filière Cryptolib 64bit est chargée, a minima, par lsass.exe :

Description du service	Nom du service	Nom du processus
Isolation de clé CNG	KeyIso	%WINDIR%\system32\lsass.exe
Netlogon	Netlogon	%WINDIR%\system32\lsass.exe
Emplacement protégé	ProtectedStorage	%WINDIR%\system32\lsass.exe
Gestionnaire de comptes de sécurité	SamSs	%WINDIR%\system32\lsass.exe
Agent de stratégie IPsec	PolicyAgent	%WINDIR%\system32\lsass.exe

Tableau 31 : Impact : Nouvelles DLL 64bit pour systèmes 64bit

et par les applications 64bit utilisant les cartes CPx.

La filière Cryptolib 32bit peut être chargée par les applications 32bit (SysWOW64) : deux CSP Cryptolib sont chargés en parallèle.

- ⇨ Impact sur la prise de trace
- ⇨ Impact sur le nombre de sessions PKCS#11
- ⇨ Impact sur le chemin des DLL chargées

Applications impactées

Applications n'utilisant pas ODI.
Applications utilisant ODI : vérifier roadmap ODI pour évolution installation Cryptolib v5.

Tableau 32 : Impact : Nouvelles DLL 64bit pour systèmes 64bit

13.2 Installations Full PC/SC et GALSS factorisées

Impact 13		

Un seul installeur pour les deux filières

Un seul installeur Cryptolib CPS v5 pour les deux filières (GALSS et Full PC/SC), au lieu de deux installeurs Cryptolib CPS v4 (un pour la filière GALSS, un autre pour le Full PC/SC).

Solution

Revoir l'installation et le déploiement de la solution

Applications impactées

Applications n'utilisant pas ODI.
Applications utilisant ODI : vérifier roadmap ODI pour évolution installation Cryptolib CPS v5.

Tableau 33 : Impact : Installations GALSS et PC/SC factorisées

13.3 Possibilité de personnaliser les installations

Impact 14

Installations paramétrables

Avec la Cryptolib CPS v5, les installations MSI sont paramétrables en ligne de commande

Solution

Cf. **MANUEL_INST_UTIL**

L'installeur MSI installe automatiquement les composants v4 Full PC/SC :

- si le GALSS n'est pas présent (le fichier galsvw32.exe ne doit pas être dans %WINDIR%)
- **et** si la Cryptolib CPS v4 Full PC/SC est présente

Dans tous les autres cas, ce sont les composants v4 GALSS qui sont installés par défaut.
Il est possible de « composer » les paramètres disponibles.

Vérifier le paramétrage proposé par défaut :

L'installation par défaut correspond à l'installation du CCM en mode de détection manuel, en fixant la fréquence de détection des évènements lecteurs en mode automatique à 2 sec et en fixant la fréquence de détection des évènements lecteurs en mode manuel à 600 sec :

```
msiexec /i CryptolibCPS-xx.yy.zz.msi DETECTIONMODE = 0 WATCHONTIMER = 2
WATCHOFFTIMER = 600 [/quiet]
```

Revoir l'installation et le déploiement de la solution, par exemple :

- Installer les composants en mode détection automatique:
  `msiexec /i CryptolibCPS-xx.yy.zz.msi DETECTIONMODE = 1 [/quiet]`
- Installer les composants en fixant la fréquence de détection des évènements lecteurs en mode automatique :
  `msiexec /i CryptolibCPS-xx.yy.zz.msi WATCHONTIMER = 2 [/quiet]`
- Installer les composants en fixant la fréquence de détection des évènements lecteurs en mode manuel :
  `msiexec /i CryptolibCPS-xx.yy.zz.msi WATCHOFFTIMER = 600 [/quiet]`
- Restaurer en ligne de commande :
  `msiexec /i CryptolibCPS-xx.yy.zz.msi RESTAURE = 1 [/quiet]`
- Installer les composants v4 GALSS:
  `msiexec /i CryptolibCPS-xx.yy.zz.msi CPS2ter = 1 [/quiet]`
- Installer les composants v4 Full PC/SC:
  `msiexec /i CryptolibCPS-xx.yy.zz.msi CPS2ter = 2 [/quiet]`

Applications impactées

Applications n'utilisant pas ODI.
Applications utilisant leurs installeurs, appelant les installeurs ASIP Santé.
Applications utilisant ODI : vérifier roadmap ODI pour évolution installation Cryptolib CPS v5.

Tableau 34 : Impact : Installations paramétrables

13.4 Configuration de la Cryptolib CPS

Impact 15 

La manière de configurer la Cryptolib CPS v5 est différente de celle de la Cryptolib CPS v4.

La configuration permet principalement de fixer un niveau de traces.

La Cryptolib CPS v4 se configure via un fichier unique :
%ALLUSERSPROFILE%\santesocial\CPS\coffre\cps_pkcs11_safe.ini
(ou cps_pkcs11_pcsc.ini, en fonction de la filière utilisée)

La Cryptolib CPS v5 se configure, en fonction des OS, via :

- la base de registre
- des variables d'environnement
- des fichiers de configuration

Solution

Revoir la façon de configurer la Cryptolib CPS

Sous Windows, les .reg d'activation et de désactivation des traces de la Cryptolib CPS v5 sont fournis. Par défaut, les traces sont désactivées (configuration attendue en production).

Cf. **MANUEL_INST_UTIL**

Applications impactées

Toutes applications.

Tableau 35 : Impact : Configuration des traces

13.5 Nouvelle version de CPS-Gestion

Impact 16

Une nouvelle version de CPS-Gestion est fournie avec la Cryptolib CPS v5

La version de CPS-Gestion fournie avec la Cryptolib CPS v4 s'appuie sur l'API CPS.

La nouvelle version fournie avec la Cryptolib CPS v5 s'appuie sur la librairie PKCS#11.

Solution

Cryptolib CPS v5 : deux versions du programme CPS-Gestion sont fournies avec cette Cryptolib.

Exemple sous Windows :

- **CPS-Gestion v6.xx** : version s'appuyant sur l'API PKCS#11, disponible par défaut, accessible depuis le menu « **Démarrer** » > « **Programmes** » > « **Santé Social** » > « **CPS** » > « **Gestionnaire de la carte CPS** »
- **CPS-Gestion v5.xx** : même version que celle fournie avec la Cryptolib CPS v4, s'appuyant sur l'API CPS, à utiliser pour vérifier l'installation, accessible de la façon suivante (dépendant de l'architecture de l'OS) :
  - **%ProgramFiles%\santesocial\CPS\cpgesw32_CPS2ter.exe**
  - **%ProgramFiles(x86)%\santesocial\CPS\cpgesw32_CPS2ter.exe**

Cf. **MANUEL_INST_UTIL**

Applications impactées

CPS-Gestion.

Tableau 36 : Impact : Nouvelle version de CPS-Gestion

13.6 Changement des menus Démarrer sous Windows

Impact 17		

Les menus créés par l'installeur MSI sous Windows ont changé

Solution

Le menu :
Cryptolib CPS v4 : « Démarrer > Programmes > Cryptolib CPS »

Est remplacé par :
Cryptolib CPS v5 : « Démarrer > Programmes > Santé Social > CPS »

Applications impactées

CCM (« Gestionnaire de certificats CPS ») et CPS-Gestion (« Gestionnaire de la carte CPS »).

Tableau 37 : Impact : Changement des menus Démarrer sous Windows

13.7 Changement de nom du package RPM Linux

Impact 18		

Le nom du package RPM sous Linux a changé

Solution

Le nom de package « cryptolib »:
Cryptolib CPS v4 : `rpm -qa | grep -i cryptolib`

Est remplacé par « Cryptolib »
Cryptolib CPS v5 : `rpm -qa | grep -i Cryptolib`

Applications impactées

Toutes applications.

Tableau 38 : Impact : Changement de nom de package RPM Linux

13.8 Changement de nom du Tokend sous Mac OS X

Impact 19		

Le nom du Tokend sous Mac OS X a changé

Solution

Cryptolib CPS v4 : `GIP-CPS.tokend`

Cryptolib CPS v5 : `CPS3.tokend`

Applications impactées

Applications accédant à la carte via le Tokend.

Tableau 39 : Impact : Changement de nom du Tokend sous Mac OS X

14Gestion du cache des fichiers carte

Impact 20

Le fichier de cache des certificats de la carte :
Cryptolib CPS v4 : …\santesocial\CPS\coffre\ccert.bin

Est remplacé par un répertoire de cache des fichiers de la carte :
Cryptolib CPS v5 : …\santesocial\CPS\cache\

Solution

Revoir l'installation et le déploiement de la solution

Exemple sous Windows :

Les mécanismes de réplication du cache de certificats par copie de ccert.bin sont impactés (Profils itinérants…). Il est nécessaire de recopier plusieurs fichiers pour l'implémenter à l'identique avec la Cryptolib CPS v5.

Il est nécessaire d'accorder des droits d'accès en R/W sur le répertoire :
%ALLUSERSPROFILE%\santesocial\CPS\cache\
au lieu du fichier :
%ALLUSERSPROFILE%\santesocial\CPS\coffre\ccert.bin

Il est nécessaire d'accorder des droits d'accès en R/W sur le répertoire :
%USERPROFILE%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Virtualized\C\ProgramData\santesocial\cps\cache\
pour Internet Explorer en mode protégé.

Le mécanisme de cache via ccert.bin est maintenu en parallèle pour les applications utilisant la Cryptolib CPS v4.

La Cryptolib CPS v5.0.3 utilise un format de cache incompatible avec celui des versions Cryptolib CPS v5.0.6 et supérieures :

- Les Cryptolib CPS v5.0.6 et supérieures recréent seules les caches corrompues ou qu'elles ne comprennent pas
- La Cryptolib CPS v5.0.3 ne le fait pas et ne fonctionne pas avec des fichiers de cache créés par les Cryptolib CPS v5.0.6 et supérieures : il faut donc effacer ces fichiers manuellement pour refaire fonctionner une Cryptolib CPS v5.0.3.

Cf. **MANUEL_INST_UTIL**

Applications impactées

Toutes applications.

Tableau 40 : Impact : Gestion du cache des fichiers carte

15Données métiers

15.1 Changement des données métiers

Impact 21		

Les objets métiers ont changé entre le volet CPS2ter et le volet IAS.

Solution

Revoir l'installation et le déploiement de la solution

Exemple de différences :

- Tags E3 à E8 en CPS2ter, Tags E3, E4, E5, ED et EE en IAS
- Dans la structure « Activité, Situation d'exercice » (tags E8 ou EE) : disparition du tag 88 de CPS2ter et apparition du tag 86 en IAS

Cf. **DONNEES_METIER**

Applications impactées

Toutes applications.

Tableau 41 : Impact : Changements de données métiers

15.2 Conseils sur les méthodes d'accès aux données du dico

La Cryptolib CPS v5 continue d'apporter et d'installer une API et une implémentation d'accès aux données du dictionnaire (« DICO »).

A terme :

- les données (quantités, valeurs..) sont amenées à changer
    - du fait notamment des évolutions constatées des métiers « Santé Social »
- le format du fichier DICO-FR.GIP sera sans doute amené à changer
    - du fait notamment du point précédent
- la manière de récupérer ces informations (fichier versus webservices) pourrait changer

Les 2 DLLs cptabw32.dll et cptabw64.dll (cette dernière étant fournie à partir de la Cryptolib CPS v5) sont actuellement utilisées par CPS Gestion pour exploiter les données de DICO-FR.GIP.

Un premier conseil : ne pas « embarquer » les données du dico dans le code.

Un second conseil : préparer les logiciels en les concevant de manière à anticiper un changement de grammaire de DICO-FR.GIP et en apportant ses propres implémentations de lecture de ce type de données, par exemple en implémentant le canevas DAO (Data Access Object pattern).

Ce type de préparation/migration peut être organisé sur le modèle de ce qui est présenté plus haut pour l'API CPS.

15.3 Conseils sur l'exploitation des données métiers

15.3.1 Exemple : exploitation de "CPS_ID_CARD"

Le code suivant exploite la donnée "CPS_ID_CARD" pour extraire la "Catégorie_Carte".

Si l'intention est bonne, ce type de code n'est pas acceptable. La colonne "Commentaire" explique pourquoi.

#	Code	Commentaire
0	`import org.bouncycastle.asn1.ASN1InputStream;`	
1	`import org.bouncycastle.asn1.DERApplicationSpecific;`	Conseil: pas besoin de Bouncycastle et de toute sa hiérarchie de classes (2MO) pour exploiter les TLV contenant les données métiers de la carte CPx !
2	`import org.bouncycastle.asn1.DEROctetString;`	
3	`import org.bouncycastle.asn1.DERSequence;`	
4	`import org.bouncycastle.asn1.DERTaggedObject;`	
5	`[...]`	
6	`byte[] data = new byte[1048576];`	1048576 = 1024*1024 bytes alloués alors que la structure "CPS_ID_CARD" contient 31 bytes. Conseil: en fonction de la méthode de lecture de cette donnée (PKCS#11, APDU…), se préoccuper de l'allocation, même en Java!
7	`[...]`	
8	`// read "CPS_ID_CARD" structure and put it in the data bytearray here`	Lecture de la donnée "CPS_ID_CARD" depuis la carte
9	`[...]`	
10	`ASN1InputStream ais = new ASN1InputStream(data);`	Début de l'extraction de la "Catégorie_Carte"
11	`DERApplicationSpecific o1 = (DERApplicationSpecific)ais.readObject();`	
12	`[...]`	
13	`DERSequence cpsIdCard = (DERSequence) o1.getObject(IMPLICIT_TAG_SEQUENCE);`	IMPLICIT_TAG_SEQUENCE = 16 (ne pas mettre "16" en dure…)
14	`DERTaggedObject categorieCarteTlv = (DERTaggedObject)seq.getObjectAt(2);`	Offset 2 = 3ième Tag dans **DONNEES_METIER**. Conseil : Récupérer le sous-tag par la valeur du tag (ici : recherche du tag 0x82)
15	`DEROctetString categorieCarteValue = (DEROctetString) tgo.getObject();`	
16	`[...]`	

Tableau 42 : Contre-exemple d'exploitation des données « CPS_ID_CARD »

16Migration de l'API CPS vers l'API PKCS#11 de la Cryptolib CPS v5

16.1 Documents de référence

Les documents de référence permettant de migrer de l'API CPS vers l'API PKCS#11 de la Cryptolib CPS v5 sont:

Code	Titre	Fichier	Date
MANUEL_PROG	Cryptolib CPS v5 Manuel de Programmation	ASIP-PUSC-PSCE_MP_Cryptolib-CPS-v5-Manuel-de-programmation_20131016_v1.5.0.pdf	24/07/2013
API_CPS	APIs Des Services Carte du Professionnel de Santé	win32504_0.zip\CW320504\doc\cpmpracp.doc	
DONNEES_METIER	Les données métier de la CPS3 Volets CPS2ter et IAS	ASIP_CPS3_Données-métier_v1.0.2.pdf	
PKCS11_EXT	CryptoLib CPS3 Spécifications externes du module PKCS#11	SpécificationsExternes_PKCS11_CryptoLib_CPS3_v1.0.1.pdf	

16.2 Migration

#	Fonction API CPS	Ordinal	Description	Equivalent Cryptolib CPS v5
01	CPS_OuvertureSession	@1	Signaler un numéro de connexion logique au Gestionnaire d'Accès (GALSS) en utilisant un nom logique qui attribue au doublet Application-LAD (coupleur-fente). Ce numéro est conservé jusqu'à la fermeture de la session et doit être fourni en paramètre pour toutes les autres APIs d'accès à cette ressource LAD	La Cryptolib CPS v5 abstrait les types de lecteurs PSS ou PC/SC. En conséquence, la notion de session « GALSS » disparait. La Cryptolib CPS v5 reprend la notion de session introduite par le standard PKCS#11. Cette session n'est pas une session lecteur GALSS, mais une session cryptographique indépendante du support physique permettant d'effectuer des opérations sécurisées.

#	Fonction API CPS	Ordinal	Description	Equivalent Cryptolib CPS v5
02	CPS_FermetureSession	@2	Libérer les ressources qui lui étaient assignées dans les tables internes du gestionnaire et de provoquer la fermeture de la session Application - Ressource	Idem point précédent.
03	CPS_TestSession	@3	Connaître le nom de l'application éventuellement en session exclusive avec une ressource	Idem point précédent.
04	CPS_IntroductionCarte	@4	Effectuer l'introduction d'une carte dans la fente du lecteur	Remplacé par les appels PKCS#11 : C_Initialize C_OpenSession C_Login
05	CPS_RetraitCarte	@5	Mettre hors tension d'une carte dans le lecteur	Remplacé par les appels PKCS#11 : C_CloseSession C_Logout C_Finalize
06	CPS_TestPresenceCarte	@6	Tester la présence d'une carte dans le lecteur et retourne son état ainsi que, le cas échéant, son « ATR »	Remplacé par les appels PKCS#11 : C_Initialize C_GetSlotList C_GetSlotInfo (C_FindObjectsInit, C_FindObjects, C_FindObjectsFinal)
07	CPS_InformationsCarte	@7	Retourner les informations caractérisant une carte	Remplacé par l'interprétation du retour de l'appel PKCS#11 C_GetAttributeValue("CPS_ID_CARD")
08	CPS_InformationsPorteur	@8	Récupérer les informations de base concernant le porteur de la carte	Remplacé par l'interprétation du retour de l'appel PKCS#11 C_GetAttributeValue("CPS_NAME_PS")
09	CPS_ListeApplications	@9	Récupérer la liste, par groupe de 16 maximum, des applications disponibles dans la carte CPS	Non remplacé car non utilisée

#	Fonction API CPS	Ordinal	Description	Equivalent Cryptolib CPS v5
10	CPS_LecSituationPS	@11	Récupérer les données d'une situation d'exercice	Remplacé par l'interprétation du retour de l'appel PKCS#11 C_GetAttributeValue("CPS_ACTIVITY_XX_PS") avec XX entre 01 et 16
11	CPS_LecSitFacturation	@12	Lire par groupe de 8 au plus les situations de facturation du Professionnel de Santé	Remplacé par l'interprétation du retour de l'appel PKCS#11 C_GetAttributeValue("CPS2TER_SIT_FACT") en Cryptolib CPS v4
				Non remplacé en Cryptolib CPS v5 : passer par les API FSV du GIE-SV
12	CPS_PresCodePorteur	@14	Présenter un code porteur à la carte	Remplacé par l'appel PKCS#11 C_Login
13	CPS_ModifCodePorteur	@15	Modifier le code porteur associé à une carte	Remplacé par l'appel PKCS#11 C_SetPin
14	CPS_RecycleCodePorteur	@16	Recycler du code porteur par présentation du super code	Remplacé par les appels PKCS#11 : C_Login avec le code porteur SO : présente le code PUK à la carte C_InitPIN en passant le nouveau code porteur : initialise le code porteur de l'utilisateur C_Logout : met fin à la session loguée SO
15	CPS_LectureTopoCarte	@17	API interne	Non remplacé
16	CPS_GenereDefiAuthen	@18	Générer une valeur aléatoire utilisable comme valeur de défi dans la procédure d'authentification de la CPS. Cette API n'effectue pas d'accès à la carte.	Remplacé par l'appel PKCS#11 C_GenerateRandom
17	CPS_CondenseMessage	@20	Calculer un condensat sur un message en vue d'une signature	Remplacé par les appels PKCS#11 : C_DigestInit C_Digest C_DigestUpdate C_DigestFinal Rappel la CPS3 n'accepte pas de signer un condensat

#	Fonction API CPS	Ordinal	Description	Equivalent Cryptolib CPS v5
18	CPS_CondenseFichier	@28	Calculer un condensât sur un fichier en vue d'une signature	Remplacé par les appels PKCS#11 : C_DigestInit C_Digest C_DigestUpdate C_DigestFinal Rappel la CPS3 n'accepte pas de signer un condensat
19	CPS_DirSituationPS	@33	Récupérer la liste des identifiants de fichiers de situations d'exercice du PS présents sur la CPS	Non remplacé
20	CPS_CalculAuthen2	@34	Provoquer le déclenchement du mécanisme d'authentification de la CPS sur un défi fourni de l'extérieur. Cette API nécessite que le code porteur soit actif.	Remplacé par les appels PKCS#11 : C_SignInit C_Sign C_SignUpdate C_SignFinal
21	CPS_CalculSign2	@35	Provoquer le déclenchement du mécanisme de signature de la CPS sur un condensât.	Remplacé par les appels PKCS#11 : C_SignInit C_Sign C_SignUpdate C_SignFinal
22	CPS_VerifAuthen2	@36	Provoquer le déclenchement des mécanismes de vérification du résultat issu d'une authentification.	Remplacé par les appels PKCS#11 : C_VerifyInit C_Verify C_VerifyUpdate C_VerifyFinal
23	CPS_VerifSign2	@37	Provoquer le déclenchement des mécanismes de vérification du résultat issu d'une signature	Remplacé par les appels PKCS#11 : C_VerifyInit C_Verify C_VerifyUpdate C_VerifyFinal

#	Fonction API CPS	Ordinal	Description	Equivalent Cryptolib CPS v5
24	CPS_LectureX509	@38	Récupérer un certificat X509 qu'elle construit à partir des données lues dans la carte. Le certificat est renvoyé à l'application sous la forme d'une unique chaîne d'octets	Remplacé par l'interprétation du retour de l'appel PKCS#11 : C_GetAttributeValue("Certificat d'Authentification CPS") ou C_GetAttributeValue("Certificat de Signature CPS") Voir plus haut les recommandations sur l'utilisation des usages de clés.
25	CPS_VerificationX509	@39	Provoquer le mécanisme de vérification de la signature du certificat X509, à partir de la clé publique reçue de l'appelant	Non remplacé au sein de la Cryptolib CPS v5
				Remplacé les appels système correspondant
26	CPS_InterpretationX509_CPS	@40	Extraire les données d'un certificat présenté et supposé bien issu à l'origine d'une CPS. Il est de la responsabilité de l'application de vérifier ensuite si les valeurs de ces champs certifiés correspondent effectivement à celles espérées (exemple : contrôle du numéro de carte…)	Non remplacé au sein de la Cryptolib CPS v5
				Remplacé par l'interprétation des données métiers ou de l'ASN.1 des certificats à la discrétion des applications (API système / framework disponibles).
27	CPS_Genere_Alea_CPS	@41	Générer une valeur aléatoire via la carte CPS (qui pourra par exemple servir de clé de session de chiffrement) (le code porteur doit être actif)	Remplacé par l'interprétation du retour de l'appel PKCS#11 C_GenerateRandom
28	CPS_InformationsPS2	@42	Récupérer les informations concernant le professionnel de santé, porteur de la carte	Remplacé par l'interprétation du retour de l'appel PKCS#11 C_GetAttributeValue("CPS_INFO_PS")
29	CPS_InformationPoste	@138	Récupérer une image de la configuration du poste, à savoir les versions des différents composants du kit CPS, les types et versions du système d'exploitation, de la machine et le contenu du fichier GALSS.INI. Cette API peut retourner des codes d'erreur d'accès au fichier GALSS.INI	Non remplacé au sein de la Cryptolib CPS v5
				A remplacer par les appels systèmes équivalents (WMIC, rpm, …)

Tableau 43 : Migration API CPS vers API PKCS#11

17Annexe – Fiche d'évaluation de migration

Projet :			
Impact	**Solution**		**Observations**
1	-	☐	
2	-	☐	
3	-	☐	
4		☐	
5	-	☐	
6	-	☐	
7	-	☐	
8	-	☐	
9	-	☐	
10	-	☐	
11	-	☐	
12	-	☐	
13	-	☐	
14	-	☐	
15	-	☐	
16	-	☐	
17	-	☐	
18	-	☐	
19	-	☐	

Projet :			
Impact	**Solution**		**Observations**
20	-	☐	
21	-	☐	

Tableau 44 : Fiche d'évaluation de migration

18Annexe – Table des figures

19Annexe – Liste des tableaux

20Notes

[fin du document]

Agence des systèmes d'information partagés de santé
9, rue Georges Pitard - 75015 Paris
Tel : 01 58 45 32 50
esante.gouv.fr